

# SEKE

## 2010

**Proceedings of the  
Twenty-Second  
International Conference on  
Software Engineering &  
Knowledge Engineering**

**San Francisco Bay  
July 1-3, 2010**



**PROCEEDINGS**

# **SEKE 2010**

**The 22<sup>nd</sup> International Conference  
on Software Engineering &  
Knowledge Engineering**

**Sponsored by**

Knowledge Systems Institute Graduate School, USA

**Technical Program**

**July 1-3, 2010**

**Hotel Sofitel, Redwood City, San Francisco Bay, USA**

**Organized by**

Knowledge Systems Institute Graduate School

Copyright © 2010 by Knowledge Systems Institute Graduate School

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of the publisher.

ISBN-10: 1-891706-26-8 (paper)  
ISBN-13: 978-1-891706-26-4

Additional Copies can be ordered from:  
Knowledge Systems Institute Graduate School  
3420 Main Street  
Skokie, IL 60076, USA  
Tel:+1-847-679-3135  
Fax:+1-847-679-3166  
Email:office@ksi.edu  
<http://www.ksi.edu>

Proceedings preparation, editing and printing are sponsored by  
Knowledge Systems Institute Graduate School

Printed by Knowledge Systems Institute Graduate School

# SEKE2010 Foreword

On behalf of the Program Committee Co-Chairs and the Program Committee of the 2010 International Conference on Software Engineering and Knowledge Engineering (SEKE2010), we sincerely welcome you to contribute and attend SEKE-2010 in San Francisco, California, USA.

The International Conference on Software Engineering and Knowledge Engineering has entered its 22nd year. In the past twenty-one years, the International Conference on Software Engineering and Knowledge Engineering has provided a unique and important forum for academic and industrial researchers and practitioners to exchange research ideas, results and application experience in software engineering and knowledge engineering.

This year's technical program is prepared and organized by a great team of Program Co-Chairs, who are listed below.

## **Program Co-Chairs:**

- Marek Reformat, University of Alberta, Canada
- S. Masoud Sadjadi, Florida International University, USA
- Du Zhang, California State University Sacramento, USA

It has been my great honor and pleasure to serve as the SEKE2010 Program Committee Chair and work with this great team and the program committee members to prepare a rich and solid technical program as well as the high quality conference proceedings. The published conference proceedings contain the papers accepted and selected for presentation at SEKE2010 based on a rigorous review process. I hope it will serve as a valuable reference for the research community in the coming years.

We received an overwhelming 253 submissions from 32 countries. The acceptance rate for full papers is 33%, and that for short papers is 29%. This year, authors from 32 countries (Algeria, Argentina, Australia, Austria, Brazil, Canada, China, Colombia, Egypt, France, Germany, India, Iran, Israel, Italy, Japan, Jordan, Malaysia, Mexico, Morocco, Netherlands, Poland, Saudi Arabia, South Korea, Spain, Sweden, Switzerland, Taiwan, Tunisia, Turkey, United Kingdom, and United States) will present papers at the conference.

As the Program Chair for this Conference, I am very grateful to have this opportunity to work with three distinguished SEKE2010 Program Committee Co-Chairs and the committed program committee members. Their excellent support and prompt review efforts led to the successful organization of SEKE2010 technical program. I want to extend my sincere and deepest thanks to Dr. Xiaoying Bai, Dr. Shihong Huang as the Publicity Co-Chairs, Dr. Hironori Washizaki as the Asia Liaison, Dr. Jose Carlos Maldonado as the South America Liaison. My appreciation also goes to the keynote speakers and special address presenter for sharing their visions, insights, and experiences with the conference attendee about emergent technologies and trends, research topics and issues in both academic research and industry applications. Moreover, I like to express my appreciation to the organizers of the workshops and special sessions for their great contributions and supporting efforts to make SEKE2010 a great success. In addition, I would like to thank Dr. S. K. Chang, the Steering Committee Chair, Dr. Guido Wirtz, the Conference Chair, and Dr. Guenther Ruhe, the Honorary Conference Chair, for their excellent guidance throughout the conference preparation process. Last but not the least; we owe a special gratitude to Mr. Daniel Li and Mr. Alec Liu from the Knowledge Systems Institute for their great efforts and timely support.

Finally, I truly hope you will enjoy the technical program of SEKE2010, have productive discussion, great presentation and networking. Of course, I sincerely hope you will all explore and enjoy the unique summer weather and various attractions in San Francisco area.

Jerry Gao  
SEKE2010 Program Chair

# **The 22<sup>nd</sup> International Conference on Software Engineering & Knowledge Engineering (SEKE 2010)**

**July 1-3, 2010  
Hotel Sofitel, Redwood City, San Francisco Bay, USA**

## **Conference Organization**

### **Steering Committee Chair**

*Shi-Kuo Chang, University of Pittsburgh, USA*

### **Steering Committee**

*Vic Basili, University of Maryland, USA*

*Bruce Buchanan, University of Pittsburgh, USA*

*C. V. Ramamoorthy, University of California, Berkeley, USA*

### **Conference Chair**

*Guido Wirtz, Bamberg University, Germany*

### **Honorary Conference Chair**

*Guenther Ruhe, University of Calgary, Canada*

### **Program Chair**

*Jerry Gao, San Jose State University, USA*

## Program Co-Chairs

**Marek Reformat**, *University of Alberta, Canada*  
**S. Masoud Sadjadi**, *Florida International University, USA*  
**Du Zhang**, *California State University, USA*

## Advisory Committee

**Daniel Cooke**, *Texas Tech University, USA*  
**Natalia Juristo**, *Madrid Technological University, Spain*  
**Taghi Khoshgoftaar**, *Florida Atlantic University, USA*  
**Guenther Ruhe**, *University of Calgary, Canada*

## Program Committee

**Alain Abran**, *Ecole de technologie superieure - Universite du Quebec, Canada*  
**Silvia Teresita Acuna**, *Universidad Autonoma de Madrid, Spain*  
**Taiseera Albalushi**, *Sultan Qaboos University, Oman*  
**Edward Allen**, *Mississippi State University, United States*  
**Doo-hwan Bae**, *KAIST, South Korea*  
**Ebrahim Bagheri**, *National Research Council Canada, Canada*  
**Rami Bahsoon**, *University of Birmingham, United Kingdom*  
**Xiaoying Bai**, *Tsinghua University, China*  
**Maria Teresa Baldassarre**, *University of Bari, Italy*  
**Purushotham Bangalore**, *University of Alabama at Birmingham, United States*  
**Fevzi Belli**, *Univ. Paderborn, Germany*  
**Nicolas Belloir**, *University of Pau, France*  
**Alessandro Bianchi**, *Department of Informatics - University of Bari, Italy*  
**James Bieman**, *Colorado State University, United States*  
**Jean-michel Bruel Bruel**, *IRIT / Universite de Toulouse, France*  
**Barrett Bryant**, *University of Alabama at Birmingham, United States*  
**Kai-yuan Cai**, *Beijing University of Aeronautics and Astronautics, China*  
**Danilo Caivano**, *Department of Informatics - University of Bari, Italy*  
**Gerardo Canfora**, *Universita del Sannio, Italy*  
**Giovanni Cantone**, *University of Roma Tor Vergata - DISP, Italy*  
**Jeffrey Carver**, *University of Alabama, United States*  
**Jaelson Castro**, *Universidade Federal de Pernambuco - UFPE, Brazil*  
**Christine Chan**, *University of Regina, Canada*  
**Keith Chan**, *The Hong Kong Polytechnic University, Hong Kong*  
**Kuang-nan Chang**, *Eastern Kentucky University, United States*  
**Ned Chapin**, *InfoSci Inc., United States*  
**Shu-ching Chen**, *Florida International University, United States*  
**Zhenyu Chen**, *Software Institute, Nanjing University, China*  
**Yung-pin Cheng**, *Dept. of CS, National Taiwan Normal Univ., Taiwan*  
**Yoonsik Cheon**, *University of Texas at El Paso, United States*  
**Peter Clarke**, *Florida International University, United States*  
**Nelly Condori-fernandez**, *Valencia University of Valencia, Spain*  
**Panos Constantopoulos**, *Athens University of Economics & Business, Greece*  
**Daniel Cooke**, *Texas Tech University, United States*

**Kendra Cooper**, *UT-Dallas, United States*  
**Maria Francesca Costabile**, *University of Bari, Italy*  
**Karl Cox**, *University of Brighton, United Kingdom*  
**Jose Luis Cuadrado**, *University of Alcala, Spain*  
**Juan J. Cuadrado-gallego**, *University of Alcala, Spain*  
**Alfredo Cuzzocrea**, *ICAR-CNR and University of Calabria, Italy*  
**Jose Luis De La Vara**, *Universidad Politecnica de Valencia, Spain*  
**Deepak Dhungana, Lero**, *The Irish Software Engineering Research Centre., Ireland*  
**Massimiliano Di Penta**, *University of Sannio, Italy*  
**Scott Dick**, *University of Alberta, Canada*  
**Teresa Diez**, *University of Alcala, Spain*  
**Maria Dominguez**, *University of Alcala, Spain*  
**Jin Song Dong**, *NUS, Singapore*  
**Jing Dong**, *University of Texas at Dallas, United States*  
**Dirk Draheim**, *University of Innsbruck, Austria*  
**Philippe Dugerdil**, *HEG - Univ. of Applied Sciences, Switzerland*  
**Schahram Dustdar**, *TU Vienna, Austria*  
**Christof Ebert**, *Vector Consulting Services, Germany*  
**Raimund Ege**, *NIU, United States*  
**Magdalini Eirinaki**, *Computer Engineering Dept, San Jose State University, United States*  
**Faezeh Ensan**, *University of New Brunswick, Canada*  
**Onyeka Ezenwoye**, *South Dakota State University, United States*  
**Davide Falessi**, *University of Rome, TorVergata, Italy*  
**Behrouz Far**, *University of Calgary, Canada*  
**Robert Feldt**, *Chalmers University of Technology, Sweden*  
**Eduardo B. Fernandez**, *Florida Atlantic University, United States*  
**Marian Fernandez De Sevilla**, *University of Alcala, Spain*  
**Renata Fortes**, *University of Sao Paulo, Brazil*  
**Jerry Gao**, *San Jose State University, United States*  
**Kehan Gao**, *Eastern Connecticut State University, United States*  
**Alessandro Garcia**, *PUC-Rio, United States*  
**Felix Garcia**, *University of Castilla-La Mancha, Spain*  
**Ignacio Garcia Rodriguez De Guzman**, *University of Castilla-La Mancha, Spain*  
**Raul Garcia-castro**, *Universidad Politecnica de Madrid, Spain*  
**Holger Giese**, *Hasso Plattner Institute at the University of Potsdam, Germany*  
**Itana Gimenes**, *Universidade Estadual de Maringá, Brazil*  
**Swapna Gokhale**, *Univ. of Connecticut, United States*  
**Wolfgang Golubski**, *Zwickau University of Applied Sciences, Germany*  
**Jeff Gray**, *University of Alabama, United States*  
**Des Greer**, *Queen's University Belfast, United Kingdom*  
**Eric Gregoire**, *Universite d'Artois, France*  
**Christiane Gresse Von Wangenheim**, *UFSC - Federal University of Santa Catarina, Brazil*  
**Xudong He**, *Florida International University, United States*  
**Miguel Herranz**, *University of Alcala, Spain*  
**Mong Fong Horng**, *National Kaohsiung University of Applied Sciences, Taiwan*  
**Shihong Huang**, *Florida Atlantic University, United States*  
**Hoh Peter In**, *Korea University, South Korea*  
**Clinton Jeffery**, *University of Idaho, United States*  
**Jason Jung**, *Yeungnam University, South Korea*  
**Natalia Juristo**, *Universidad Politecnica de Madrid, Spain*  
**Taghi Khoshgoftaar**, *Florida Atlantic University, United States*  
**Sascha Konrad**, *Siemens Corporate Research, United States*

**Gunes Koru**, *UMBC, United States*  
**Nicholas Kraft**, *The University of Alabama, United States*  
**Dhananjay Kulkarni**, *Boston University, United States*  
**Vinay Kulkarni**, *Tata Consultancy Services, India*  
**Gihwon Kwon**, *Kyonggi University, South Korea*  
**Mark Last**, *Ben-Gurion University, Israel*  
**Konstantin Laufer**, *Loyola University Chicago, United States*  
**Jeff Lei**, *University of Texas at Arlington, United States*  
**Bixin Li**, *School of Computer Science and Engineering, Southeast University, China*  
**Ming Li**, *Nanjing University, China*  
**Tao Li**, *Florida International University, United States*  
**Chien-hung Liu**, *National Taipei University of Technology, Taiwan*  
**Shih-hsi Liu**, *California State University, Fresno, United States*  
**Xiaodong Liu**, *Edinburgh Napier University, United Kingdom*  
**Yan Liu**, *Pacific Northwest National Laboratory, United States*  
**Yi Liu**, *GCSU, United States*  
**Hakim Lounis**, *UQAM, Canada*  
**Joan Lu**, *University of Huddersfield, United Kingdom*  
**Heiko Ludwig**, *IBM Research, United States*  
**Jose Carlos Maldonado**, *ICMC-USP, Brazil*  
**Antonio Mana**, *University of Malaga, Spain*  
**Miriam Martinez**, *University of Alcalá, Spain*  
**Hong Mei**, *Peking University, China*  
**Hsing Mei**, *Fu Jen Catholic University, Taiwan*  
**Emilia Mendes**, *University of Auckland, New Zealand*  
**Ali Mili**, *NJIT, United States*  
**Ana M. Moreno**, *Universidad Politécnica de Madrid, Spain*  
**Henry Muccini**, *University of L'Aquila, Italy*  
**Enriqueta Muel**, *Universidad de Alcalá, Spain*  
**Martin Neil**, *Queen Mary, University of London, United Kingdom*  
**Kia Ng**, *ICSRiM - University of Leeds, United Kingdom*  
**Ngoc Thanh Nguyen**, *Wroclaw University of Technology, Poland*  
**Allen Nikora**, *Jet Propulsion Laboratory, United States*  
**Mehmet Orgun**, *Macquarie University, Australia*  
**Eric Pardede**, *La Trobe University, Australia*  
**Witold Pedrycz Pedrycz**, *University of Alberta, Canada*  
**Jun Peng**, *Chongqing University of Science and Technology, China*  
**Antonio Piccinno**, *University of Bari, Italy*  
**Alfonso Pierantonio**, *University of L'Aquila, Italy*  
**Rick Rabiser**, *Johannes Kepler University, Austria*  
**Damith C. Rajapakse**, *National University of Singapore, Singapore*  
**Rajeev Raje**, *IUPUI, United States*  
**Jose Angel Ramos**, *Universidad Politecnica de Madrid, Spain*  
**Marek Reformat**, *University of Alberta, Canada*  
**Robert Reynolds**, *Wayne State University, United States*  
**Daniel Rodriguez**, *The university of Alcalá, Spain*  
**Guenther Ruhe**, *U of Calgary, Canada*  
**Samira Sadaoui**, *University of Regina, Canada*  
**Masoud Sadjadi**, *Florida International University, United States*  
**Ramon Sagarna**, *The University of Birmingham, United Kingdom*  
**Salvatore Alessandro Sarcia**, *University of Rome "Tor Vergata", Italy*  
**Kamran Sartipi**, *McMaster University, Canada*

**Douglas Schmidt**, *Vanderbilt University ISIS, United States*  
**Andreas Schoenberger**, *Distributed and Mobile Systems Group, University of Bamberg, Germany*  
**Naeem (jim) Seliya**, *University of Michigan - Dearborn, United States*  
**Tony Shan**, *Keane Inc, United States*  
**Rajan Shankaran**, *Macquarie University, Australia*  
**Yi-dong Shen**, *Institute of software/Chinese academy of sciences, China*  
**Michael Shin**, *Computer Science/Texas Tech University, United States*  
**George Spanoudakis**, *City University, United Kingdom*  
**Xiao Su**, *San Jose State University, United States*  
**Rajesh Subramanyan**, *Siemens Corporate Research, United States*  
**Jeff Tian**, *Southern Methodist University, United States*  
**Genny Tortora**, *University of Salerno, Italy*  
**Mark Trakhtenbrot**, *Holon Institute of Technology, Israel*  
**Peter Troger**, *Humboldt University of Berlin, Germany*  
**T.h. Tse**, *The University of Hong Kong, Hong Kong*  
**Michael Vanhilst**, *Florida Atlantic University, United States*  
**Sira Vegas**, *Universidad Politecnica de Madrid, Spain*  
**Silvia Vergilio**, *Federal University of Parana (UFPR), Brazil*  
**Arndt Von Staa**, *PUC-Rio, Brazil*  
**Huanjing Wang**, *Western Kentucky University, United States*  
**Qianxiang Wang**, *Peking University, China*  
**Hironori Washizaki**, *Waseda University, Japan*  
**Victor Winter**, *University of Nebraska at Omaha, United States*  
**Guido Wirtz**, *Distributed Systems Group, Bamberg University, Germany*  
**Eric Wong**, *University of Texas at Dallas, United States*  
**Franz Wotawa**, *TU Graz, Austria*  
**Hui Wu**, *University of Alabama at Birmingham, United States*  
**Haiping Xu**, *University of Massachusetts Dartmouth, United States*  
**Chi-lu Yang**, *Taiwan Semiconductor Manufacturing Company Ltd., Taiwan*  
**Hongji Yang**, *De Montfort University, United Kingdom*  
**Huiqun Yu**, *East China University of Science and Technology, China*  
**Cui Zhang**, *California State University Sacramento, United States*  
**Du Zhang**, *California State University, United States*  
**Jing Zhang**, *Motorola Inc., United States*  
**Min-ling Zhang**, *College of Computer and Information Engineering, Hohai University, China*  
**Zhenyu Zhang**, *The University of Hong Kong, Hong Kong*  
**Zhinan Zhou**, *Samsung Telecommunications America, United States*  
**Hong Zhu**, *Oxford Brookes University, United Kingdom*  
**Xingquan Zhu**, *Florida Atlantic University, United States*  
**Eugenio Zimeo**, *University of Sannio, Italy*  
**Andrea Zisman**, *City University London, United Kingdom*

## Publicity Co-Chairs

**Xiaoying Bai**, *Tsinghua University, China*  
**Shihong Huang**, *Florida Atlantic University, USA*

## **Asia Liasion**

**Hironori Washizaki**, *Waseda University, Japan*

## **South America Liasion**

**Jose Carlos Maldonado**, *University of Sao Paulo, Brazil*

## **Industry Advisory Committee**

**Terry Potter**, *USA*

**A. J. Rhem**, *USA*

**Ajay Sharma**, *USA*

**Maria Trujillo**, *USA*

**Kirti Vaidya**, *USA*

## **Proceedings Cover Design**

**Gabriel Smith**, *Knowledge Systems Institute Graduate School, USA*

## **Conference Secretariat**

**Judy Pan**, *Chair, Knowledge Systems Institute Graduate School, USA*

**Dennis Chi**, *Knowledge Systems Institute Graduate School, USA*

**Chen-Cheang Huang**, *Knowledge Systems Institute Graduate School, USA*

**Chih-Fong Liu**, *Knowledge Systems Institute Graduate School, USA*

# Table of Contents

<b>Foreword</b> .....	iii
<b>Conference Organization</b> .....	iv
<b>Prodigious Data, Logic, Processing, and Usage</b>	
<i>Dr. Alfred Z. Spector</i> .....	1
<b>Building A Smarter Planet With University Collaboration: Empowering People Through Information Integration</b>	
<i>Dr. Josephine M. Cheng</i> .....	2
<b>The multi-core programming challenge</b>	
<i>Dr. Daniel Cooke</i> .....	3
<b>Future Research Directions for Software Engineering and Knowledge Engineering</b>	
<i>Dr. Guenther Ruhe</i> .....	5
<b>Machine Learning with Value-Based Software Engineering</b>	
Absent features or missing values? <i>Wen Zhang, Ye Yang, Qing Wang</i> .....	6
Capturing Antagonistic Stakeholder Value Propositions in Value-Based Software Development <i>Du Zhang</i> .....	12
The effects of human-computer interaction modes for weak learners in an animation learning environment <i>Yu-Fang Yeh</i> .....	18

## Software Requirement Engineering

Quality Indicators in Requirements Elicitation (S)  
*Aneesh Krishna, Andreas Gregoriades, Chattrakul Sombattheera* ..... 24

A UML Profile Oriented to the Requirements Collecting and Analyzing for the  
Multi-Agent Systems Project  
*Gilleanes Thorwald Araujo Guedes, Rosa Maria Vicari* ..... 28

Business-Object Oriented Requirements Analysis Framework for Data Warehouses  
(S)  
*Anirban Sarkar, Sankhayan Choudhury, Nabendu Chaki, Swapan Bhattacharya* 34

Soft Systems in Requirements Engineering: A Case Study (S)  
*Alejandra Yopez Lopez, Nan Niu* ..... 38

Textual Software Requirements Specifications in the Context of Software  
Architecting (S)  
*Matthias Galster, Armin Eberlein, Mahmood Moussavi* ..... 42

Conditions for ignoring failures based on a requirements model  
*João Pimentel, Emanuel Santos, Jaelson Castro* ..... 48

## Software Validation

Ontology-Driven Enterprise Application Integration  
*G. Bucci, V. Sandrucci, E. Vicario* ..... 54

Introduction of Time and Timing Variability in Usage Model based Testing  
*Sebastian Siegl, Reinhard German, Kai-Steffen Hielscher* ..... 61

FLAT A Fast Lattice-Based Algorithm for Test Suite Reduction (S)  
*Ahmed Raafat Abuzeid, Haitham S. Hamza, Ismail Abdel Hamid Taha* ..... 67

## Interoperability and Semantic Web technologies

Semantic Document Architecture for Desktop Data Integration and Management  
*Saša Nešić, Dragan Gašević, Mehdi Jazayeri* ..... 73

Using the whole structure of ontology for semantic relatedness measurement (S)  
*Ehsan KhounSiavash, Ahmad Baraani-Dastjerdi* ..... 79

Composer-Science: A Semantic Service Based framework for Workflow  
Composition in e-Science Projects (S)  
*Laryssa Machado da Silva, Regina Braga, Fernanda Campos* ..... 84

## Software Quality Assurance

A Stochastic Model for Optimizing the Patching Time of Software Bugs (S)  
*Yong Wang, Dianxiang Xu, William M Lively, Dick B. Simmons* ..... 88

Do More People Make the Code More Defect Prone?: Social Network Analysis in  
OSS Projects  
*Salifu Alhassan, Bora Çağlayan, Ayşe Bener* ..... 93

Performance Analysis of a Web Server with Dynamic Thread Pool Architecture  
*Jijun Lu, Swapna S. Gokhale* ..... 99

Some Improvements for More Precise Model Checking  
*Zhi Zhang, Qingkai Zeng, Ming Huang* ..... 106

Software Development Effort and Quality Prediction Using Bayesian Nets and  
Small Local Qualitative Data  
*Łukasz Radliński* ..... 113

Multi-Tracker Collaboration in Bittorrent Systems  
*Sonia Gulrajani, Anuja Oka, Xiao Su* ..... 117

An Evaluation of Tie-Breaking Strategies for Fault Localization Techniques  
*Xiaofeng Xu, Vidroha Debroy, W. Eric Wong, Donghui Guo* ..... 123

## Software Measurement

An Ontology Model to Support the Automated Evaluation of Software <i>Raúl García-Castro, Miguel Esteban-Gutiérrez, Mick Kerrigan, Stephan Grimm</i>	129
A Review of Parametric Effort Estimation Models for the Software Project Planning Process <i>Pablo Rodríguez-Soria, J.J. Cuadrado-Gallego, J.A. Gutiérrez de Mesa, Borja Martín-Herrera</i>	135
Information-Theoretic Metrics for Project-Level Scattering and Tangling <i>Erik Linstead, Lindsey Hughes, Cristina Lopes, Pierre Baldi</i>	141
Synchronization Complexity Metric (S) <i>Peter Yastrebenetsky, Mark Trakhtenbrot</i>	147
Measurement Model of Software Requirements Derived from System Maintainability Requirements <i>Alain Abran, Khalid T. Al-Sarayreh, Juan J. Cuadrado-Gallego</i>	153

## Software Architectures

Evolution Styles to Capitalize Evolution Expertise within Software Architectures (S) <i>Oliver Le Goer, Dalila Tamzalit, Mourad Oussalah</i>	159
Reasoning about Attribute Architectures <i>Tacksoo Im, John D. McGregor</i>	165
Formal Specification of Software Architecture Security Tactics (S) <i>Andrew Wyeth, Cui Zhang</i>	172
Enterprise Systems Development: Impact of Aspect Oriented Software Architecture (S) <i>Pawan Kumar Verma, Deepak Dahiya</i>	176

## Software Test Automation, Practice, and Standardization

Reducing Black-box Test Suite Using Input Parameter Relationships  
*Lixin Wang* ..... 180

Introducing Automated Environment Configuration Testing in an Industrial Setting  
*Caryna Pinheiro, Vahid Garousi, Frank Maurer, Jonathan Sillito* ..... 186

An Ontology-based Software Test Generation Framework  
*Valeh H. Nasser, Weichang Du, Dawn MacIsaac* ..... 192

Automated Integration Testing and Verification of a Secured SOA Infrastructure -  
an Experience Report in eHealth  
*Mario Bernhart, Thomas Artner, Andreas Mauczka, Thomas Grechenig* ..... 198

## Software Engineering with Computational Intelligence and Machine Learning

A Novel Software Metric Selection Technique Using the Area Under ROC Curves  
*Taghi M. Khoshgoftaar, Kehan Gao* ..... 203

Automatic Bug Triage using Semi-Supervised Text Classification  
*Jifeng Xuan, He Jiang, Zhilei Ren, Jun Yan, Zhongxuan Luo* ..... 209

Ensemble Feature Selection Technique for Software Quality Classification  
*Huanjing Wang, Taghi M. Khoshgoftaar, Kehan Gao* ..... 215

## Web Services

Negotiating Software Acquisition Supported by Web Services in a Distributed  
Software Development Process (S)  
*Gabriel Costa Silva, Itana Maria de Souza Gimenes, Marcelo Fantinato, Maria  
Beatriz Felgar de Toledo* ..... 221

Transforming Service-Oriented Business Models into Web Service Specifications  
*Hugo Estrada, Itzel Morales-Ramirez, Alicia Martinez, Oscar Pastor* ..... 225

Reliable Web Service Selection based on Transactional Risk (S) <i>Ying Yin, Xizhe Zhang, Bin Zhang</i> .....	231
Selecting Web Services for Choreography Implementation: Compatibility Checking Approach with Access Control <i>Emad Elabd, Emmanuel Coquery, Mohand-Said Hacid</i> .....	235
A Model-driven Approach to Flexible Multi-Level Customization of SaaS Applications <i>Zakwan Jaroucheh, Xiaodong Liu, Sally Smith</i> .....	241
Weaving Functional and Non-Functional Attributes for Dynamic Web Service Composition (S) <i>Ajay Bansal, Srividya Kona, M. Brian Blake, Gopal Gupta</i> .....	247
<b>Software Regression Testing</b>	
Improving Cluster Selection Techniques of Regression Testing by Slice Filtering <i>Yongwei Duan, Zhenyu Chen, Zhihong Zhao, Ju Qian, Zhongjun Yang</i> .....	253
A Constrained Particle Swarm Optimization Approach for Test Case Selection <i>Luciano S. de Souza, Ricardo B. C. Prudêncio, Flavia de A. Barros</i> .....	259
Software Defect Estimation using Support Vector Regression (S) <i>Roberta A. A. Fagundes, Renata M.C.R. de Souza</i> .....	265
Analyzing the Relationship of Process Metrics And Classified Changes - A Pilot Study (S) <i>Andreas Mauczka, Mario Bernhart, Thomas Grechenig</i> .....	269
Cost-Effective Combinatorial Test Case Prioritization for Varying Combination Weights <i>Ziyuan Wang, Baowen Xu, Lin Chen, Zhenyu Chen</i> .....	273
<b>E-Commerce and Mobile Commerce</b>	
A Multi-State Bayesian Network for Skill Verification in Online Auctions <i>Ankit Goel, Haiping Xu, Sol M. Shatz</i> .....	279

An Empirical Evaluation on the Relationship Between Final Auction Price and Shilling Activity in Online Auctions <i>Fei Dong, Sol M. Shatz, Haiping Xu</i> .....	286
---	-----

## Software Framework and Application Tools

A Framework for Solar Energy Applications Photovoltaic Systems (S) <i>Papatella, F., Carvalho, T., Zarate, L., Pereira, E., Song, M.</i> .....	292
---	-----

Data manipulation API in ERP systems (S) <i>Vadym Borovskiy, Wolfgang Koch, Alexander Zeier</i> .....	298
--	-----

Ontology-Based Tools in the Service of Hardware Verification <i>Eyal Bin, Alaa Ghanayim, Karen Holtz, Eitan Marcus, Ronny Morad, Ofer Peled, Michal Rimon, Gil Shurek, Elena Tsanko</i> .....	303
--	-----

## Knowledge Engineering

Smarter Software Engineering: Knowledge factors contributing to improved Individual Performance <i>Narayanan Srinivasaraghavan, Craig McDonald, John Campbell</i> .....	309
--	-----

Lattice-Context Based Digital Paper Search (S) <i>Chongyang Shi, Zhendong Niu, Xiyi Cheng</i> .....	315
--	-----

Evaluating the Weighted Sum Algorithm for Estimating Conditional Probabilities in Bayesian Networks <i>Simon Backer, Emilia Mendes</i> .....	319
---	-----

An Ontology-based Mapping Repository for Meta-querier Customization <i>Xiao Li, Randy Chow</i> .....	325
---	-----

Knowledge Engineering to Visualize Complexity for Legacy Modernization Planning (S) <i>Sarah B. Lee, Sajjan G. Shiva, K. S. Braunsdorf</i> .....	331
---	-----

Using QVT for adapting question analysis to restricted domain QA systems (S) <i>Katia Vila, Jose-Norberto Mazón, Antonio Ferrández</i> .....	335
---	-----

## Software Maintenance and Evolution

Towards an Automation of Software Evolution Good Practices  
*Chouki Tibermacine, Soraya Sakhraoui, Vincent Le Gloahec, Régis Fleurquin, Salah Sadou* ..... 339

VESTA: A View-based Software Quality Assessment Model for Software Evolution Management (S)  
*Wei-Chung Hu, Chia Hung Kao, Feng Pu Yang, Hewijin Christine Jiau, Kuo-Feng Ssu* ..... 345

Detecting Emergent Behavior in Distributed Systems Using Scenario-Based Specifications  
*Mohammad Moshirpour, Abdolmajid Mousavi, Behrouz H. Far* ..... 349

## Human-Computer Interface and Interaction

MMWA-ae: boosting knowledge from Multimodal Interface Design, Reuse and Usability Evaluation  
*Americo Talarico Neto, Renata Pontin M. Fortes, Rafael Rossi, Solange Rezende* 355

Human-Computer Interface Design Guidelines: An Expert System (S)  
*Tiago Cinto, Cecilia Sosa Arias Peixoto* ..... 361

Assisting Developers to Read Code Help-Documents Efficiently through Discovering Document-section Relationships  
*Lijie Wang, Leye Wang, Ge Li, Bing Xie* ..... 367

## Empirical Software Engineering and Software Economics

Validity Threats in Empirical Software Engineering Research - An Initial Survey  
*Robert Feldt, Ana Magazinius* ..... 374

A Comparative Study of Attribute Weighting Techniques for Software Defect Prediction Using Case-based Reasoning  
*Elham Paikari, Michael M. Richter, Guenther Ruhe* ..... 380

Software Project Portfolio Selection A Modern Portfolio Theory Based Technique <i>Hélio R. Costa, Márcio O. Barros, Ana Regina Rocha</i> .....	387
---	-----

## Software Validation and Verification

Runtime Constraint Checking Approaches for OCL, A Critical Comparison (S) <i>Carmen Avila, Amritam Sarcar, Yoonsik Cheon, Cesar Yeep</i> .....	393
---	-----

Refinement Checking for Interface Automata with Z Notation <i>Zining Cao</i> .....	399
---	-----

Multi-objective Genetic Algorithms: Construction and Recombination of Passive Testing Properties <i>César Andrés, Mercedes G. Merayo, Manuel Núñez</i> .....	405
---	-----

Formal Verification of UML 2.0 Sequence Diagram <i>Sachoun Park, Taeman Han, Gihwon Kwon</i> .....	411
---	-----

## Formal Methods and Modeling

A Hierarchical Timed Coloured Petri Nets for BPMN-based Process Analysis (S) <i>Ching Huey Wang, Pei Shu Huang, Feng Jian Wang</i> .....	417
---	-----

Temporal Filter A Temporal Extension to Wireshark Display Filter (S) <i>Shaochun Wang</i> .....	421
--	-----

System Modeling from Extended Task Descriptions (S) <i>Jose Luis de la Vara, Juan Sánchez</i> .....	425
--	-----

Specification patterns can be formal and still easy <i>Fernando Asteasuain, Víctor Braberman</i> .....	430
---	-----

A Context Conceptual Model for a Distributed Software Development Environment (S) <i>Ana Paula Chaves, Elisa H. M. Huzita, Vaninha Vieira, Igor Steinmacher</i> .....	437
--	-----

## Service-Oriented Architectures and Applications

Service Automation Architecture as adopted by Unified Communication Audit Tool <i>Shadan Saniepour Esfahani, Talal Siddiqui</i> .....	443
Knowledge Based Service Oriented Architecture for M&A (S) <i>Debasis Chanda, Dwijesh Dutta Majumder, Swapan Bhattacharya</i> .....	448
ISE - Integrated Service Engineering: Applying an Architecture for Model to Model Transformations <i>Hao Hu, Gregor Scheithauer, Guido Wirtz</i> .....	452
A Model-based Business Process Diagnosis Method in Service Oriented Architecture (S) <i>Soo Ho Chang, Soo Dong Kim</i> .....	458
Ontology-Based Dependency-Guided Service Composition for User-Centric SOA <i>Wei-Tek Tsai, Peide Zhong, Jay Elston, Yinong Chen, Xiaoying Bai</i> .....	462
Feature Modeling for Service Variability Management in Service-Oriented Architectures <i>Mohammad Abu-Matar, Hassan Gomaa, Minseong Kim, Ahmed Elkhodary</i> .....	468

## Software Test Automation and Experience

TestDrive - A Cost Effective Way to Create and Maintain Test Scripts for Web Applications (S) <i>Sachin Patel, Priya Gupta, Prafullakumar Surve</i> .....	474
Ontology-Based Test Case Generation For Simulating Complex Production Automation Systems (S) <i>Thomas Moser, Gregor Dürr, Stefan Biffel</i> .....	478
PLeTs-Test Automation using Software Product Lines and Model Based Testing (S) <i>Elder de M. Rodrigues, Leonardo D. Viccari, Avelino F. Zorzo</i> .....	483

Experience with Maintenance of a Functional GUI Test Suite using IBM Rational Functional Tester (S) <i>Yuri Shewchuk, Vahid Garousi</i> .....	489
--	-----

### **Agent-Based Systems**

A Multiagent System for Automate Detection and Diagnosis of Active Tuberculosis on Chest Radiograph and CT Thorax (S) <i>Abdel-Halim Hafez Elamy, Behrouz H. Far, Richard Long</i> .....	495
---	-----

Impact Analysis Model for Brasília Area Control Center using Multi-agent System with Reinforcement Learning (S) <i>Antonio Carlos de Arruda Junior, Alessandro Ferreira Leite, Cícero Roberto Ferreira de Almeida, Alba Cristina Magalhaes Alves de Melo, Li Weigang</i> .....	499
---	-----

Mobile Agents for Active Media <i>Ichiro Satoh</i> .....	503
---	-----

An End-user Domain-specific Model to Drive Dynamic User Agents Adaptations <i>Ingrid Nunes, Simone D.J. Barbosa, Carlos J.P. de Lucena</i> .....	509
---	-----

### **Ontologies and Slow Intelligence Methodology**

Towards Generation of Domain Ontology from LMF Standardized Dictionaries <i>Feten Baccar Ben Amar, Bilel Gargouri, Abdelmajid Ben Hamadou</i> .....	515
--	-----

An Ontology-based Configurator for Customized Product Information based upon the Slow Intelligence Systems Approach <i>Emilio Zegarra, Francesco Colace, Massimo De Santo, Shi-Kuo Chang</i> .....	521
---	-----

UFOCoRe: Exploring Fuzzy Relations According to Specific Contexts <i>Mauricio Jacó Cerri, Cristiane A. Yaguinuma, Marcela Xavier Ribeiro, Marilde T. P. Santos</i> .....	529
---	-----

### **Software Vulnerability**

A String Constraint Solver for Detecting Web Application Vulnerability <i>Xiang Fu, Chung-Chih Li</i> .....	535
--	-----

Towards a Structured Model for Software Vulnerabilities (S) <i>Ming Huang, Yisha Lu, Qingkai Zeng</i> .....	543
 <b>Social Networks and Web Mining</b>	
Dynamic and semantic social networks analysis: a new model based on a multidisciplinary approach. (S) <i>Christophe Thovex, Francky Trichet</i> .....	548
Incremental Construction of Topic Hierarchies using Hierarchical Term Clustering <i>Ricardo M. Marcacini, Solange O. Rezende</i> .....	553
 <b>Software Security</b>	
A Framework for Detecting Code Piracy Using Class Structure <i>Patrice Arruda, Pierre Chamoun, Dwight Deugo</i> .....	559
Detection of Malicious Software Engineer Intrusion <i>Michael E. Shin, Nipul Patel, Sneha Deep Sethia</i> .....	565
Developing Precise Misuse Cases with Security Robustness Analysis (S) <i>Mohamed El-attar</i> .....	571
 <b>Software Product Lines and Tools</b>	
Developing configurable extensible code generators for model-driven development approach <i>Souvik Barat, Vinay Kulkarni</i> .....	577
Feature based Structuring and Composing of SDLC Artifacts (S) <i>Nishigandha Hirve, Tukaram Muske, Ulka Shrotri, R. Venkatesh</i> .....	583
Distributed and Adaptive Execution of Condor DAGMan Workflows (S) <i>Selim Kalayci, Gargi Dasgupta, Liana Fong, Onyeka Ezenwoye, S. Masoud Sadjadi</i> .....	587

Towards Automated Synthesis of Executable Eclipse Tutorials <i>Nuyun ZHANG, Gang HUANG, Ying ZHANG, Ning JIANG, Hong MEI</i> .....	591
---	-----

## **Knowledge Engineering Systems**

An Examination of a Rule-Based Expert System to Aid in the Implementation of the CMMI Framework (S) <i>Tessa Adderley, Sheryl Duggins, Frank Tsui</i> .....	599
--	-----

Tools for Ontology Modeling and Visualization (S) <i>Simon Suigen Guo, Christine W. Chan, Robert Harrison</i> .....	604
--	-----

Modeling and Testing a Knowledge Base for Instructing Users to Choose the Classification Task in Relational Data Mining (S) <i>Lidia Martins da Silva, Ana Estela Antunes da Silva</i> .....	608
---	-----

gOntt, a Tool for Scheduling and Executing Ontology Development Projects <i>Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, Oscar Muñoz, Martín Vigo</i> .....	614
--	-----

## **Software Project Management**

A Project Monitoring Cockpit Based On Integrating Data Sources in Open Source Software Development <i>Stefan Biffel, Wikan Danar Sunindyo, Thomas Moser</i> .....	620
--	-----

Software Configuration Management as a Crosscutting Concern: An Example on Software Testing <i>Elisa Yumi Nakagawa, João Vitor Tornisiello Trevisan, José Carlos Maldonado</i> .....	628
---	-----

Decision Support for Staffing of the Next Software Product Release (S) <i>Emadoddin Livani, Guenther Ruhe</i> .....	634
--	-----

Predicting Project Health Prior to Inception (S) <i>Rose Williams, Jim Graham, Katrina Reffett, Myles Wallace</i> .....	640
--	-----

## Software Engineering Tools and Experience Report

OCL Evaluation on AUTOSAR Model (S)  
*Sachoun Park, Taeman Han, Hyoungju Lim, Gihwon Kwon* ..... 645

Intertwining Implementation with the RealSpec Executable Real-Time  
Specification Language (S)  
*Amir A. Khwaja, Joseph E. Urban* ..... 649

Self-Management of Component Executors for Robot Applications (S)  
*Michael E. Shin, Hemanth Thimme Gowda, Taeghyun Kang, Sunghoon Kim,  
Seungwook Jung, Choulsoo Jang, Byoungyoul Song* ..... 653

## Agent-Based Software Engineering and Applications

A Multi-Agent Model for a Business Continuity Information Network  
*Lily Chang, Xudong He* ..... 657

Agent-based Architecture for Service Ontology evolution management (S)  
*Soumaya Slimani, Salah Baïna, Karim Baïna* ..... 664

Intelligent Software Agent Design Issues with Extensions to the Descartes  
Specification Language (S)  
*Vinitha H.Subburaj, Joseph E.Urban* ..... 668

Meta Context for Agent Planning (S)  
*Csaba Egyhazy* ..... 672

## Component-Based Software Engineering

Software Components Search Approaches in the Context of COTS-based  
Development (S)  
*Nacim Yanes, Sihem Ben Sassi, Henda Hajjami Ben Ghezala* ..... 675

Architecture-centric development and evolution processes for component-based  
software  
*Huaxi (Yulin) Zhang, Christelle Urtado, Sylvain Vauttier* ..... 680

Conflict Analysis in Commercial Off-The-Shelf (COTS) Based Development  
*Hamdy Ibrahim, Tom Wanyama, Armin Eberlein, Behrouz H. Far* ..... 686

Achieve Semantic-based Precise Component Selection via an Ontology Model  
Interlinking Application Domain and MVICS  
*Chengpu Li, Xiaodong Liu, Jessie Kennedy* ..... 692

## **Process and Project Management**

A Top-Down Method for Secure SOA-based B2B Processes  
*Mostafa Madiesh, Guido Wirtz* ..... 698

Supporting Software Process Improvement in Very Small Entities through a  
Template-based Guide  
*Miguel Morales Trujillo, Francisco J. Pino, Mario Piattini* ..... 704

Scrum and Plan-driven Process Integration and its Impact on Effort Estimation  
*Nelio Alves, William Carvalho, Edgard Lamounier* ..... 710

A Case Study of Software Process Improvement Implementation  
*Daniela C. C. Peixoto, Vitor A. Bastista, Rodolfo F. Resende, Clarindo Isaías P.  
S. Pádua* ..... 716

Software Process Reuse by Pattern Weaving (S)  
*Ya-sha Wang, Xiao-yang He, Jin-gang Guo, Jia-rui Jiang* ..... 722

## **Pervasive Computing and Ubiquitous Software**

A Cross-Layer Design for Adaptive Multimodal Interfaces in Pervasive Computing  
*Jun Kong, Weiyi Zhang, Juan Li, Arjun G. Roy* ..... 726

Characteristics of Ubiquitous Software Projects: Pertinence, Relevance, and Use  
*Rodrigo Oliveira Spínola, Guilherme Horta Travassos* ..... 732

Software Engineering in the Embedded Software and Mobile Robot Software  
Development: A Systematic Mapping (S)  
*Daniel Feitosa, Katia R. Felizardo, Lucas Bueno R. de Oliveira, Denis Wolf,  
Elisa Y. Nakagawa* ..... 738

## Software Engineering Tools and Applications

A Visual Bug Report Analysis and Search Tool  
*Carlos Eduardo Albuquerque da Cunha, Yguaratã Cerqueira Cavalcanti, Paulo Anselmo M. Silveira Neto, Eduardo Santana de Almeida, Silvio Romero L. Meira* 742

CFM: A File Manager with Multiple Categorization Support (S)  
*Ali Sajedi Badashian, Hamidreza Afzali, Iman Khalkhali, Morteza Ashurzad Delcheh, Mohammad Shoja Shafiei, Mehregan Mahdavi* ..... 748

TSRR: A Software Resource Repository for Trustworthiness Resource Management and Reuse  
*Junfeng Zhao, Bing Xie, Yasha Wang, Yongjun Xu* ..... 752

## Software Dependability and Reliability

A Log-Assisted Approach Enforcing Consistency in the Presence of Exceptions  
*Nikolas Nehmer* ..... 757

An Automatic Failure Mode and Effect Analysis Technique for Processes Defined in the Little-JIL Process Definition Language  
*Danhua Wang, Jingui Pan, George S. Avrunin, Lori A. Clarke, Bin Chen* ..... 765

## Network and System Security

Secure ad-hoc routing protocol  
*Thouraya Bouabana-Tebibel, Rym Nesrine Guibadj, Sara Mehar* ..... 771

A Pattern Methodology for Modeling Network Forensic Investigations in Converged Tactical Environments (S)  
*Juan C. Pelaez* ..... 777

P2PsecT: Peer-to-peer Security Testbed (S)  
*Eduardo Segura, Xiao Su* ..... 783

## Software Development and System Design

Model-Driven Development of Java Enterprise Applications (S) <i>Andre Pflueger, Wolfgang Golubski, Tobias Haubold</i> .....	787
A documentation approach for the self-adaptive system design <i>Wenhui Zhu, David Lorge Parnas</i> .....	791
Designing Aspects with Use Cases: A Case Study <i>Junhua Ding, Christopher R. Westbrook, M. N. H. Tabrizi</i> .....	797
<b>Reviewer's Index</b> .....	804
<b>Author's Index</b> .....	807

**Note: (S) means short paper.**

# **Keynote I: Prodigious Data, Logic, Processing, and Usage**

**Dr. Alfred Z. Spector**

Vice President, Research and Special Initiatives  
Google, USA

## **Abstract:**

While processor speed and storage capacity have grown remarkably, the geometric growth in user communities, online computer usage, and the availability of data is in some ways even more remarkable. This growth has engendered some truly amazing systems, with even greater possibilities in the future. In this talk, under the rubric of Hybrid Intelligence, I'll discuss a few of the great opportunities we have to harness this data availability to build systems of immense potential. However, I'll also discuss the systems challenges that arise. While today's large scale systems are evolutionarily based on the distributed computing technologies envisioned in the 70's and 80's, sheer scaling has led to many unanticipated challenges. In this talk, I'll also describe many fascinating challenges to ever more cost-effective, reliable, and secure systems.

## **About the Speaker:**

Alfred joined Google in November of 2007 and is responsible for the research across Google and also a growing collection of special initiatives typically projects with high strategic value to the company, but somewhat outside the mainstream of current products. They include Google's health, open source, and university initiatives.

Previously, Alfred was Vice President of Strategy and Technology IBM's Software Business, and prior to that, he was Vice President of Services and Software Research across IBM. He was also founder and CEO of Transarc Corporation, a pioneer in distributed transaction processing and wide area file systems, and was an Associate Professor of Computer Science at Carnegie Mellon University, specializing in highly reliable, highly scalable distributed computing.

Alfred received his Ph.D. in Computer Science from Stanford and his A.B. in Applied Mathematics from Harvard. He is a member of the National Academy of Engineering, American Academy of Arts and Sciences, a Fellow of the IEEE and ACM, and the recipient of the 2001 IEEE Computer Society's Tsutomu Kanai Award for work in scalable architectures and distributed systems.

## **Keynote II:**

# **Building A Smarter Planet With University Collaboration: Empowering People Through Information Integration**

**Josephine M. Cheng**  
IBM Fellow and Vice President  
IBM Research - Almaden

### **Abstract:**

We are all now connected -- economically, technically and socially. Our planet is becoming smarter. Infusing intelligence into the way the world literally works -- the systems and processes that enable physical goods to be developed, manufactured, bought and sold, services to be delivered ... everything from people and money to oil, water and electrons to move, and billions of people to work and live. All these become possible via information integration scattering in many different data sources: from the sensors, on the web, in our personal devices, in documents and in databases, or hidden within application programs. Often we need to get information from several of these sources to complete a task. However, this can be a difficult or time consuming endeavor. This talk describes some information-intensive tasks, choosing examples from such areas as healthcare, science, the business world and our personal lives. It will discuss the barriers to getting information together and delivering it to the people that need it, in a form they can understand, review key research on information integration and information interaction, indicate how the combination may enable real progress, and illustrate where research challenges remain.

### **About the Speaker:**

Josephine M. Cheng is an IBM Fellow and Vice President of IBM Research - Almaden in San Jose, California. She oversees more than 400 scientists and engineers doing exploratory and applied research in various hardware, software and service areas, including: nanotechnology, materials science, storage systems, data management, web technologies, workplace practices and user interfaces. Cheng has been at the forefront of relational database technology for over twenty years. She was also principally responsible for developing IBM's database technology for the Web, allowing people to access huge amounts of data via the internet that was previously accessible only through proprietary systems. Cheng received the Asian American Engineer of the Year award in 2003. She was inducted into the United States National Academy of Engineering (NAE) in 2006 for sustained leadership and contributions to relational database technology and its pervasive applications to a wide range of digital operational systems. In 2006 she was named one of the Top 10 Software Leaders in China and received the Professional Achievement Award from UCLA in 2007. Currently, she is a guest professor at Tsinghua University and Shanghai University; advisory board committee to the School of Software and Microelectronics, Peking University and chairs the advisory board committee of the Department of Computing of Hong Kong Polytechnic University. She is also on the Electrical Engineering and Computer Science (EECS) advisory board of the University of California at Berkeley (UCB) , as well as a member of San Jose State University's Engineering Industry Advisory Council. Cheng also serves on the advisory board of the UCB Center for Information Technology Research in the Interest of Society (CITRIS) and is a board member of the Bay Area Science and Innovation Consortium (BASIC). Josephine Cheng has been awarded 28 patents for her inventions. Cheng was educated at the University of California, Los Angeles (B.S., 1975, Mathematics and Computer Science; M.S., 1977, Computer Science).

# **Keynote III:**

## **The multi-core programming challenge**

**Daniel Cooke**  
**Paul Whitfield Horn Professor**  
**Computer Science Department**  
**Texas Tech University**

### **Abstract:**

History has shown that new languages typically penetrate computing practice in the context of a significant change in computing hardware. The multi-core is considered to be one of the most radical hardware changes in history. Apart from its power to dramatically enhance computing performance, the multi-core is likely to impact our every day life. Parallel processing systems have been and are time consuming and expensive to build and require a significantly higher level of expertise to develop. With the advent of multi-core systems, the average programmer will need to develop this higher level capability. There is a widely held belief that to exploit the full potential of the multi-core systems new languages and computing practices must be developed.

To date parallel programming has been the exclusive purview of so-called extreme programmers. They face daunting challenges not faced by the sequential programmer. They must effectively identify, expose, and express parallelisms. These challenges faced by the parallel programmer are due to an additional dimension of pitfalls one must avoid. First there are significant errors (like race, deadlock, and starvation) that occur only in parallel programs. Nancy Leveson pointed out that testing alone does not necessarily discover these errors due to non-determinism. This was tragically demonstrated in the Therac 25 system. Secondly, a programmer parallelizing a sequential program may introduce new logic errors in the program. In other words the parallel code may not satisfy the specification that satisfied by the sequential program. Thirdly, it is very easy for a naïve parallel programmer to produce parallel programs that actually perform worse than the sequential versions. This is not necessarily an exhaustive list of pitfalls, but it does point out that transitioning from sequential to parallel programming can have a significant impact on functional and nonfunctional requirements.

In addition to a transition from sequential to parallel programming, many believe that programmers are also likely to transition from procedural or object-oriented programming to functional programming. Many functional languages avoid the need to expose and express parallelisms, requiring the programmer to simply identify parallelisms. Furthermore, the functional paradigm facilitates formal verification by treating computation as the evaluation of mathematical functions and by avoiding state maintenance reducing side effects. Thus, the emergence of modern functional programming languages will play an increasing role in making formal methods for industrial control technologies more cost effective.

In the near future, the average programmer may have to transition from being a procedural programmer to being a functional programmer (i.e., a language paradigm shift) as well as from being a sequential programmer to being a parallel programmer (i.e., yet an additional paradigm shift). We hypothesize that the transition from procedural sequential to functional parallel programming is not a  $2N$  problem (where  $N$  is the number of paradigm shifts). It is likely to be at least an  $N^2$  problem. It will be of enormous difficulty. In this talk I will identify the issues facing modern programmers and present an overview of a technical solution we have developed.

### **About the Speaker:**

Dr. Daniel Cooke serves as the Paul Whitfield Horn Professor of the Computer Science Department at Texas Tech University and as Director of its Center for Advanced Intelligent Systems. Previously, Dr. Cooke served as the Manager of NASA's Intelligent Systems Program, a national research initiative in computer science aimed at NASA relevant problems. Cooke led the activities to establish the technical content of the program, took it from formulation to implementation, and helped establish the program office, which he headed at NASA Ames Research Center in Mountain View, California.

Since 1990, Daniel Cooke has published more than 95 technical papers in the areas of computer language design and software engineering. He has served as PI or Co-PI on research grants totaling more than \$14 million, edited many journal special issues, published a book on Computer Language Design, edited a book on Computer Aided Software Engineering, and served as chair or vice-chair for 19 international conferences or workshops. He currently serves as the Chair of the NASA Ames Research Institute for Advanced Computer Science Scientific Advisory Council, Software Engineering Area Editor for IEEE Computer, the Formal Methods Area Editor of the International Journal of Software Engineering and Knowledge Engineering, and as an editor of the International Journal of Semantic Computing.

Dr. Cooke has been an American Electronics Association Fellow, a MacIntosh-Murchison Faculty Fellow, and held the MacIntosh-Murchison Chair in Engineering at U.T. El Paso. In 1996 he was the recipient of the University of Texas at El Paso's Distinguished Achievement in Research Award. In 2001, Cooke received the NASA Ames Research Center Information Sciences Award for leadership in establishing a Model Strategic Research Initiative for NASA. In 2002, he received the NASA Exceptional Achievement Medal and the NASA Group Award, for Contributions to the CICT program. In 2006 he was the recipient of the IEEE Computer Society's Technical Achievement Award for work on SequenceL.

Dr. Cooke discovered two new computational laws upon which computing can be based, leading to the language SequenceL. SequenceL has been used to prototype Guidance, Navigation, and Control Systems for the space shuttle and the crew exploration vehicle. A byproduct of the laws is the identification of parallelisms inherent in a problem solution. In June, 2009 Texas Multicore Technologies, Inc was founded to commercialize a SequenceL to multicore compiler. The company is now working with leading software and hardware companies to improve their ability to parallelize their codes for multicore processing.

# **Future Research Directions for Software Engineering and Knowledge Engineering**

**Guenther Ruhe**  
**Industrial Research Chair in Software Engineering**  
**University of Calgary, Canada**

## **About the Speaker:**

Guenther Ruhe holds an Industrial Research Chair in Software Engineering at the University of Calgary. His interdisciplinary research in the areas of product release planning and project management includes mathematical optimization, software measurement, process modeling and simulation as well as empirical methods. He received a doctorate degree in Mathematics with emphasis on Operations Research from Freiberg University, Germany and a doctorate degree in Computer Science from University of Kaiserslautern, Germany. From 1996 until 2001, he was deputy director of the Fraunhofer Institute for Experimental Software Engineering in Germany.

Ruhe has published more than 160 reviewed research papers at journals, workshops and conferences. He is the Associate Editor of the IST journal and IJSEKE, and Editorial Board member for a number of journals. He is a member of the ACM, the IEEE Computer Society and the Informatics Society GI of Germany.

# Absent features or missing values?

Wen Zhang, Ye Yang, Qing Wang

Laboratory for Internet Software Technologies, Institute of Software, Chinese Academy of Sciences, Beijing 100190, P.R.China  
{zhangwen, ye, wq}@itechs.iscas.ac.cn

## Abstract

*To clarify the essence of unobserved values of software effort dataset, we comparatively investigate the effectiveness of regarding the unobserved values as absent features and missing values with the task of predicting software effort. When regarding unobserved values as absent features, max-margin classification is used to classify the effort directly. While regarding unobserved values as missing values, we use different imputation methods, including MINI (mean imputation based  $k$  nearest neighbor hot-deck imputation), CMI (class mean imputation) and MI (mean imputation) to impute missing values firstly and then SVM (support vector machine) is used to classify software efforts. The experiments show that the treatment of regarding unobserved values in software effort dataset as missing values produces more desirable performance measured by accuracy in using historical data for software effort classification than regarding unobserved values as absent features. Moreover, among the mentioned three imputation methods, on CSBSG data set, CMI has better performance than MINI, and on ISBSG data set, MINI has better performance than CMI. We explain the outcome in this paper.*

## 1. Introduction

When software data was used for effort prediction, the basic motivation behind this attempt is that a historical data set of software project effort can be used to develop a predictive model, by either statistical methods such as linear regression [1] and correlation analysis [2] or machine learning methods such as ANN (Artificial Neural Network) [3] and SVM (Support Vector Machine) [4], to predict the effort of future projects. However, one difficulty confronted with researchers to construct the predictive model is that the historical data set contains a large amount of unobserved values (missing data). Due to the small

sizes of most historical databases, the common practice of ignoring projects with missing data will lead to biased and inaccurate predictive model, which may be detrimental to the accuracy of effort prediction [5][6][7][8]. For this reason, the study of handling missing data in software engineering is becoming an active field.

The problem of handling missing data in effort prediction can be formulated as follows. We refer effort prediction, here, not effort estimation, to prediction of a class label of effort for a project. Effort estimation refers to predict a specific value of the effort of a project and is usually implemented by regression methods.

Assuming a data set  $Y_{com}$  containing projects with attributes for effort prediction as  $Y_{com} = (D_1, \dots, D_i, \dots, D_m)^T$ , where  $D_i$  is a historical project and  $D_i = (x_{i1}, \dots, x_{ij}, \dots, x_{in})^T$  is represented by  $n$  attributes  $X_i$  ( $1 \leq i \leq n$ ) of  $Y_{com}$ . Furthermore, we assume that  $h_i$  denotes the class label of effort for project  $D_i$ , i.e., the label of  $D_i$ . For each  $x_{ij}$ , which is the value of attribute  $X_j$  ( $1 \leq j \leq n$ ) on  $D_i$ , it will be observed or missing.

In order to evaluate the performances of missing data handling techniques, we divide the whole historical data set into two sets: one is used for training predictive model (training set) and the other is used for testing the performance of the model (testing set). That is, we can rewrite  $Y_{com}$  as  $Y_{com} = D = (D_{train} | D_{test}) = (D_1, \dots, D_k | D_{k+1}, \dots, D_m)^T$ , where  $k$  is the predefined number of projects in training set and  $m$  is the total number of projects in  $Y_{com}$ . For instance, in the usually adopted 3-fold cross-validation method,  $k$  is predefined as  $2m/3$  and

the remaining  $1m/3$  projects are used for testing the prediction model.

The difference of training project  $D_i (1 \leq i \leq k)$  and testing project  $D_j (k \leq j \leq m)$  lies in that, the class label of effort  $h_i$  is known for  $D_i$  but  $D_j$  is unlabeled when used for testing. By combing missing data handling technique and machine learning, a predictive model denoted by  $M$  is produced using training set. Further, if we define a Boolean function  $F$  as Equation 1, then the performance of  $M$  can be evaluated by accuracy using Equation 2.

$$F(M(D_j), h_j) = \begin{cases} 1, & \text{if } M(D_j) = h_j, \quad k < j \leq m \\ 0, & \text{if } M(D_j) \neq h_j \end{cases} \quad (1)$$

$$accuracy = \frac{1}{m-k} \sum_{k < j \leq m} F(M(D_j), h_j) \quad (2)$$

The remainder of this paper is organized as follows. Section 2 presents related work concerning absent features and imputation methods in effort prediction. Section 3 describes max-margin which is proposed for classifying data with absent features automatically. Section 4 presents existing imputation methods for handling missing values in software data set. Section 5 conducts experiments to explore the unobserved values in software data set belonging to absent features or missing values. Section 6 concludes this paper and indicates our future work.

## 2. Related work

Recently, the problem of handling incomplete data with absent features has been presented by G. Chechik et al [9]. The basic idea of absent features is that missing data comes up not as the effect of unobserved values of features due to measurement noise or corruption, but as the effect of structurally absent features. In traditional formulation of machine learning, we usually assume that all data samples have the same set of features to characterize their profiles. However, this assumption may be not exactly feasible if some data samples have varying set of features due to their inherent properties.

Imputation methods are widely studied in handling missing data in software effort prediction and estimation. Song and Shepperd [8] propose a new imputation method called MINI (Mean Imputation based k-nearest-Neighbor hot-deck Imputation) to impute missing values in historical data set. We will introduce this method in details in Section 4.3.

Similar researches on missing data imputation are conducted in Myrtveit et al [6] and Strike et al [7].

## 3. Max-margin classification for handling absent features

### 3.1 The formulation of classification with absent features

To be consistent with the section of introduction, assuming we have  $m$  historical projects (also called as data samples interchangeably)  $Y_{com} = (D_1, \dots, D_i, \dots, D_m)^T$  and each  $D_i (1 \leq i \leq m)$  is represented by  $n$  features  $X_i (1 \leq i \leq n)$  and the  $n$ th feature  $X_n$  is the class label of software effort of historical project  $D_i$ . In terms of absent features, we regard each historical project  $D_i$  is characterized by the subset of feature set  $F$  and  $F = \{X_1, \dots, X_n\}$ . If a historical project  $D_i$  has all the features in  $F$ , that is, there is no absent features in  $D_i$ , then its feature set  $F_i = F$  and  $D_i \in R^n$ . In most cases of absents features, for a historical project  $D_i$ , its number of features is less than  $n$ , that is, its feature set  $F_i \subset F$ . Thus, each historical project  $D_i$  can be viewed as embedded in the relevant subspace  $R^{|F_i|} \subseteq R^n$ . Since these historical projects share features, the prediction model learned from the historical data should have parameters that are consistent across historical projects, even if these historical projects do not lie in the same feature space. Thus, in classification of feature models, our task is to construct a predictive model  $M$  as described in Equation 3, which can be used to predict  $H_n$ , the class labels of efforts of projects.

$$M : D_k \rightarrow H_n, \quad 1 < k \leq m \quad (3)$$

Here,  $D_k$  is different from the  $D_j$  in Equation 1, where  $D_j \in R^{|F|}$  ( $|F| = n$ ) but  $D_k \in R^{|F_i|}$  ( $F_i \subseteq F$ ).

### 3.2 The solution of Max-margin classification

Max-margin classification is proposed by Vapnik [10] in classic SVM for the task of binary classification. In Chechik et al [9], C-SVM is adapted for Max-margin classification of data samples with absent features.

Their experiments on edge prediction in metabolic pathways and automobile detection in images validate the effectiveness of their method, which is shown in Equation 4, in the problem of structural missingness, i.e. absent features.

$$\min_{w,b,\xi,s} \frac{1}{2} \|w\|^2 + C \sum_i \xi_i, \quad (4)$$

with constraints

$$\frac{1}{s_i} (h_i (w^{(i)} (x_{i1}, \dots, x_{in})^T) + b) \geq 1 - \xi_i, \quad (5)$$

$$i = 1, \dots, m$$

and

$$s_i = \|w^{(i)}\| / \|w\|, \quad i = 1, \dots, m. \quad (6)$$

To be consistent with Section 1, we also assign  $h_i \in \{-1, 1\}$  denoting the class label of effort for project  $D_i$  and  $(x_{i1}, \dots, x_{in})^T$  are corresponding values of features in  $F_i$ . Here,  $w^{(i)}$  is a vector obtained by taking the entries of  $w$  that are relevant for  $D_i$ . For example, assuming  $w$  is made out as  $(0.5, 0.5, 0.5)$  and the vector  $(x_{i1}, \dots, x_{i(n-1)})^T$  is  $(1, \text{null}, 1)$ . “null” denotes a missing value at the second feature of  $(x_{i1}, \dots, x_{in})^T$ . We will derive  $w^{(i)}$  as  $(0.5, 0.5)$  and  $s_i$  is equal to  $\|w^{(i)}\| / \|w\|$ , that is, 0.8165. The detailed solution of max-margin classification can see reference [9].

## 4. Imputation methods for handling missing values

### 4.1 Mean Imputation

This method imputes each missing value with the mean of observed values. Assuming we have a historical project and its value on feature is missing. Then mean imputation will use as the value for all missing values on feature in historical projects and is the number of historical projects whose values on feature are observed. This method is very simple and easily to implement but the variance of imputed values will be underestimated. For example, if a question about personal income is less likely to be answered by those with low incomes, then the imputed value, which is the mean of the reported values, will decrease the variance [8].

### 4.2 Class Mean Imputation

Instead of using all the historical projects for imputation, CMI imputes missing value of a historical project with the mean of observed values which are in the same class as the project. Assuming we have a historical project  $D_i$  and its value on feature  $X_j$  ( $1 \leq j \leq n$ ) is missing. We define a set of historical projects with values on the  $n$ th feature equal to  $h_i$ :  $D(h_i) = \{D_p \mid h_p = h_i, 1 \leq p \neq i \leq m\}$ . Then CMI will impute  $\bar{x}_{ij} = \frac{1}{N_j} \sum_{k=1}^{N_j} x_{kj}$  as the value for missing values on feature  $X_j$  for historical project  $D_i$ , and  $N_j$  is the number of historical projects in  $D(x_{in})$  whose values on feature  $X_j$  are observed.

### 4.3 MINI

MINI method regards that the improvement of variance by CMI is not enough [8] and should be improved further. Actually, there are two main procedures in MINI algorithm. The first procedure is key feature selection using information gain [11]. The basic idea is to rank features according to their containing information in classifying historical projects with the goal of removing the redundant and irrelevant features in the data.

Here, we argue that the “KeyFeaturesExtraction” method proposed in Song and Shepperd [8] (Section 3.1 in their paper) is the same as the naïve method that we rank the features according to their information gain firstly and remove the features with low information gain. According to the formula of Entropy [11], the computation information gain of a given feature only needs the labels of projects and values of the feature, and has nothing to do with values of other features in the data set. The second procedure is to use KNN method to select top similar projects of a project to impute missing value of the project. In Song and Shepperd [8],  $k$  is predefined as 2 since Kadoda et al [12] suggested this to perform consistently better than higher values of  $k$  in effort prediction. That is, MINI imputes a missing value of a project using the values of the corresponding feature from its 2 most similar projects. More details of algorithm of MINI algorithm can see Song and Shepperd [8] (Section 3.2 in their paper).

## 5. Experiments

### 5.1 The data sets

The data sets used in this paper are (International Software Benchmarking Standard Group, online: <http://www.isbsg.org>) and CSBSG (Chinese Software Benchmarking Standard Group) databases. The ISBSG database contains 1238 projects from insurance, government, etc, of 20 different countries. The ISBSG data set has 70 attributes and many attributes have no values in the corresponding places. In order to validate that the proposed techniques can be effectively used in small data set, we extract 173 projects with missing values on their attributes as the sample data set for experiments. After eliminating attributes with too few observed values, this sample data set has 39 attributes (30 nominal attributes and 9 continuous attributes) and a large proportion of missing values, exceeding 50% on some attributes. We categorized the project efforts in ISBSG data set into 3 classes as listed in Table 4.

**Table 4** The number of projects in each class of ISBSG sample data

Class label of effort	Number of projects
Low	83
Medium	65
High	30

**Table 5** The number of projects in each class of CSBSG sample data

Class label of effort	Number of projects
Low	110
2High	43

CSBSG database contains 1103 projects from Chinese software industry. It was created in 2006 with its mission to promote Chinese benchmarking standards of software productivity. Those projects in CSBSG database were collected from 140 organizations and 15 regions across China by Chinese association of software industry [13]. Originally, CSBSG database contains 179 attributes. In the experiments, we assigned 93 projects and 97 attributes (15 nominal attributes and 86 continuous attributes) as the experimental data set because most CSBSG projects have too much incomplete information that we believe they are impossible to be handled by any technique effectively. We divided the assigned CSBSG projects into 2 classes shown in Table 5.

### 5.2 Experimental design

We compare the effectiveness of between regarding unobserved values as absent features and regarding unobserved values as missing values in classifying

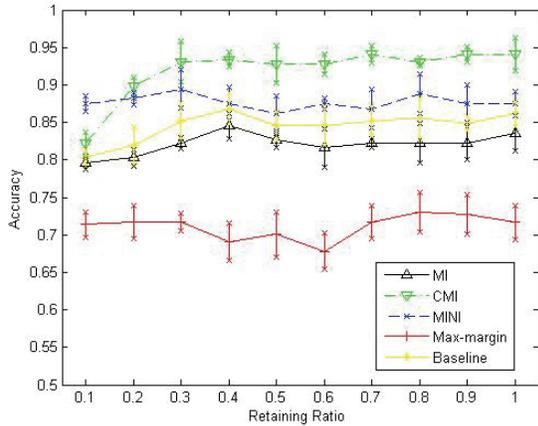
software efforts. When regarding unobserved values as absent features, the Max-margin classifier as Equations 4, 5 and 6 are employed to classify the software efforts. When regarding unobserved values as missing values, we firstly impute the missing values using the imputation methods introduced in Section 4 respectively and then standard C-SVM are used to classify the software effort data. Here, the linear kernel is used in SVM classifier as it has been proven with favorite performance in other applications [14, 15].

The baseline of performance evaluation is conducted on regarding all the missing values in the data sets as zeros (zero imputation). We set a parameter as retaining ratio (between 0.1 and 1 with increments of 0.1) for feature selection in each experiment. To make experimental results comparable, for MINI algorithm, Entropy [11] is used to rank the features in data sets in descending order. For example, if we set retaining ratio as 0.3, then the features with top Entropy values at the percentage of 30 will be used for software effort classification. For absent features, MI, CMI and zero imputation, features are selected from dataset randomly at the number predefined by the retaining ratio. For example, if we set retaining ratio as 0.3, then the percentage of 30 features are assigned randomly for the experiments.

In each experiment, we assign one class of projects in Tables 4 and 5 with positive label (+1) and projects in other classes are assigned with negative label (-1). 3-fold cross validation is used for performance evaluation, that is, each time, we randomly select 2/3 projects with both positive and negative labels for training and the remaining 1/3 projects for testing. Then, classification tasks are conducted on all the 3 classes for ISBSG data set and on all the 2 classes for CSBSG data set, and performance is averaged on these classes with 10 repetitions.

### 5.3 Experimental results

Figure 1 illustrates the performances of standard SVM for software effort using different methods for handling missing values in comparison with max-margin classification, which regarding unobserved values in data set as absent features of software projects. Figure 2 illustrates the comparative results using same methods with those of Figure 1 on ISBSG data set. The horizontal axis is the retaining ratio as defined in Section 5.2. The vertical axis is the accuracy of classifier on corresponding data sets (either imputed or original).

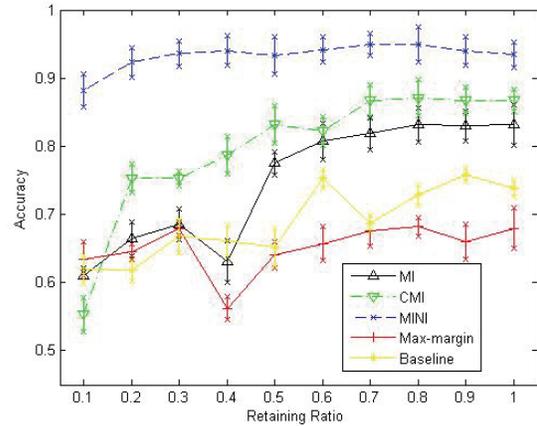


**Figure 1** Performances of standard SVM classification with MI, CMI MINI, and max-margin classification on CSBSG dataset in comparison with baseline performance.

We can see from that Figure 1 that standard SVM on CSBSG data set with CMI produces the best performance among all the methods used for effort classification. In Figure 2, standard SVM on ISBSG dataset with MINI produced the best performance. This outcome illustrates that absent features are not the decisive factors in determining the performance of software effort classification when using SVM classifier compared with missing values. To be further, for the unobserved values in software effort data set, more effort should be invested in imputing missing values of the data rather than regarding them as absent features.

In Figure 1, the standard SVM with imputation methods including CMI, MINI, MI and baseline, outperforms max-margin at all retaining ratios. This outcome illustrates that absent features rarely occur in CSBSG data set. When retaining ratio is small, the imputation methods have very similar performance. However, CMI has the best performance when the retaining ratio is larger than 0.4. We explain this outcome that the more features of the data set which the imputation methods are used on, the better performance the imputation methods would produced.

In Figure 2, most methods have very similar performance except MINI when retaining ratio is small (less than 0.4). We conjecture that there are some absent features with small number in ISBSG data set. When retaining ratio is small, these absent features play an important role in effort classification. However, when retaining ratio becomes larger, their influences on performance of effort classification are less significant in comparison with the influences brought about by missing values in ISBSG data set.



**Figure 2** Performances of standard SVM classification with MI, CMI MINI, and max-margin classification on ISBSG dataset in comparison with baseline performance.

In Figures 1 and 2, we can roughly draw that CMI and MINI outperform other methods in handling unobserved values in software effort data set. In CSBSG data set, CMI has better performance than MINI. While in ISBSG data set, MINI outperforms CMI. We explain that the variance of CSBSG data set is larger than that of ISBSG data set. The basic idea of MINI is to augment variance of imputed values using K-nearest neighbors (K is set as 2 here). In CMI, the variance of imputed values depends on the variance of values in different classes. For CSBSG data set, the better performance of CMI than MINI illustrates that the increased variance by MINI has taken negative effects on effort classification.

## 6. Concluding remarks and future work

In this paper, we investigate the unobserved values of software effort data set. By regarding the unobserved values as missing values and absent features respectively, we compare the performances of classifying software efforts under these two conditions. To be specific, we use MI, CMI and MINI to impute the missing values of CSBSG and ISBSG data set respectively, before the data sets are used to classify software effort automatically by SVM. Max-margin, which is a newly proposed method for handling absent features, is used to classify software effort. We use zero imputation as the baseline for comparison.

The experiment results show that the unobserved values of software data sets (CSBSG and ISBSG) is more prone to be missing values than absent features. This is the main contribution of this paper. In both CSBSG and ISBSG data sets, standard C-SVM on data sets processed by imputation methods can produce better performance on effort classification than max-

margin classification which regards unobserved values as absent features. As for the imputation methods, CMI and MINI have better performances than other methods. Moreover, the variance of data set has a decisive effect on the quality of the data set imputed by imputed methods. When the data set has a small variance, MINI is a desirable imputation method. However, when the data set has a large variance, CMI will produce a better performance.

We conjecture that statistical quality of data sets is a dominant factor deciding the performance of proposed methods. Thus, in the future work, we plan to study the statistical characteristics, such as correlation and distribution of values of different features, of CSBSG and ISBSG data. We also want to explore more software effort data sets to see whether or not there are some common points in statistical quality among them.

## 7. Acknowledgments

This work is partially supported by the National Natural Science Foundation of China under Grant No. 90718042, the National Hi-Tech R&D Plan of China under Grant No. 2007AA010303, and the National Basic Research Program (973 program) under Grant No. 2007CB310802. This work is also partially supported by the Foundation of Young Doctors of Institute of Software, Chinese Academy of Sciences, under Grant No. ISCAS2009-DR03.

## 8. References

[1] N. Ohlsson, H. Alberg. Predicting fault-prone software modules in telephone switches. *IEEE Transactions on Software Engineering*, 22(12), pp.886-894, 1996.

[2] T. Zimmermann, R. Premraj, A. Zeller. Predicting Defects for Eclipse. In *Proceedings of the third Workshop on Predictor Models in Software Engineering (PROMISE'07)*.

[3] Q. P. Hu, M. Xie, and S. H. Ng. Early Software Reliability Prediction with ANN Models. In *Proceedings of the 12th Pacific Rim International Symposium on Dependable Computing (PRDC'06)*.

[4] F. Xing, P. Guo, and M. R. Lyu. A Novel Method for Early Software Quality Prediction Based on

Support Vector Machine. In *Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering (ISSRE'05)*.

[5] A. Mockus. Missing Data in Software Engineering. In F. Shull et al. (eds.), *Guide to Advanced Empirical Software Engineering*, pp.185-200, 2008.

[6] I. Myrtveit, E. Stensrud, and U. H. Olsson. Analyzing Data Sets with Missing Data: An Empirical Evaluation of Imputation Methods and Likelihood-Based Methods. *IEEE Transactions on Software Engineering*, 27(11), pp.999-1013, 2001.

[7] K. Strike, K. E. Emam, and N. Madhavji. Software Cost Estimation with Incomplete Data. *IEEE Transactions on Software Engineering*, 27(10), pp.890-908, 2001.

[8] Q. Song and M. Shepperd. A new imputation method for small software project data sets. *The Journal of Systems and Software*, 80, pp.51-62, 2007.

[9] G. Chechik, G. Heitz, G. Elidan, P. Abbeel, D. Koller.: Max-margin Classification of Data with Absent Features. *Journal of Machine Learning Research*, 9, pp.1-21, 2008.

[10] W. Zhang, T. Yoshida, X. J. Tang: Distribution of multi-words in Chinese and English Documents, *International Journal of Information Technology and Decision Making*, 8(2), pp.1-17, 2009.

[11] J.R. Quinlan. *Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.

[12] G. Kadoda, M. Cartwright, M. J. Shepperd. Issues on the effective use of CBR technology for software project prediction. In: *4th Intl. Conf. on Case Based Reasoning*, Vancouver, 2001.

[13] M. He, M. Li, Q. Wang, Y. Yang, K. Ye. An Investigation of Software Development Productivity in China. In *Proceedings of International Conference on Software Process, LNCS 5007*, pp. 381-394, 2008.

[14] W. Zhang, T. Yoshida, X. J. Tang: Text classification based on multi-word with support vector machine, *Knowledge-based Systems*, 21 (8), 879-886, 2008.

[15] Y. M. Yang, X. Liu, A re-examination of text categorization methods, in: *Proceedings on the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Berkeley, CA, 1999 pp. 42-49.

# Capturing Antagonistic Stakeholder Value Propositions in Value-Based Software Development

Du Zhang

*Department of Computer Science  
California State University  
Sacramento, CA 95819-6021  
zhangd@ecs.csus.edu*

## Abstract

*The primary objective in value-based software engineering is to integrate articulated stakeholder value propositions into the full spectrum of software engineering principles and practices so as to increase the value for software assets. However, a potential pitfall exists that threatens the objective: stakeholder value propositions are based on diverse models and assumptions behind different models often are incompatible, conflicting or inconsistent with each other. In this paper, we define five types of antagonistic stakeholder value propositions based on their logical forms, and propose algorithms to capture them. The take-home message is that having a set of compatible and coherent stakeholder value propositions is of pivotal importance to the success of value-based software engineering and capturing conflicting stakeholder value propositions is a necessary condition before attempting to reconcile them.*

**Keywords:** *value-based software engineering, stakeholder value propositions, inconsistent assumptions.*

## 1. Introduction

The primary objective in value-based software engineering (VBSE) is to integrate articulated stakeholders value propositions (SVPs) into the full spectrum of software engineering principles and practices so as to increase the return on investment. A potential pitfall is that stakeholders possess value propositions that are based on diverse models, and assumptions behind different models often times are incompatible, conflicting or inconsistent with each other. To succeed in VBSE, a critical issue is how to transform or reconcile such diverse and at times conflicting SVPs into a set of compatible and coherent objectives during a software system development life cycle. Toward that end, we need to be able to recognize and identify antagonistic SVPs (or inconsistent assumptions underpinning models used in SVPs) in the first place.

The results as reported in [5] depicted a rather sobering picture of the state-of-the-practice. Key stakeholders such as users, developers, acquirers, and maintainers utilize a variety of models (process, product, property, and success) to articulate their SVPs which can easily lead to inconsistency. Through a subset of antagonistic SVPs in [5], we got a glimpse as to how convoluted the situation can get in an actual software system development life cycle. Identifying and capturing antagonistic SVPs before attempting to reconcile them is a challenging task.

Software system development is inherently hard. As eloquently spelled out in Brooks' classic paper of "No Siler Bullet: Essence and Accident in Software Engineering" [6], the four essential difficulties (complexity, conformity, changeability, and invisibility) still remain valid in developing large software systems. Inconsistent or conflicting SVPs are bound to arise. These inconsistencies, if not properly reconciled, would result in value reduction or destruction for the software asset, thus threatening the main objective of value creation in VBSE. How to leverage or manage inconsistency in value-neutral software development received significant attention and research since the late eighties and early nineties [9, 11-17]. With the augmented business and economic dimensions, VBSE has to double the effort in combating the inconsistency issue [18].

The focus of this paper is on how to identify and capture antagonistic SVPs so as to pave the way for their reconciliation. We define five types of conflicting SVPs based on the logic forms of how model assumptions contradict with each other. These five types of antagonistic SVPs include: complementary, mutually exclusive, incompatible, precedence, and disagreeing inconsistency. We then describe five algorithms that can be utilized to detect those types of conflicting SVPs. The prerequisite for the algorithms is that the set of assumptions for models used by stakeholders to express their SVPs can be represented as formulas of the first order logic and that certain predicates (such as mutually exclusive, synonymous, antonymous, or reified) can be pre-specified and available to the algorithms (see Sections 4 – 7 for details).

The rest of the paper is organized as follows. Section 2 offers an overview on the related work. Definitions, algorithms and examples are given in Sections 3 through 7 on how to capture the following five different types of antagonistic SVPs: complementary, mutually exclusive, incompatible, precedence, and disagreeing inconsistencies in model assumptions. Finally Section 8 concludes the paper with remark on future work.

## 2. Related Work

Managing inconsistency in software system development process has drawn increasing attention in the last twenty years. The issue has been recognized as having direct and significant impact on the quality of the software systems developed.

The work in [2, 3, 8, 9] highlights the importance and complex nature in capturing and reconciling conflicting SVPs. What makes this process challenging is the fact that the specifics of SVPs “tend to be emergent through experience rather than obtainable through surveys” [5]. Different groups of success-critical stakeholders and diverse sets of models they use further complicate the issue.

Though not directly dealing with the issue in the context of VBSE, the study on model clashes in software system development offered some comprehensive results [1]. Majority of the models had been considered in the following categories in [1]: (1) process models (waterfall, evolutionary development, spiral development, agile/rapid application development, and so forth); (2) product models (UML, COBRA, COTS, product line, and so forth); (3) property models (cost, schedule, performance, reliability, security, portability, evolveability, reusability, and so forth); and (4) success models (correctness, stakeholder win-win, business case, and so forth). A detailed inventory of model assumptions was provided that allowed a list of model clash instances to be identified. The classification of model clashes, however, was based on model combinations: process-process, process-product, process-property, process-success, product-product, product-property, product-success, property-property, property-success, and success-success.

Our approach in this paper is orthogonal to the model-centric approach in [1]. We rely on identifying inconsistency through logical instrument, which is at a finer granularity and lends itself to the automation process.

The results in [7] focused on identifying requirements conflicts through quality attributes and traceability. The approach is based on detecting requirements conflicts through analyzing contradictions among the requirements’ quality attributes. If the attributes are conflicting, then so are the respective requirements. Trace dependency is utilized to help establish if attributes are contradicting with each other.

A framework was described in [12] for managing inconsistency in software development. The cornerstone of the proposed framework is a set of consistency rules that describe some proper relationships that must be observed between or among a set of descriptions (analysis models, process models, specifications, design patterns, test plans, and so forth). A violation of a consistency rule by a set of descriptions is referred to as an inconsistency. There are two major components in the framework: inconsistency diagnosis, and inconsistency handling.

The work in [17] reported results in inconsistency management in requirement engineering. The approach attempts to deal with inconsistency at the process (elaboration), product (requirements) and instance (running system) levels. A classification of inconsistencies in requirement engineering has been proposed that includes: process-level deviation, instance-level deviation, terminology clash, designation clash, structure clash, conflict, divergence, competition, and obstruction. Formal techniques and heuristics were proposed to resolve conflicts and divergences.

The results in [15] introduced the concept of overlapped models as the necessary condition for inconsistency. A framework was proposed that consisted of the following activities: detection of overlaps, detection of inconsistencies, diagnosis of inconsistencies, handling of inconsistencies, tracking, and specification and application of an inconsistency management policy. When handling inconsistencies, a course of action is committed only after possible actions are identified, the costs and benefits of each of such actions evaluated, and risks of non-action assessed.

## 3. Complementary Inconsistency

**Definition 1.** Given two SVPs  $V_i$  and  $V_k$ , and two assumptions  $\alpha_i \in V_i$  and  $\alpha_k \in V_k$ , let  $\alpha_i$  and  $\alpha_k$  be denoted by the first-order formulas  $L_i$  and  $L_k$ , respectively. If  $L_i$  and  $L_k$  satisfy the following,

- $L_i$  and  $L_k$  contain the same predicate symbol  $p$ ;
- The predicate symbol  $p$  has the same arity and same terms at corresponding positions in  $L_i$  and  $L_k$ ; and
- $L_i = \neg L_k$  (truth values of the two are opposite);

then we say that there is a *complementary* inconsistency between  $\alpha_i$  and  $\alpha_k$ , which is denoted as  $\alpha_i \neq \alpha_k$ .  $\square$

Let  $\Omega$  be the set of all the formulas representing assumptions for models used by stakeholders to express their SVPs. We have the following algorithm for detecting complementary inconsistency between assumptions.

---

### Algorithm 1. Capturing complementary inconsistency

---

Input:  $\Omega$ ;  
Output:  $\mathfrak{I}_{\neq}$ ; //total number of complementary

```

//conflicting cases
 $\wp_{\neq}$  //set of predicates for which there exist
//complementary literals in  $\Omega$ 
 $\wp_{\neq} = \emptyset$ ;
for  $p \in \Omega$  do {
  if  $(p(t_1, \dots, t_n) \in \Omega \wedge \neg p(t_1, \dots, t_n) \in \Omega)$  then {
     $\wp_{\neq} = \wp_{\neq} \cup \{p\}$ ;
     $\text{Conflict}_{\neq}(p) = \emptyset$ ;
  }
}
for  $p_i \in \wp_{\neq}$  do {
  while  $(p_i(t_1, \dots, t_n) \in \Omega \wedge \neg p_i(t_1, \dots, t_n) \in \Omega)$  do {
     $\text{Conflict}_{\neq}(p_i) = \text{Conflict}_{\neq}(p_i) \cup \{p_i(t_1, \dots, t_n), \neg p_i(t_1, \dots, t_n)\}$ ;
  }
}
 $\mathfrak{I}_{\neq} = \bigcup_{p_i \in \wp_{\neq}} \text{Conflict}_{\neq}(p_i)$ 

return( $\wp_{\neq}$ );
return( $\mathfrak{I}_{\neq}$ ); □

```

**Example 1.** When we have two SVPs that are based on an Agile/Open-Source process model ( $V_{\text{aos}}$ ) and a High Assurance ( $V_{\text{ha}}$ ) property model, their respective assumptions  $\alpha_{\text{aos}}$  and  $\alpha_{\text{ha}}$  contain the following [1]:

- $\alpha_{\text{aos}}$ : software development may proceed without complete requirement specification, which can be denoted as:  $\neg \text{CompleteReqSpec}(x)$ .
- $\alpha_{\text{ha}}$ : a full-verified system must be demonstrated to comply with a complete set of requirement specification, which can be denoted as:  $\text{CompleteReqSpec}(x)$ .

$\alpha_{\text{aos}}$  and  $\alpha_{\text{ha}}$  will be identified by Algorithm 1 as evidence for complementary inconsistency.

#### 4. Mutually Exclusive Inconsistency

**Definition 2.** Given two SVPs  $V_i$  and  $V_k$ , and two assumptions  $\alpha_i \in V_i$  and  $\alpha_k \in V_k$ , if  $\alpha_i$  and  $\alpha_k$  contain literals  $L_i$  and  $L_k$ , respectively, such that:

- $L_i$  and  $L_k$  contain different predicate symbols  $p$  and  $q$ ;
- The predicate symbols  $p$  and  $q$  belong to a concept group in which predicates represent relationships that are mutually exclusive and jointly exhaustive; and
- $L_i = \neg L_k$  (truth values of the two are opposite);

then we say that there is a *mutually exclusive inconsistency* between  $\alpha_i$  and  $\alpha_k$ , which is denoted as  $\alpha_i \neq \alpha_k$ . □

Similarly, we can define an algorithm to identify the mutually exclusive inconsistency between model assumptions.

---

#### Algorithm 2. Capturing mutually exclusive Inconsistency

---

```

Input:  $\Omega$ ;
       $\text{Pred}_{\text{mutex}} = \{\{\dots, p, q, \dots\}_{\kappa} \mid (\kappa \text{ is a group of concepts}) \wedge (p \in \kappa) \wedge (q \in \kappa) \wedge (p \neq q)\}$ 
Output:  $\mathfrak{I}_{\neq}$ ; //total number of mutually exclusive
          //conflicting cases
           $\wp_{\neq}$ ; //mutually exclusive predicate pairs
 $\wp_{\neq} = \emptyset$ ;
for  $\{p, q\} \subseteq \Omega$  do {
  if  $[(p(t_1, \dots, t_n) \in \Omega) \wedge (q(t_1, \dots, t_n) \in \Omega) \wedge (\delta \in \text{Pred}_{\text{mutex}}) \wedge (\{p, q\} \subseteq \delta)]$  then {
     $\wp_{\neq} = \wp_{\neq} \cup \{p, q\}$ ;
     $\text{Conflict}_{\neq}(\{p, q\}) = \emptyset$ ;
  }
}
for  $\{p, q\} \in \wp_{\neq}$  do {
  while  $[(p(t_1, \dots, t_n) \in \Omega) \wedge (q(t_1, \dots, t_n) \in \Omega) \wedge (\{p(t_1, \dots, t_n), q(t_1, \dots, t_n)\} \not\subseteq \text{Conflict}_{\neq}(\{p, q\}))]$  do {
     $\text{Conflict}_{\neq}(\{p, q\}) = \text{Conflict}_{\neq}(\{p, q\}) \cup \{\{p(t_1, \dots, t_n), q(t_1, \dots, t_n)\}\}$ ;
  }
}
 $\mathfrak{I}_{\neq} = \bigcup_{\{p, q\} \in \wp_{\neq}} \text{Conflict}_{\neq}(\{p, q\})$ 

return( $\wp_{\neq}$ );
return( $\mathfrak{I}_{\neq}$ ); □

```

**Example 2.** If we have three SVPs  $V_i$ ,  $V_j$ , and  $V_k$ , and their respective assumptions on the platform for the system to be developed:

- $\alpha_i$ : the platform for the system  $x$  is Windows, which can be denoted as:  $\text{Windows}(x)$ .
- $\alpha_j$ : the platform for the system  $x$  is Unix, which can be denoted as:  $\text{Unix}(x)$ .
- $\alpha_k$ : the platform for the system  $x$  is MacOS, which can be denoted as:  $\text{MacOS}(x)$ .

Windows, Unix and MacOS can be considered as predicates belonging to the mutually-exclusive-and-jointly-exhaustive concept group of platforms. Through Algorithm 2, we can catch those three pairs of mutually exclusive inconsistency regarding the platform choice.

#### 5. Incompatible Inconsistency

Given a literal  $L_j$ , we can define its *synonymous* literal  $L_j$  as one that is syntactically different (having a different predicate symbol), but logically equivalent to  $L_i$ . we use  $L_i \cong L_j$  to denote that. For a literal  $L_i$  and the negation of its synonym  $\neg L_j$ , we use  $L_i \neq \neg L_j$  to describe

their incompatibility. Alternatively, we can define a literal  $L_k$  that is *antonymous* to  $L_i$  as one that is syntactically different (having a different predicate symbol), and logically opposite to  $L_i$ . and use  $L_i \neq L_k$  to denote it. For instance:

$FastResponseTime(x) \neq SlowResponseTime(x)$ ;  
 $FastResponseTime(x) \cong RapidResponseTime(x)$ ;  
 $FastResponseTime(x) \neq \neg RapidResponseTime(x)$ .

**Definition 3.** Given two SVPs  $V_i$  and  $V_k$ , and two assumptions  $\alpha_i \in V_i$  and  $\alpha_k \in V_k$ , if  $\alpha_i$  and  $\alpha_k$  contain literals  $L_i$  and  $L_k$ , respectively, such that:

- $L_i \neq L_k$ ; or
- $L_i \neq \neg L_k$  where  $L_i$  and  $L_k$  are synonymous;

then we say that there is an *incompatible inconsistency* between  $\alpha_i$  and  $\alpha_k$ , which is denoted as  $\alpha_i \neq \alpha_k$ .  $\square$

Algorithm 3 below captures incompatible inconsistency between model assumptions.

---

### Algorithm 3. Capturing incompatible inconsistency

---

Input:  $\Omega$ ;  
 $Pred_{syno} = \{\{\dots, p, q, \dots\} \mid p \cong q\}$   
//sets of synonymous predicates  
 $Pred_{anto} = \{\{\dots, r, s, \dots\} \mid r \neq s\}$   
//sets of antonymous predicates  
Output:  $\mathfrak{I}_{\neq}$ ; //set of incompatible conflicting cases  
 $\wp_{\neq}$ ; //set of incompatible predicate pairs  
 $\wp_{\neq} = \emptyset$ ;  
**for**  $p \in \Omega$  **do** {  
  **if**  $[[[(p(t_1, \dots, t_n) \in \Omega) \wedge (\neg q(t_1, \dots, t_n) \in \Omega)] \vee$   
 $[(\neg p(t_1, \dots, t_n) \in \Omega) \wedge (q(t_1, \dots, t_n) \in \Omega)]] \wedge$   
 $(\delta \in Pred_{syno}) \wedge (\{p, q\} \subseteq \delta)]$  **then** {  
     $\wp_{\neq} = \wp_{\neq} \cup \{p, q\}$ ;  
     $Conflict_{\neq}(\{p, q\}) = \emptyset$ ;  
  }  
  **if**  $[(p(t_1, \dots, t_n) \in \Omega) \wedge (r(t_1, \dots, t_n) \in \Omega) \wedge (\delta \in Pred_{anto})$   
 $\wedge (\{p, r\} \subseteq \delta)]$  **then** {  
     $\wp_{\neq} = \wp_{\neq} \cup \{p, r\}$ ;  
     $Conflict_{\neq}(\{p, r\}) = \emptyset$ ;  
  }  
}  
**for** each  $\{p, q\} \in \wp_{\neq}$  **do** {  
  **while**  $[(p(t_1, \dots, t_n) \in \Omega) \wedge (\neg q(t_1, \dots, t_n) \in \Omega)]$  **do** {  
     $Conflict_{\neq}(\{p, q\}) = Conflict_{\neq}(\{p, q\}) \cup$   
 $\{\{p(t_1, \dots, t_n), \neg q(t_1, \dots, t_n)\}\}$ ;  
  }  
  **while**  $[(\neg p(t_1, \dots, t_n) \in \Omega) \wedge (q(t_1, \dots, t_n) \in \Omega)]$  **do** {  
     $Conflict_{\neq}(\{p, q\}) = Conflict_{\neq}(\{p, q\}) \cup$   
 $\{\{\neg p(t_1, \dots, t_n), q(t_1, \dots, t_n)\}\}$ ;  
  }  
}

**while**  $[(p(t_1, \dots, t_n) \in \Omega) \wedge (q(t_1, \dots, t_n) \in \Omega) \wedge (p \neq q)]$  **do** {  
   $Conflict_{\neq}(\{p, q\}) = Conflict_{\neq}(\{p, q\}) \cup$   
 $\{\{p(t_1, \dots, t_n), q(t_1, \dots, t_n)\}\}$ ;  
}  
}  
 $\mathfrak{I}_{\neq} = \bigcup_{\{p, q\} \in \wp_{\neq}} Conflict_{\neq}(\{p, q\})$   
**return**( $\wp_{\neq}$ );  
**return**( $\mathfrak{I}_{\neq}$ );  $\square$

**Example 3.** There is a set of requirements from stakeholders for a video-on-demand system given as an example in [7]. Two SVPs, one based on the response time performance property model and the other on a mobile device product model, yield two assumptions  $\alpha_{rt}$  and  $\alpha_{md}$  as follows [7]:

- $\alpha_{rt}$ : one second max to start playing a movie, which can be translated into:  $FastResponseTime(x)$ .
- $\alpha_{md}$ : runnable on mobile devices, which can be translated into:  $SlowResponseTime(x)$  (because a mobile device's limited resources would have adverse impact on time performance).

$\alpha_{rt}$  and  $\alpha_{md}$  will be captured as incompatible model assumptions by Algorithm 3.

## 6. Precedence Inconsistency

Let *Precedence* be a predicate denoting the precedence relationship between two artifacts  $x$  and  $y$  in a software system development life cycle.  $Precedence(x, y)$  indicates that  $x$  precedes  $y$ , and is anti-symmetric:

$$\forall x \forall y [Precedence(x, y) \supset \neg Precedence(y, x)].$$

**Definition 4.** Given two SVPs  $V_i$  and  $V_k$ , and two assumptions  $\alpha_i \in V_i$  and  $\alpha_k \in V_k$ , if  $\alpha_i$  and  $\alpha_k$  assert two opposite precedence relationships for artifacts  $x$  and  $y$ :

- $\alpha_i$  asserts  $Precedence(x, y)$ ; and
- $\alpha_k$  asserts  $Precedence(y, x)$ ;

then we say that there is a *precedence inconsistency* between  $\alpha_i$  and  $\alpha_k$ , which is denoted as  $\alpha_i \neq \alpha_k$ .  $\square$

---

### Algorithm 4. Capturing precedence inconsistency

---

Input:  $\Omega$ ;  
Output:  $\mathfrak{I}_{\neq}$ ; //set of precedence conflicting cases  
**while**  $[(Precedence(t_i, t_k) \in \Omega) \wedge (Precedence(t_k, t_i) \in \Omega)$   
 $\wedge (\{Precedence(t_i, t_k), Precedence(t_k, t_i)\} \not\subseteq$   
 $Conflict_{\neq}(\{t_i, t_k\}))]$  **do** {  
   $Conflict_{\neq}(\{t_i, t_k\}) =$   
 $\{Precedence(t_i, t_k), Precedence(t_k, t_i)\}$ ;  
}  
}

$$\mathfrak{S}_{\neq} = \bigcup \text{Conflict}_{\neq} (\{t, t'\})$$

return( $\mathfrak{S}_{\neq}$ );  $\square$

**Example 4.** One of the assumptions in Waterfall process model is that “requirements are completely specified before implementation” ( $\alpha_{wf}$ ) [1]. On the other hand, the success model of IKIWISI (I’ll know it when I see it) [4] has a following assumption: “implementation exists for some requirements” ( $\alpha_{iki}$ ) [1]. If these two assumptions are used by stakeholders in their respective SVPs for developing a software system, we have the following:

- $\alpha_{wf}$ : which can be translated into the following:  
 $\forall x \forall y [Requirement(x) \wedge Implementation(y) \supset$   
 $Precedence(x, y)]$
- $\alpha_{iki}$ : which can be translated into:  
 $\exists x \exists y [Requirement(x) \wedge Implementation(y)$   
 $\wedge Precedence(y, x)]$

After applying normalization processes and anti-symmetric property to  $\alpha_{wf}$  and  $\alpha_{iki}$ , we obtain the following clauses where  $A$ , and  $B$  are ground terms from the domain that denote a specific piece of requirement and a specific implementation, respectively:

$\neg Requirement(x) \vee \neg Implementation(y) \vee \neg Precedence(y, x)$   
 $Requirement(A)$   
 $Implementation(B)$   
 $Precedence(B, A)$

The derivation of an empty clause in the resolution tree (Figure 1) indicates that the two are inconsistent. Of course, Algorithm 4 will capture instances of this precedence inconsistency.

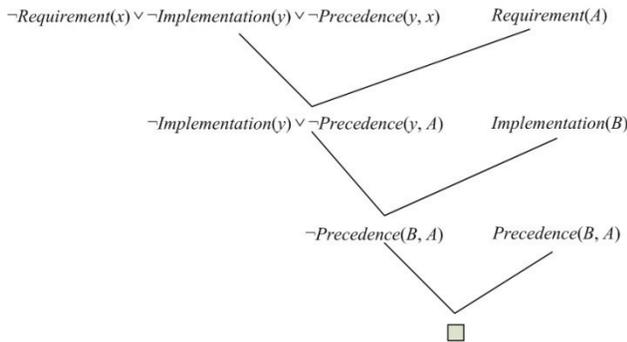


Figure 1. Resolution tree for  $\alpha_{wf}$  and  $\alpha_{iki}$ .

## 7. Disagreeing Inconsistency

An artifact  $x$  is considered as the reification of another artifact  $y$  if both  $x$  and  $y$  represent the same entity or concept or quantity but at different levels of abstraction. We use  $x \cong y$  to represent such a reified

relation. For instance,  $MemoryCapacity(agent1, 4000MB)$  is a reification of  $MemoryCapacity(agent1, 4GB)$ . For two artifacts  $x$  and  $y$  representing the same entity or concept, if there is no longer the reified relation between them, then we say that  $x$  and  $y$  are disagreeing and use  $x \not\cong y$  to denote that effect.

**Definition 5.** Given two SVPs  $V_i$  and  $V_k$ , and two assumptions  $\alpha_i \in V_i$  and  $\alpha_k \in V_k$ , if  $\alpha_i$  and  $\alpha_k$  contain literals  $L_i$  and  $L_k$ , respectively, such that:

- $L_i$  and  $L_k$  contain the same predicate symbol;
- $L_i$  and  $L_k$  refer to the same entity or concept; and
- $L_i$  and  $L_k$  contain disagreeing terms:  
 $(t_i \in L_i) \wedge (t_k \in L_k) \wedge (t_i \not\cong t_k)$

then we say that there is a *disagreeing* inconsistency between  $\alpha_i$  and  $\alpha_k$ , which is denoted as  $\alpha_i \not\cong \alpha_k$ .  $\square$

### Algorithm 5. Capturing disagreeing inconsistency

Input:  $\Omega$ ;  
 $Pred_{rei} = \{p \mid p(t_i, t_j) \wedge p(t_i, t_k) \wedge (t_i \not\cong t_k)\}$   
Output:  $\mathfrak{S}_{\not\cong}$  //set of disagreeing conflicting cases  
 $\wp_{\not\cong}$  //set of disagreeing predicates  
 $\wp_{\not\cong} = \emptyset$ ;  
**for** each  $p \in Pred_{rei}$  **do** {  
  **if**  $(p(t_i, t_j) \in \Omega \wedge p(t_i, t_k) \in \Omega \wedge (t_i \not\cong t_k))$  **then** {  
     $\wp_{\not\cong} = \wp_{\not\cong} \cup \{p\}$ ;  
     $Conflict_{\not\cong}(p) = \emptyset$ ;  
    **while**  $[(p(t_i, t_j) \in \Omega) \wedge (p(t_i, t_k) \in \Omega) \wedge (t_i \not\cong t_k)$   
       $\wedge (\{p(t_i, t_j), p(t_i, t_k)\} \notin Conflict_{\not\cong}(p))]$  **do** {  
       $Conflict_{\not\cong}(p) = Conflict_{\not\cong}(p) \cup \{p(t_i, t_j), p(t_i, t_k)\}$ ;  
      }  
    }  
  }  
}  
 $\mathfrak{S}_{\not\cong} = \bigcup_{p \in \wp_{\not\cong}} Conflict_{\not\cong}(p)$

return( $\wp_{\not\cong}$ );

return( $\mathfrak{S}_{\not\cong}$ );  $\square$

**Example 5.** When specifying software architecture using agile development and product line models in SVPs, the two models entail the following assumptions,  $\alpha_{agi}$  and  $\alpha_{pl}$  [1]:

- $\alpha_{agi}$  assumes that simple design focuses on just current increment product capabilities, which amounts to  $Architecture(x, cpc)$  ( $cpc$  specifies the current capabilities for the system  $x$ );
- $\alpha_{pl}$  assumes that product line architecture is required for successful product line reuse, which corresponds to  $Architecture(x, aa)$  ( $aa$  is the comprehensive application architecture for the system  $x$ ).

Using Algorithm 5, we can identify that  $cpc \not\cong aa$ , hence the disagreeing inconsistency.

## 8. Conclusion

As an improved software development paradigm, VBSE recognizes the importance of business and stakeholders value considerations, and integrates economic, management and cognitive factors with the technical activities. The success of VBSE hinges on having a set of compatible and coherent SVPs for software system development life cycles. However, this objective is threatened by the fact that SVPs are often based on diverse models and assumptions behind different models often are incompatible, conflicting or inconsistent with each other. A necessary condition is to capture conflicting SVPs before attempting to reconcile them through different approaches as discussed in [5].

The contribution of this reported work is that we have defined five types of antagonistic SVPs based on their logical forms, and have proposed algorithms to capture them. Our approach is at a finer granularity compared with the model-centric approach [1] and lends itself to the automation process.

Future work includes the following. While empirical study and evaluation will be conducted with the five types of inconsistent SVPs, additional types of antagonistic SVPs will be pursued. Tool support will be highly desirable to help automate the detection process.

## Acknowledgements

The author would like to thank anonymous reviewers for their valuable comments which help improve the paper's presentation.

## References

1. M. Al-Said, Identifying, Analyzing, and Avoiding Software Model Clashes, Ph.D. Dissertation, University of Southern California, 2003.
2. S. Biffl, A. Aurum, B. Boehm, H. Erdogmus, and P. Grunbacher (ed.), *Value-Based Software Engineering*, Springer, Berlin, 2006.
3. B. Boehm and A. Jain, "An Initial Theory of Value-Based Software Engineering," in *Value-Based Software Engineering*, S. Biffl et al (ed.), Springer, Berlin, 2006.
4. B. Boehm, "Requirements that Handle IKIWISI, COTS, and Rapid Change," *IEEE Computer* Vol. 33, No. 7, 2000, pp.99-102.
5. B. Boehm, "Value-Based Software Engineering: Seven Key Elements and Ethical Considerations," in *Value-Based Software Engineering*, S. Biffl et al (ed.), Springer, Berlin, 2006.
6. F. Brooks, "No Silver Bullet: Essence and Accident in Software Engineering," *IEEE Computer*, Vol. 20, No. 4, 1987, pp.10-19.
7. A. Egyed and P. Grunbacher, "Identifying Requirements Conflicts and Cooperation: How

Quality Attributes and Automated Traceability Can Help," *IEEE Software*, November/December 2004, pp.50-58.

8. H. Erdogmus, J. Favaro, and M. Halling, "Valuation of Software Initiatives under Uncertainty: Concepts, Issues, and Techniques," in *Value-Based Software Engineering*, S. Biffl et al (ed.), Springer, Berlin, 2006.
9. A. Finkelstein, D. Gabbay, A. Hunter, J. Kramer and B. Nuseibeh, "Inconsistency handling in Multi-Perspective Specification," *IEEE Transactions on Software Engineering*, Vol.20, No.8, 1994, pp.569-578.
10. P. Grunbacher, S. Koszegi and S. Biffl, "Stakeholder Value Proposition Elicitation and Reconciliation," in *Value-Based Software Engineering*, S. Biffl et al (ed.), Springer, Berlin, 2006.
11. J. Grundy, J. Hosking, and W. B. Mugridge, "Inconsistency Management for Multiple-View Software Development Environments," *IEEE Transactions on Software Engineering*, Vol. 24, No. 11, November 1998, pp. 960-981.
12. B. Nuseibeh, S. Easterbrook, and A. Russo, "Leveraging Inconsistency in Software Development," *IEEE Computer*, Vol. 33, No. 4, April 2000, pp. 24-29.
13. W. N. Robinson and S. D. Pawlowski, "Managing Requirements Inconsistency with Development Goal Monitors," *IEEE Transactions on Software Engineering*, Vol. 25, No. 6, 1999, pp. 816-835.
14. I. Sommerville, P. Sawyer, and S. Viller, "Managing Process Inconsistency Using Viewpoints," *IEEE Transactions on Software Engineering*, Vol. 25, No. 6, 1999, pp. 784-799.
15. G. Spanoudakis and A. Zisman, "Inconsistency Management in Software Engineering: Survey and Open Research Issues," in *Handbook of Software Engineering and Knowledge Engineering*, S.K. Chang (ed.), World Scientific Publisher, 2001, pp.329-380.
16. C. Toffolon and S. Dakhli, "A Framework for Software Engineering Inconsistencies Analysis and reduction," *Proceedings of IEEE International Conference on Computer Software and Applications*, 1998, pp.19-21.
17. A. van Lamsweerde, R. Darimont, and E. Letier, "Managing Conflicts in Goal-Driven Requirements Engineering," *IEEE Transactions on Software Engineering*, Vol. 24, No. 11, November 1998, pp. 908-926.
18. D. Zhang, "Taming Inconsistency in Value-Based Software development," *Proceedings of 21<sup>st</sup> International Conference on Software Engineering and Knowledge Engineering*, Boston Mass. July 2009, pp.450-455.

# The effects of human-computer interaction modes for weak learners in an animation learning environment

Yu-Fang Yeh

Department of Knowledge Management, Aletheia University, R.O.C.

yufang.yeh@mt.au.edu.tw

## Abstract

*The purpose of this study was to investigate the interaction mode most effective for weak learners, who lack fundamental knowledge and often had difficulty in comprehension of the learning subject, in learning with animation. Participants were twenty-seven undergraduate students enrolled in an Introductory Data Structures course. They were randomly assigned into one of the three treatments: pure-reasoning, predicting-orientation, or reasoning-predicting-combination interaction. They interacted with two animations. They then received a learning experience survey and a learning transfer test immediately after interacting with each animation which demonstrated an example of AVL-tree insertion algorithm and data structure. Our findings indicate that the 'reasoning-predicting-combination' interaction led to the greatest transfer performance and was rated by students as the most useful interaction for understanding the animation content. The findings recommend that for weak learners, interaction for reasoning was effective to develop near-transfer ability at the initial learning phase, whereas when learners' knowledge grew to be capable of near-transfer task, the predicting-orientation interaction became helpful to gain far-transfer knowledge.*

**Keywords:** human-computer interaction, animation, multimedia learning, eLearning, data structures

## 1. Introduction

In recent years, animation has been widely used to present and teach complex materials which are hard to convey by static images and text. However, a large body of studies point out that animation does not guarantee effective learning, it is the interactivity that is the important element of an effective learning environment [1]. Researchers found that interaction such as dialog between human and computer can promote learning

involvement and learning responsibility, as well as trigger off active processing [2]. However, to reach the above-mentioned effects, interaction must be well designed in accordance with learners' learning potential, the ability to learn new information and skills [3]. An interactive activity that is not tailored to learner's learning potential, too simple to learn new things or too hard to respond, often results in "redundant effect" [4], the adding interactive activity either has no effect or detracted from learning. On the contrary, a well-designed interaction will enable learners to explore their learning potential and therefore have augmented learning outcomes. However, the design difficulty is what the design principle for dynamically adaptive to learner's learning potential. In other words, each learner has individual learning potential, and moreover with learning progress, learner's learning potential may grow, therefore how the interaction hence should adjust its assistance strategy in time in order to stratify the renewed learning potential of the learner is an important issue.

Research in instructional animation has suggested that interaction dynamically adjusting to learner's need is superior to fixed interaction [5]. However, how interactive animation should be designed so that it would match specific levels of learning potential remains unclear. The current study intends to answer this question.

## 2. Effective interactions in animation program

Yeh, Chen, Hung and Hwang [6] identify two effective interaction modes for animation learning: reasoning and predicting-orientation interactions. Reasoning interaction requests learners to propose reasons from the perspective of the subject-specific domain knowledge to support the animation demonstration. This interaction helps learners to relate the animation procedures with the underlying rationales and principles. It assists learners to connect conceptual knowledge with procedural knowledge and consequently

develop a comprehensive knowledge structure of the learning content. However, in case that learners are able to spontaneously integrate related elements into a knowledge chunk, the external reasoning activities may become redundant [4], consuming learner's limited mental resources on familiar cognitive skills and thereby interfering learning. At the time when learners already have a general knowledge structure of the learning content, interaction should shift to focus on supporting the learner to verify his/her understanding. Guiding learners to sequentially predict what will be acted on the next screen of the animation is a good method to diagnose the understanding about the animation content and stimulate the learner to amend misconceptions if any.

Based on our previous study [6], it was found weak learners generally benefit from external reasoning activity at the initial phase of knowledge construction. However, with learning progress, the knowledge and learning potential advance of the learner, we doubt whether the learning effect of reasoning activity remains at later learning phases, which aims to develop higher-order cognitive skills such as problem-solving skills. On the other hand, the predicting-orientation interactive activity asks the learner to predict animation operation(s) going to be presented on the next scene. This interaction mode keeps on directing the learner to the next predicting activity if the learner makes a correct prediction. Once the learner makes a wrong prediction, this interaction mode plays the animation demonstration and then provides reasoning activity to help the learner amending knowledge faults. In general, predicting-orientation activity is rather mental loading for weak learners at the initial stage of learning because they usually have no sufficient knowledge to generate meaningful predictions [6]. Our previous study showed that predicting-orientation interaction had poor learning effects at the initial learning phase of weak learners, we wonder if it has better learning effects at the later learning phase for weak learners. We also want to examine if the reasoning-predicting-combination interaction in which learners reason the animation presentation at the beginning learning phase and predict animation presentation at the later learning phase will produce the better learning experience and outcomes as compared to pure-reasoning and predicting-orientation interaction modes. Accordingly, this study developed three interaction modes: pure-reasoning, predicting-orientation, and reasoning-predicting-combination interactions, and investigated which interaction mode was most beneficial for weak learners in learning with animation based on the measures of cognitive transfer performance and affective learning experience.

### **3. Method**

#### **3.1 Participants**

Participants were students enrolled in an Introductory Data Structures course of a university in Taiwan. Because the target students of the current study was weak learners, the teacher of the Introductory Data structures course helped us selecting out students who had difficulty in comprehension of the subject data structures and performed poor in the min-term examination of the course as our potential participants. In addition, those potential participants who had scores below the mean on the prerequisite knowledge test for the learning content, AVL-tree insertion algorithm, were the participants. Consequently, 27 students took part in this study. They volunteered to participate in this study for extra credit of the course.

#### **3.2 The animation learning program**

A web-based animation learning program demonstrates two examples of AL-tree insertion algorithm. For the current research purpose, we designed three versions of the learning program, which differ in their interaction. They were a fixed interaction for reasoning, a dynamic interaction of predicting orientation, and a dynamic interaction of reasoning-predicting-combination. First, in the pure-reasoning interaction version, the program fixedly interacts with learners to complete reasoning sentences to support the actions of the animation through the two examples learning.

Second, in the dynamic interaction of predicting-orientation version, through the two examples the program prompts learners to predict the algorithm going to act on the next screen of the animation, and asks the learners who made a wrong prediction to reason the animation actions. In this version, learners who made a correct prediction directly went to the next predicting activity without watching animation demonstration and doing reasoning activity.

Third, in the dynamic interaction of reasoning-predicting-combination version, the first example interacts with learners to complete reasoning sentences, and the second example provides the predicting-orientation interaction.

#### **3.3 Measures**

##### **3.3.1 Learning transfer tests**

To assess the extent of understanding about the learning content, we administered self-designed knowledge-transfer tests twice, immediately after the first and the second example learning respectively. An isomorphic question as the example demonstrated in the animation was used to assess near-transfer knowledge and a question that required the participants to amend the algorithm taught in the animation example in order to solve the given problem was used to assess far transfer knowledge. Answers were scored zero point for an incorrect answer or one point for a correct answer. Overall, a maximum of three points was achievable for near-transfer test and far-transfer test, respectively. Examples of questions are follows: ‘What is the sum of balance factors of all nodes in the tree after the key adds into the tree?’, ‘What are the critical nodes that need to rotate to restore the property of AVL-tree’, and ‘Draw out the AVL tree after the insertions of following keys.’

### 3.3.2 Learning experience survey

As outlined above, the quality of learning experience provides a modeling of learning affection of the learner. The learning experience was therefore surveyed. This study surveyed learners’ learning experience in terms of the degree of content difficulty, mental effort invested for learning, usefulness of the interaction with the computer. They were measured by using a scale from 1-7, where 1 = extremely low and 7 = extremely high.

### 3.4 Procedures

The study consists of three main steps. First, all participants were given 20 minutes to read a description of 443 word and three figures introducing what is an AVL tree and how the AVL tree rotates to restore its property. Second, all participants took a prerequisite-knowledge test. Third, participants were randomly assigned into one of the three experimental conditions. Based on a pilot study, all participants were given 30 minutes to interact with their respective version of animation program. They were asked to rate their learning experience, and took the learning-transfer test right completion of each example learning. In total, the learning-experience survey and learning-transfer examine were administered twice.

### 3.5 Data analysis

This study used between-subjects comparisons. ANOVAs (analysis of variance) were conducted to

compare effectiveness on cognitive learning outcomes and affective learning experience among experimental conditions. The interaction mode (pure-reasoning, predicting-orientation, reasoning-predicting-combination) was the independent variable. Five dependent measures (difficulty of the learning content, mental effort investment, usefulness of the interaction, near-transfer performance, far-transfer performance) were used to assess how the participants learned about this animation of AVL-tree insertion.

## 4. Results

Analyzing students’ performance in the pre-requisite knowledge for the learning content by an ANOVA revealed no differences across experimental conditions,  $F(2, 24) = < 1$ ).

With regard to the cognitive learning outcomes, the experimental conditions varied strongly in terms of near-transfer performance of the first post-test right after the first example learning,  $F(2, 24) = 6.00$ ,  $MSE = 4.11$ ,  $p < 0.01$ . Bonferroni-adjusted posthoc tests showed that students in the condition ‘pure-reasoning interaction’ as well as in the condition ‘reasoning-predicting-combination interaction’ had much higher near-transfer performance scores than students in the condition ‘predicting-orientation interaction’ ( $p < 0.05$ ).

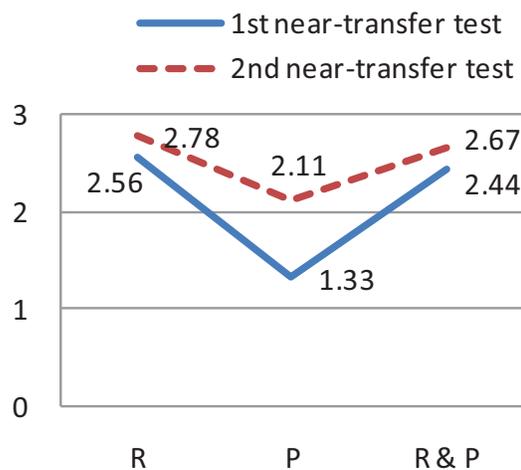


Fig. 1 Near-transfer performance among conditions

Note: R = pure-reasoning-interaction condition;

p = predicting-orientation-interaction condition;

R & P = reasoning-predicting-combination interaction condition

Analyzing students’ performance in the second post-test right after the second example learning, the ANOVA showed that the conditions differed significantly

in students' far-transfer performance.  $F(2, 24) = 5.06$ ,  $MSE = 3.37$ ,  $p < 0.05$ . Bonferroni-adjusted posthoc tests revealed that students in the condition 'reasoning-predicting-combination interaction' had higher far-transfer performance scores than students in the other conditions (all  $p < 0.05$ ).

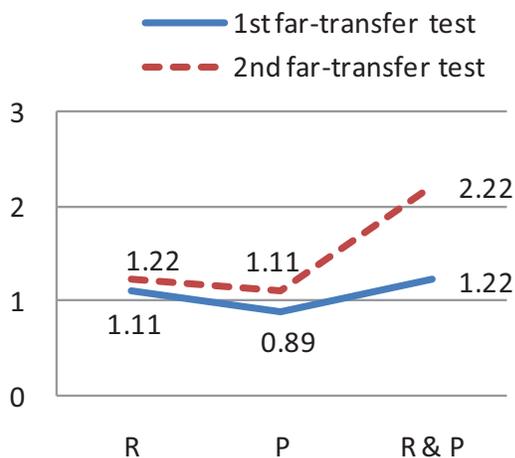


Fig. 2 Far-transfer performance among conditions

Note: R = reasoning-interaction condition;  
 p = predicting-orientation-interaction condition;  
 R & P = reasoning-predicting-combination interaction condition

Subsequently, the three experimental conditions were compared with regard to learning experience survey by a series of ANOVAs for content difficulty, mental effort investment, and usefulness of the interaction. For content difficulty analysis, the main effect was significant for the first example ( $F(2, 24) = 4.99$ ,  $MSE = 3.37$ ,  $p < 0.05$ ) as well as the second example learning ( $F(2, 24) = 6.08$ ,  $MSE = 7.26$ ,  $p < 0.01$ ). For mental effort investment analysis, the main effect was significant for understanding the second example,  $F(2, 24) = 4.46$ ,  $MSE = 6.78$ ,  $p < 0.05$ . For assessing usefulness of the interaction, the main effects of both the first example ( $F(2, 24) = 6.54$ ,  $MSE = 14.11$ ,  $p < 0.01$ ) and the second example ( $F(2, 24) = 13.65$ ,  $MSE = 17.82$ ,  $p < 0.01$ ) learning survey were significant.

Bonferroni-adjusted posthoc comparisons regarding to the learning experience data revealed the following picture: The condition 'reasoning-predicting-combination interaction' was superior to the condition 'predicting-orientation interaction' due to greater perceived usefulness of interaction through both examples learning (all  $p < 0.05$  and  $p < 0.01$ ) and lower perceived content difficulty in the first example learning ( $p < 0.05$ ); The condition 'reasoning-predicting-combination interaction' was

superior to the condition 'pure-reasoning interaction' due to greater perceived usefulness of interaction in the second examples learning ( $p < 0.01$ ); the condition 'pure-reasoning interaction' was superior to the condition 'predicting-orientation interaction' due to greater perceived usefulness of interaction in the first example ( $p < 0.05$ ), as well as less perceived content difficulty and mental-effort demand in the second example learning ( $p < 0.05$ ).

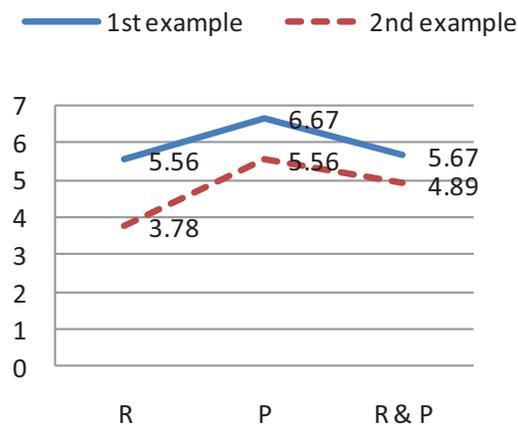


Fig. 3 Perceived content difficulty among conditions

Note: R = reasoning-interaction condition;  
 p = predicting-orientation-interaction condition;  
 R & P = reasoning-predicting-combination interaction condition

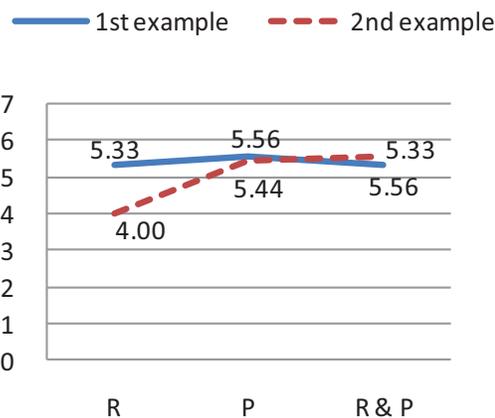


Fig. 4 Mental effort investment among conditions

Note: R = reasoning-interaction condition;  
 p = predicting-orientation-interaction condition;  
 R & P = reasoning-predicting-combination interaction condition

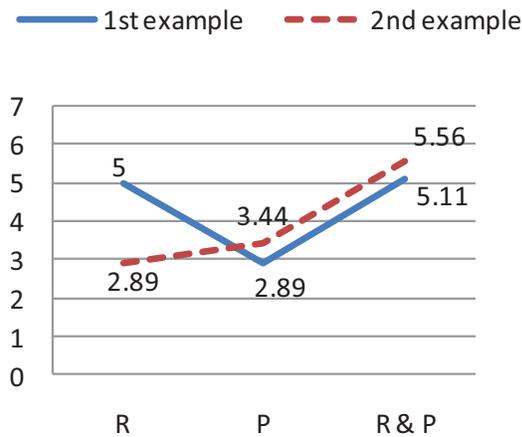


Fig. 5 Usefulness of interaction among conditions

Note: R = reasoning-interaction condition;

p = predicting-orientation-interaction condition;

R & P = reasoning-predicting-combination interaction condition

## 5. Discussion and conclusions

This study proposed three interaction modes and administered an experiment to investigate which interaction mode was most beneficial for weak learners when learning with animation. Our findings reveal that the interaction mode ‘reasoning-predict combination’ led weak learners to the greatest learning-transfer performance and was rated by weak learners the most useful for helping understanding the animation content, though it did not significantly reduce perceived content difficulty and mental effort demand as compared to other interaction modes. These results could be interpreted that that ‘reasoning-predict combination’ interaction dynamically adjusted interaction strategy to fit into the current knowledge level and learning potential of the learner, maintained the interactive activities on a challenging level (not too easy to get tired of the interaction and not too difficult to respond to the interaction) so that it successfully encouraged the learner keeping investing effort on refining their cognitive skills, hence resulting in superior learning outcomes to their counterparts receiving other interaction modes.

Our results also indicate that the reasoning interaction led to good near-transfer performance but had little effects on far-transfer learning. Weak learners receiving reasoning interaction in the first example learning had superior near-transfer performance to their counterparts in the predicting-orientation interaction condition. However, weak learners who continued receiving pure reasoning interaction in the second example learning learned little, no significant improvement on both near- and far-transfer scores.

Additionally, data of the learning experience survey indicate that students in the pure-reasoning interaction condition perceived less content difficulty and spent less mental effort on learning in the second example learning than they did in the first example learning. An explanation for the fading effect of pure-reasoning interaction might be that fixed interactive activities might lose students’ motivation and attention to the learning content so that they ignored the diversity between examples and learned in an easy way (less mental effort investment), consequently little knowledge acquisition.

Furthermore, the current study also found that predicting-orientation interaction is least effective for weak learners through the whole learning phases. Students in the ‘predicting-orientation’ interaction condition performed poorest in both near-and far-transfer tests, and students reported that predicting tasks were very mental demanding, they increased learning difficulty, and they were least useful for assisting understanding. These findings are in line with Yeh *et al.*’s findings that predicting-orientation interaction is not a good teaching strategy for learners with weak knowledge base of the learning subject. The current study further indicates that students who had weak knowledge base could not benefit from predicting-orientation interaction because weak learners usually have not enough knowledge to do meaning prediction, at the most time the predicting task only led to trial-and-error learning, little help for deep understanding. The predict-orientation interaction promoted cognitive skills development only if the learner had fundamental knowledge students about the learning subject. This claimed was supported by the current study finding that students who had good near-transfer scores received predicted-orientation interaction had superior far-transfer performance to their counterparts continuing receiving pure reasoning tasks.

The results of our study are in line with those of past studies that interaction is a crucial consideration in instructional animation program design. Interactions un-tailored to the knowledge level and learning potential of the learner have little effects on cognitive skills development even though the learner continues practicing with the same type of interaction. Meanwhile, the current study included only 27 participants, which might limit the generalizability of the research findings. It is suggested that larger sample-size studies be conducted to provide additional empirical evidence. Moreover, only three interaction approaches were compared in the current study. Future research could be conducted to examine various interaction approaches in effects of cognitive skills development and affective learning experience for different characteristics of learners.

## 6. References

- [1] Mayer, R. E., & Moreno, R. (2002). Animation as an aid to multimedia learning. *Educational Psychology Review, 14*, 87-99.
- [2] Betrancout, M. (2005). The animation and interactivity principles in multimedia learning. In R. E. Mayer (Ed.), *The Cambridge handbook of multimedia learning* (pp. 287-296). Cambridge, UK: Cambridge University Press.
- [3] Fiszdon, J. M., McClough, J. F., Silverstein, S. M., Bell, M. D., Jaramillo, J. R., & Smith, T. E. (2006). Learning potential as a predictor of readiness for psychosocial rehabilitation in schizophrenia. *Psychiatry Research, 143*, 159-166.
- [4] Sweller, J. (2005). The redundancy principle in multimedia learning. In R. E. Mayer (Ed.), *The Cambridge handbook of multimedia learning* (pp. 159-167). Cambridge, UK: Cambridge University Press.
- [5] Moreno, R., & Mayer, R. (2007). Interactive multimodal learning environments. *Educational Psychology Review, 19*, 309-326.
- [6] Yeh, Y.-F., Chen, M.-C., Hung, P.-H., & Hwang, G.-J. Optimal self-explanation prompt design in dynamic multi-representational learning environments. *Computers & Education* (2009), doi: 10.1016/j.compedu.2009.10.013

# Quality Indicators in Requirements Elicitation

Aneesh Krishna<sup>1</sup>, Andreas Gregoriades<sup>2</sup>, Chattrakul Sombattheera<sup>3</sup>

1. Department of Computing, Faculty of Science & Engineering, Curtin University of Technology, WA 6102, Perth, Australia
2. Computer Science & Engineering Department, European University Cyprus
3. Mahasarakham University, Faculty of Informatics, Mahasarakham 44150, Thailand  
[a.krishna@curtin.edu.au](mailto:a.krishna@curtin.edu.au), [a.gregoriades@eu.cy](mailto:a.gregoriades@eu.cy), [chattrakul.s@msu.ac.th](mailto:chattrakul.s@msu.ac.th)

## Abstract

*Domain knowledge is found to be a crucial factor in attaining quality of requirements. System analysts, who have adequate knowledge of software often don't usually understand the organizational context well. This paper, presents a novel approach that utilizes the Tropos, Software System-Business Model (SS-BM) to requirements list elicitation. In the proposed method, requirements items are mapped to SS-BM elements based on their semantic meanings. The mapped requirements list and SS-BM are analyzed in specific sequence according to multiple checking points. Problems with the requirements list are identified in the analysis process and possible improvements are proposed. Quality indicators are used to summarize the quality of requirements at the end of the elicitation process.*

## 1. Introduction

Requirements elicitation is one of the most crucial steps in software development process. Although system analysts have adequate knowledge of software technology, they often don't understand well, the organizational context in which the future system will be situated [2]. System analysts' lack of domain knowledge may result in poor requirements elicitation and even the failure of the software project.

Tropos [2] is a methodology, which is intended to support five phases of software development: early requirements, late requirements, architectural design, detailed design and implementation. In recent years pursuing, Tropos methodology, considerable research efforts have been made on both the early requirements and the late requirements [2, 3]. In early requirements phase, analysts study the existing organization to understand the problem. The output of early requirements is an organizational model [2], which includes relevant actors, their goals and dependencies. In late requirements phase, a FELRE (From Early Requirements to Late Requirements) pattern language[1] is used to transform the organizational model to a new Software System-Business Model (SS-BM), which integrates the Software System Actor (SSA) and expresses the functionality of the software system[1].

We propose a method which uses SS-BM to evaluate and elicit requirements from requirements list. Requirements list is a document provided by the customer to describe business needs at the beginning of a project, which is written in natural language. The SS-BM evolving from the organizational model expresses the functionality of the software system in a semi-formal language. In our method, we make use of SS-BM to evaluate the requirements list, then the requirements list can be optimized to reflect the intentions of the stakeholders more accurately and completely. In the mean time, the risk generated by the system analysts' lack of domain knowledge is decreased considerably. After this optimization process, the prioritized requirements list can be an improved base for the requirements specification. The proposed approach allows systems analyst to detect the correctness, completeness and hierarchical consistency of the requirements list and make improvement in the requirements elicitation process.

This paper is structured as follow: Section 2 presents overview of the proposal. Section 3 introduces the application of Tropos methodology in early requirements. Section 4 explains the late requirements generation process. Section 5 presents our proposed method of requirements elicitation. Section 6 concludes the paper.

## 2. Overview of the Proposal

The proposed method to elicit and improve requirements list is based on the organizational model [2]. The organizational model is the original input model, which is transferred to the SS-BM by the FELRE pattern language [1]. The SS-BM is mapped to the requirements list for elicitation. The mapped SS-BM and requirements list are analyzed in specific sequence according to multiple checking points. The problems of requirements list found during the analysis process will be solved based on the listed guidelines. At the end of the elicitation process, the quality indicators are used to assess the quality of requirements. This considerably improves the overall quality of requirements.

To illustrate the approach, we use the Library Management System as an example. Figure 2 shows the organizational

model of this case study. Figure 1 shows the requirements list provided by the clients. In the following sections, we shall demonstrate how to create the SS-BM from the organizational model (Figure 2) and how to use the SS-BM (Figure 3) to verify the requirements list and elicit further requirements. The motivation behind the entire process is to elicit requirements which are closer to the stakeholders' perceptions.

The library manager has provided the requirements list (Figure 1) describing the required functionality of the software system to be developed.

- |        |  |
|--------|--|
| 1.     | Circulation management                       |
| 1.1.   | Students can borrow books.                   |
| 1.1.1. | Check the inputted student information.      |
| 1.2.   | Students can return books.                   |
| 2.     | Purchase management                          |
| 2.1.   | New book processing                          |
| 2.1.1. | Input book information and scan book barcode |
| 3.     | Generate purchase plan.                      |
| 4.     | E-book management                            |
| 5.     | Provide E-books                              |
| 6.     | Students information maintenance             |

Figure 1. Library Management System Requirements List

The systems analyst has the organizational model from clients. According to the original organizational model, the library needs to realize the circulation management, purchase management and e-book management tasks. Circulation management includes borrowing and returning books. Purchase management includes generating purchase plan, purchasing books and books processing. The library also provides the online e-book. Students' information is stored in the Student Center. Library staff generates the purchase plan based on the book list provided by the bookshop.

Early requirements are describe in natural language as shown in Figure 1. A Library Management System (LMS) requirements list consists of hierarchical items, which are known as *requirements items*. For example, recruitments item 1. *Circulation management* is the parent node of requirements item 1.1. *Students can borrow books* and requirement item 1.2. *Students can return books*. The analyst needs to produce correct and complete requirements specification for a improved software quality.

### 3 Early and Late Requirements

Each software system is developed to fulfill stakeholders' needs/requirements. The Tropos modeling framework has been proved as an effective goal modeling technique to identify and analyze stakeholders and their intentions [2, 3]. Stakeholders in a given domain are represented as actors(s). Stakeholders' intentions are modeled as goals. Actors make use of plans and resources to realize their goals. The key concepts in Tropos for modeling early requirements include the actors, goals, plans, decomposi-

tions, dependencies among actors and resources [2, 3]. In the case study, Tropos organizational models (Figure 2) are used to represent the early requirements. The three actors are *Library*, *Student Center* and the *Bookshop*. The main goal of the actor *Library* is *Library Management*, which has three sub-tasks: *Manage Circulation*, *Manage E-Books* and *Manage Purchase*. The task *Manage Circulation* has two AND Decomposed tasks: task *Borrow Books* and task *Return Books*. Actor *Library* depends on the actor *Student Center* to provide resource *Student Information* and actor *Bookshop* to provide resource *Book List*. Actor *Library* is the depender. Resource *Student Information* is the dependum. Actor *Student Center* is the dependee.

The organizational model is the output of early requirements, which represent the intentions of the stakeholders and organizational context in which future system will be situated [3]. However, there is still a significant gap between the organizational model and functional requirements. To reduce the gap between early and late requirements, we implement the pattern language called "FELRE" [1].

### 4 Implementing the FELRE pattern language

To implement the FELRE pattern language, the following three steps are performed [1].

1. Identify the relevant plans to be automated. In our case study, except plan *Purchase Books* and *Stick Book Barcode* (shown as shaded plan), all the other plans need to be automated. Figure 2 shows the identification condition.

2. Place the SSA into the new organizational model. In this process, the actors that have some plans, goals or dependency relationship (to be automated) are included. In our case study, the actors *Library*, *Student Center* and *Bookshop* were included.

3. Transfer the plans or goals to be automated to the SSA. Table 1 shows a brief description of FELRE pattern language, which consist of five patterns. By applying the transformation rules defined by "FELRE", organizational model shown in Figure 2 was transformed to SS-BM shown in Figure 3. In our case study, the first four patterns were implemented as below:

(a) The Final Plan without dependencies Automation Pattern. In the case study, the plan *Manage E-Books* complied with the characteristics of the *Final Plan without dependencies Automation Pattern* (Figure 2). Figure 3 shows the results of the application of the pattern. The plan was transferred to the SS-BM and a new plan dependency – *Input E-Books*, between the actor *Library* and the SSA was generated.

(b) The General Plan or General Automation Pattern. In our case study, the goal *Library Management* complied with the characteristics of the *General Plan or General Automation Pattern* (Figure 2). The plans, such as *Manage E-Books*, *Manage Circulation* and *Manage Purchase*, are sub-plans of the goal *Library Management*. Figure 3

shows the results of the application of the pattern. Since these sub-plans had been transferred to the SSA, the goal *Library Management* was also transferred to the SSA.

(c) The *Depender-Dependee Actor Plans Automation Pattern*. In our case study, the plan *Check Student Information* complied with the characteristics of the *Depender-Dependee Actor Plans Automation Pattern* (Figure 2). The plan *Check Student Information* acts as the depender. The resource *Student Information* is the dependum. The plan *Send Information (Student Center)* is the dependee. Figure 3 shows the results of the application of the pattern. The plan decomposition is created. The plan *Check Student Information* is the parent node. The plan *Get Student Information* is the child node, which depends on the actor *Student Center* to provide *Student Information*.

(d) The *Depender Actor Plan Automation Pattern*. In our case study, the plan *Generate Purchase Plan* complied with the characteristics of the *Depender Actor Plan Automation Pattern* (Figure 2). The *Generate Purchase Plan* is the depender, which was automated. Figure 3 shows the results of the application of the pattern. Actor *Library* still depends on the actor *Bookshop* to provide resource *Book List*. Plan *Generate Purchase Plan* depends on the actor *Library* to enter book information (task *Enter BookInfo*).

## 5 Mapping requirements items with SS-BM elements

Analysts map each requirements item into SS-BM elements according to their semantic meanings. Figure 3 illustrates the semantic mappings from the requirements items to SS-BM elements. The instance, SS-BM used to specify semantic mappings from the requirements item to SS-BM elements is called *map item*. For example, the requirements item *1.1. Students can borrow books* was mapped to SS-BM element *Borrow Books* by #2 *map item*. In the SS-BM, the element mapped to requirements items is called *mapped SS-BM element*. The SS-BM element *Borrow Books* is a mapped SS-BM element. In the requirements list, the requirements items mapped to SS-BM elements is called *mapped requirements item*. Hence, the requirements item *1.1. Students can borrow books* is a mapped requirements item.

In the mapping process, one requirements item might be mapped to many SS-BM elements by one map item. For example, the SS-BM elements *Check Student Information* and *Get Student Information* were mapped to requirements item *1.1.1. Check the inputted student information* by #3 *map item*. On the other hand, more than one requirements items might be mapped onto one SS-BM element.

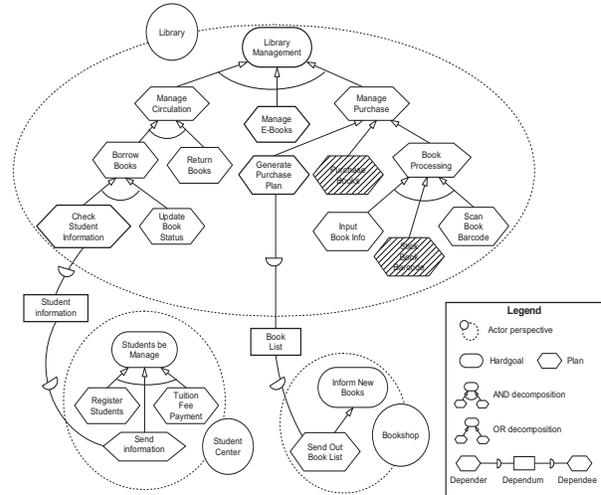


Figure 2: Library Management Organizational Model

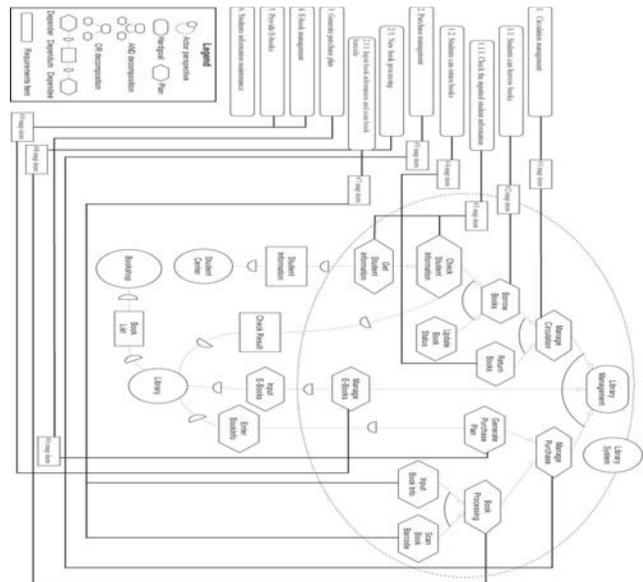


Figure 3: Library SS-BM mapped with Requirements List

### 5.1 Requirements elicitation and verification

Requirements are improved by analyzing the mapping between requirements list items and SS-BM elements according to different checking points. For each selected requirements list item, *Correctness Checking* is conducted that checks whether the requirements item is mapped to an SS-BM element. For every SS-BM elements, the *Completeness Checking*, *Parent/Child Hierarchical Consistency Checking* and *Peer/Peer Hierarchical Consistency Checking* is conducted. The *Completeness Checking* aims to check whether the SS-BM element was mapped to requirements items or not. The *Parent/Child Hierarchical Consistency Checking* aims to check whether the parent/child hierarchical relationship of the SS-BM element

and its mapped requirements item is consistent. The *Peer/Peer Hierarchical Consistency Checking* aims to check whether the peer/peer hierarchical relationship of the SS-BM element and its mapped requirements item is consistent.

Figure 4 shows the new library management system requirements list, after performing the verification and elicitation processes.

- |  |
|--|
| <ol style="list-style-type: none"> <li>1. Circulation management             <ol style="list-style-type: none"> <li>1.1. Students can borrow books.                 <ol style="list-style-type: none"> <li>1.1.1. Check whether it's a legal student                     <ol style="list-style-type: none"> <li>1.1.1.1. Get student information from Student Center</li> </ol> </li> <li>1.1.2. Update book status</li> </ol> </li> <li>1.2. Students can return books.</li> </ol> </li> <li>2. Purchase management             <ol style="list-style-type: none"> <li>2.1. New book processing                 <ol style="list-style-type: none"> <li>2.1.1. Input book information</li> <li>2.1.2. Scan book barcode</li> </ol> </li> <li>2.2. Generate purchase plan.</li> </ol> </li> <li>3. E-book management</li> </ol> |
|--|

Figure 4. New Library Management System Requirements List

## 5.2 Quality Indicator

After the elicitation process, the problems in the requirements list can be discovered. We created three measurable *quality indicators* to summarize the quality of the requirements list in numerical order. The three quality indicators are *Correctness*, *Completeness* and *Hierarchical Consistency* indicators.

- **Correctness (CO):** The proportion of requirement items that were mapped into SS-BM elements represents the correctness of the requirements list. The *Correctness Checking* result can be reflected by this indicator.

$$CO = \frac{\# \{\text{requirements items that are mapped into the SS-BM elements}\}}{\# \{\text{requirements items}\}}$$

Based on the case study, there are eleven requirements items of which, the ten are mapped to the SS-BM elements. Hence,  $CO = 10/11 = 91\%$

- **Completeness (CP):** The proportion of SS-BM elements that are mapped to the requirements items represent the completeness of the requirements list. The *Completeness Checking* result can be reflected by this indicator.

$$CP = \frac{\# \{\text{SS-BM elements that are mapped onto the requirements items}\}}{\# \{\text{SS-BM elements}\}}$$

In the case study, there are twelve SS-BM elements. The *Update Book Status* is not mapped to any requirements items and the *Manage E-Books* is mapped to two requirements items. Hence, the  $CP = 10/12 = 83\%$

- **Hierarchical Consistency (HCST):** Two types of hierarchical relationships are used to describe the relationships between two SS-BM elements or between two requirements items. They are parent-child relationship and peer-peer relationship. The proportion of hierarchical relation-

ships in SS-BM that are not conflicting with the hierarchical relationships in requirements list represent the hierarchical consistency of the requirements list. The *Parent/Child Hierarchical Consistency Checking* and *Peer/Peer Hierarchical Consistency Checking* result can be reflected by this indicator.

$$HCST = \frac{\# \{\text{hierarchical relationships in SS-BM not conflicting with the hierarchical relationships in requirements list}\}}{\# \{\text{hierarchical relationships in SS-BM}\}}$$

In our case study, there are thirteen hierarchical relationships in SS-BM. For example, the relationship between SS-BM element *Borrow Books* and element *Check Student Information* is parent-child. Elements *Borrow Books* and *Update Book Status* also have parent-child relationship. The relationship between element *Check Student Information* and element *Update Book Status* are the peer-peer relationship. There are totally three relationships between these three SS-BM elements. The SS-BM elements *Borrow Books* and *Check Student Information* were mapped to requirements item *1.1. Student can borrow books* and *1.1.1. Check the inputted student information* respectively. The SS-BM element *Update Book Status* was not mapped to any requirements item. In other words, there is only one relationship between the requirements items that the three SS-BM elements were mapped to. The relationship between requirements items *1.1. Student can borrow books* and *1.1.1. Check the inputted student information* is a parent-child relationship, which complied with the relationship of the related SS-BM elements in the SS-BM. In these three SS-BM elements, the HCST equals 33% (1/3).

In the whole case study:  $HCST = 7/13 = 54\%$

## 6 Conclusion

In this paper, a method to elicit requirements from the SS-BM is proposed. The SS-BM is derived from organizational model by implementing FELRE pattern language. In the end, the quality indicators are used to estimate the quality of the requirements elicited. We are working towards automating certain steps in the proposed approach.

## References

1. A. Martínez, O. Pastor, J. Mylopoulos, P. Giorgini: From Early to Late Requirements: A Goal-Based Approach. *AOIS 2006*: pp. 123-142
2. P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, J. Mylopoulos: Tropos: An Agent-Oriented Software Development Methodology. *Autonomous Agents and Multi-Agent Systems 8*(3): pp. 203-236 (2004)
3. A. Fuxman, J. Mylopoulos, M. Pistore, P. Traverso: Model Checking Early Requirements Specifications in Tropos. *RE 2001*: pp. 174-181

# A UML Profile Oriented to the Requirements Collecting and Analyzing for the Multi-Agent Systems Project

Gilleanes Thorwald Araujo Guedes, Rosa Maria Vicari  
*Instituto de Informática*  
*Programa de Pós-Graduação em Computação (PPGC)*  
*Universidade Federal do Rio Grande do Sul (UFRGS) - Porto Alegre - RS – Brasil*  
*gtguedes@inf.ufrgs.br, rosa@inf.ufrgs.br*

## Abstract

*This paper describes a proposal for the creation of a UML profile oriented to the requirements collecting and analyzing for the multi-agent systems project. By means of this profile we intend to model reactive and cognitive agents, as well as actions, perceptions, goals and plans for those agents in a collecting and analysing requirements level. At a first moment this profile was created to be applied to the intelligent tutoring systems project, though we believe it might be applied to multi-agent systems projects oriented to other dominions as well. In this paper we shall describe the proposed profile as well as its application on the modeling of the MCOE intelligent tutoring system.*

## 1. Introduction

In the area of Artificial Intelligence, the employment of intelligent agents as auxiliary aids to software applied to the most diverse dominions is being spread out. This practice has shown to be a good alternative for the development of complex systems, fostering a great increase of agent-supported software development in the several areas, one of those the intelligent tutoring systems, in which agents are also employed as pedagogical agents.

However, the development of this kind of system has presented new challenges to the software engineering area and this led to the surfacing of a new sub-area, blending together concepts brought over from both the software engineering and artificial intelligence areas, which is known as the AOSE - Agent-Oriented Software Engineering, whose goal is that of proposing methods and languages for projecting and modeling agent-supported software.

Despite many AOSE methods having been created for multi-agent systems projects, [1] states that these are not totally adequate for the modeling of intelligent tutoring systems. Likewise, some attempts to adapt and extend UML (Unified Modeling Language) to the multi-agent

systems projects were made; nonetheless, those attempts we have studied did not concern themselves with the extending and applying of some UML resources, such as the use-case diagram, which is mainly employed for requirements collecting and analyzing, which is an essential phase for the achievement of a good system project. Thus, we are developing a UML profile for the specific project of intelligent tutoring systems, though we believe it may be applied upon any multi-agent system project, to further which end we started by adapting the use-case diagram.

## 2. UML, Metamodels and Profiles

According to [2], the UML is a visual language for specifying, constructing, and documenting the artifacts of systems. It is a general-purpose modeling language that can be applied to all application domains.

The UML specification is defined by using a meta-modeling approach that adapts formal specification techniques. When meta-modeling, we initially establish a distinction between meta-models and models. A model typically contains model elements. These are created by instantiating model elements from a meta-model. The typical role of a meta-model is that of defining the semantics for the way model elements within a model get instantiated.

A Profile is a kind of Package that extends a reference meta-model. The primary extension construct is the Stereotype, which is defined as part of the Profiles. A profile introduces several constraints, or restrictions, on ordinary meta-modeling through the use of the metaclasses defined in this package. A profile allows for the creation of "dialects" from UML in order to adequate it to certain dominions and/or technologies.

## 3. UML-Derived Languages

Some attempts have already been tried to adapt UML for the project of multi-agent systems, though nothing specific for the project of intelligent tutoring systems. One of the first attempts was the AUML language [3].

Besides that, other languages, like AML [4], AORML [5] and MAS-ML [6] were also proposed.

However, neither of the above focus the matter of requirements collecting and analyzing nor on its modeling by means of the UML use cases diagram; and no attempt was found in those languages to extend the metaclasses employed in that diagram for applying them on the multi-agent systems project.

Among the existing AOSE methodologies Tropos [7] is one of the best oriented to the requirements collecting and analyzing, but it uses a notation completely different from that of UML. The advantage in adapting and applying UML for the modeling of multi-agent systems is the wide acceptance of this language and that most of professionals in the software engineering field can understand it, something that makes easier its application in most dominions. Although the work of [8] proposes six UML views partially-based on Tropos, none of those represents requirements or applies the use case diagram.

#### 4. The Proposed UML Profile

Considering that UML is a standard modeling language broadly accepted and understood in the software engineering area, that multi-agent systems own their proper characteristics, and that very few of works applying UML into a multi-agent systems project did care to focus the matter of requirements collecting and analyzing, we decided on creating a UML profile in which we extend the metaclasses employed by the use-case diagram, thus creating stereotypes prepared to identify the particular functionalities of this kind of system, as can be seen in the figure 1. For reasons of space we opted to divide the profile in two Figures.

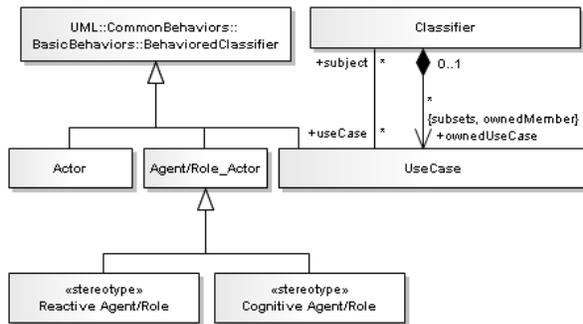


Figure 1. The Profile's First Part.

To develop this section of the profile, we began by using the original metaclass, Actor. According to [9], the Actor metaclass represents a type of role played by an entity that interacts with the subject, but which is external to the subject. Actors may represent roles played by human users, external hardware, or other subjects. An actor does not necessarily represent a specific physical entity but merely a particular facet (i.e., "role") of some entity that is relevant to the specification of its associated use cases. Thus, a single physical instance may play the role of several different actors and, conversely, a given actor may be played by multiple different instances.

However, most of the times, the software agents are not external to the software, rather they customarily are inserted in the system environment and, as they are independent, proactive, and able to interact with the software according to their goals, these should be represented as actors. So, as states [10], it is necessary to adapt this concept, considering that agents can be internal to the system, that is, an agent can be a part of the subject and, therefore, if we are to represent agents (their roles) as actors, said actors should be internally represented within the system's borders, for they belong to the software.

The representation of agents/roles as UML actors can also be seen in [11], in which agents represent active objects and are modeled within the system as rectangle-headed actors. However [10] only suggests that the concept of actor should be adapted and [11] did not created a UML profile for agents. In our work we explicitly created and derived new metaclasses from the original metaclasses endeavoring for the former to allow an adequate requirements modeling in MAS.

However, according to [2], it is not possible to take away any of the constraints that were applied to a metamodel such as UML when using a profile. Thus, instead of extending the Actor metaclass, we created a new metaclass derived from the same metaclass as the Actor metaclass had been, as can be seen in figure 1, creating the Agent/Role\_Actor Metaclass. In this new metaclass we copied all the characteristics of the Actor metaclass, including the same symbol, suppressing only the constraint that an actor needs must be external to the system. And, from this new metaclass we derived the Reactive Agent/Role and Cognitive Agent/Role metaclasses.

As can be observed, we applied the stereotype "stereotype" in both metaclasses, which means that these metaclasses will be applied as stereotypes on agent/role actors while attributing them special characteristics, for said agent/role actors shall be utilized within specific domains to represent cognitive and reactive agents/roles. We opted for creating two stereotypes so as to establish a difference between reactive and cognitive agents, considering that either group presents different characteristics.

According to [9], a use case is the specification of a set of actions performed by a system, which yields an observable result that typically affords some value for one or more actors. Each use case specifies some behavior that the subject can perform in collaboration with one or more actors. Use cases can be used both for specification of the (external) requirements on a subject and for the specification of the functionality offered by a subject. Moreover, the use cases also state the requirements the specified subject poses on its environment by defining how they should interact with the subject so that it will be able to perform its services.

Thus, with the objective of adapting the use cases concept to the MAS modeling, in the same fashion as we proceeded with the metaclass, Agent/Role\_Actor, we derived a new metaclass departing from the same metaclass as the UseCase metaclass, called "InternalUseCase" and then we created the same

relationships with the Classifier metaclass for this new metaclass as those belonging to the UseCase metaclass, as can be seen in the Figure 2, that represents the second part of our profile. We made it that way because we intended to represent goals, plans, actions and perceptions as use cases, but the semantics of the UseCase Metaclass says that use cases represent external requirements, therefore, to adapt this concept to the multi agent systems modeling we created a similar metaclass, all the while we modified its semantics to represent internal requirements.

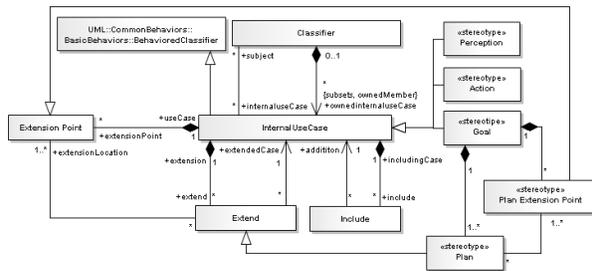


Figure 2. The Profile's Second Part

Naturally, all Internal use cases are internal to the system and can be neither used nor seen by human users. From the InternalUseCase metaclass, we extended some metaclasses to attribute special characteristics to internal use cases; these extended metaclasses will be employed as stereotypes. Thusly, we created the metaclasses, Perception and Action, to model internal use cases that contained the necessary steps for an agent to perceive or do something, a procedure also suggested by [10] to be applied in normal use cases, though as stated before in [9], these refer to the external requirements, not to the internal ones.

A third metaclass was derived from the metaclass, InternalUseCase, to represent Goals. A internal use case employing the stereotype, Goal, shall contain a description of a desire to be attained by an agent and the possible conditions for that desire to become an intention. A somewhat similar proposal for representing goals as use cases can be seen in [11].

Besides using internal use cases, we went a little further beyond the proposal presented by [11], when we considered that, just like a goal represents a desire that will not necessarily become an intention, the steps for its execution should be detailed in other or others internal use cases; in the situation of a internal use case employing the stereotype, Goal, we shall only detail those perceptions and conditions necessary for that goal to become an intention. Considering a goal might eventually have more than a plan and that this or these plans will only be accomplished within certain conditions, we decided on deriving still another metaclass, Plan, from the metaclass, Extend.

According to [9] an Extend represents a relationship from an extending use case to an extended use case that specifies how and when the behavior defined in the extending use case can be inserted into the behavior defined in the extended use case. If the condition of the extension is true at the time the first extension point is

reached during the execution of the extended use case, then all of the appropriate behavior fragments of the extending use case will also be executed. If the condition is false, the extension will not occur.

Because a plan only will be triggered after some particular condition be satisfied, we decided on extending the metaclass, Plan, from the metaclass, Extend, and associated the former to the metaclass, Goal, by means of a composite association, whose multiplicity informs that a Goal may have at least an associated Plan, although it can have many such associations; however, a single Plan can only be associated to a single Goal.

Finally, we derived the metaclass, Plan Extension Point from the metaclass, Extension Point. According to [9] an extension point identifies a point in the behavior of a use case where that behavior can be extended by the behavior of some other (extending) use case, as specified by an extend relationship. We derived this last metaclass only to set up a difference between Plan Extension Points and normal Extension Points; nonetheless, this also can serve to render explicit the condition for a plan to be executed.

An example on how to apply this profile will be presented in the next session in which the MCOE System will be modeled.

## 5. The MCOE (Multi-agent CO-operative Environment) System

The MCOE [12] architecture is composed by a hybrid society of agents that work together to achieve a common goal: to fight against the pollution and to maintain the system equilibrium of a lake. Here we have reactive agents (microorganisms, plants and fish) and cognitive agents (the Tutor agent, and students agents represented within the system by the characters, Mother Nature, Mayor, Fisherman and Tourist).

The tutor receives information about each student model. Such information is composed by the mental states of each student, the students' actions, and the environment sensors. The sensors pick up information about energy levels checked by the control. The Tutor uses such information to select a specific strategy with the associated pedagogic tactic. The students' mental states and their actions are the perceptions that the Tutor receives from the environment.

The tutor Agent has only one intention, that is, to aid the student into maintaining an adequate energy level in the environment. It believes that it may aid by sending messages to the students. The contents of these messages will depend on the strategy adopted by the tutor at the moment.

When there is only one desire, this desire is the only candidate for an intention. It will be adopted as an intention if the tutor believes that there is a sequence of actions that the students can execute. At this moment, the tutor depends on its knowledge on the student's mental states in order to decide what to do. Since it does not have (yet) this knowledge, it then waits. The tutor inserts in its beliefs, which triggers and makes itself to reconsider its options when interaction occurs.

Once the Tutor has achieved that intention, it uses its beliefs about the environment on how to fight against pollution in said environment, on how to advise the students in order to help them to control the environment, and so on.

The MCOE system already exists for some years, we are modeling systems developed before by the group, after finding limitations into modeling them with the available resources in that time, thus we are applying the profile developed in prior systems as a basis to our study.

The students were modeled as normal actors, like human users playing the game and taking up role characters, but they remain external to the system. Each character can perform different functions, meant to increase or to decrease the energy level within the lake. When a student takes up the character of Mother Nature, he/she can speed up biodegradation, add water plants, change the lake's bottom, or shine extra sunlight on it; in the event s[he] takes up the Mayor role, s[he] can add trash bins, forbid unwanted activities, apply fines, or withdraw fishing permits.

Due to the variety of functions, we decided to represent the student as a normal actor and attribute to him the "Choose Character" functionality (a normal use-case), by which the student can pick the character he will impersonate. Considering that all other functionalities depend on the chosen character, we derived sub-actors from the student actor to represent each of the characters and to attribute to each sub-actor normal use cases representing the functionalities each character can require from the system, as can be seen in Figure 3, showing an excerpt of the use case diagram that focus on this modelling.

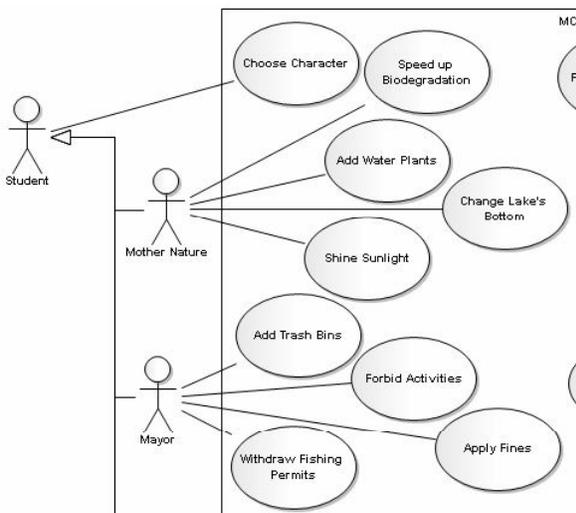


Figure 3. Normal Actors and Use Cases Modeling the Student, the Characters in the Game and the Available Functionalities.

The reactive agents (fish, plants, and micro-organisms) were modelled like agent/role actors employing the stereotype, "Reactive Agent/Role", and these actors were inserted inside the system, instead of the normal actors. The reactive agents are designed to execute determinate actions (represented by internal use cases employing the stereotype, Action), according to the

energy level presented within the lake. For example, the reactive agent, Fish, can reproduce itself, feed, move, wither, or die. When the energy level is high (a precondition for the internal use case) then the Fish can execute the first three actions, but as the energy level decreases, they can no longer reproduce, feed, or even move and will begin to wither and, if the energy lowers, they will die. Figure 4 presents this aspect of the use cases diagram.

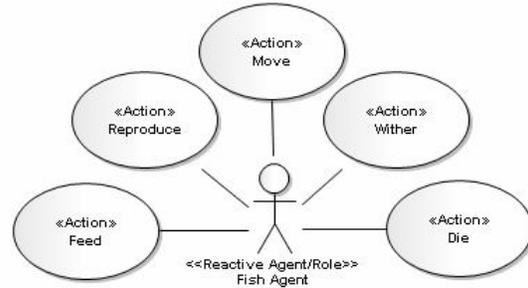


Figure 4. Reactive agents and actions modeled by stereotyped agent/role actors and stereotyped internal use-cases

Finally, we went on to the modeling of the cognitive agent, Tutor. This agent was represented as an agent/role actor with the stereotype, "Cognitive Agent/Role", and then placed inside the system. The agent, Tutor, must be able to perceive the students' mental states, to perceive the students' actions, and to probe the environment. Through this set of perceptions, the agent can determine the energy level contained in the environment (ecometer) and decide whether the goal (desire) manifested by the user to help into maintaining the energy level can become an intention and thereby trigger the plan which was associated to it.

Each of the perceptions describe above was represented as an internal use case by means of the stereotype, "Perception", as can be observed in Figure 5. The goal of aiding into the maintenance of the energy level and the plan designed to help the student were equally modelled as internal use cases with the employment of the stereotypes, "Goal", and, "Plan".

Observe that those perceptions are associated to the goal by inclusion associations. An inclusion association determines that the behavior of the use case therein included should mandatorily be executed by the use case it includes. As can be seen in the figure 2, the Include metaclass is associated with the Internal Use Case metaclass in the same way as occurs to the Use Case metaclass. These inclusions are necessary to determine whether there is a real need for triggering the plan, "Help the Student"; it is necessary to verify the students' mental states as well as to perceive which are the student's actions and to survey the environment to find out the energy level on the lake. The plan to aid the student will only be carried out when the ecometer shows the energy level is below the ideal. Notice that a "Plan extension point" was included within the internal use case along with the stereotype, Goal, and this determines in which moment and conditions the plan to aid the student can be carried out.

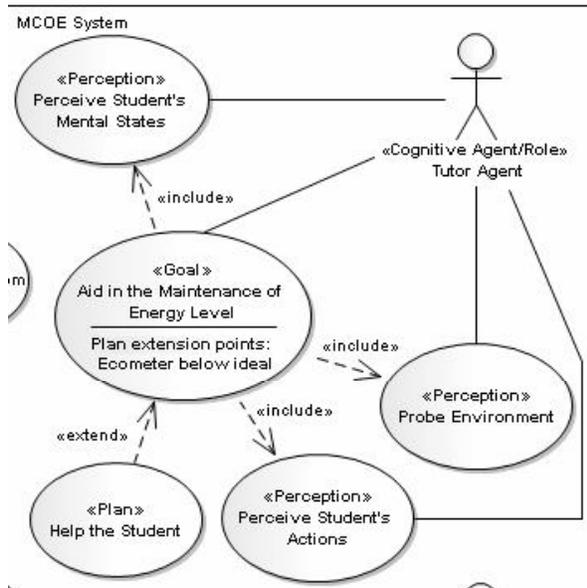


Figure 5. Cognitive Agent, Perceptions, Goal, and Plan modelled by means of Stereotyped Agent/Role Actors and Stereotyped Internal Use Cases.

The behaviour of a use case can be detailed into other diagrams, like the sequence diagram. Therefore, as can be seen in the Figure 6, we detailed the internal use case Goal, "Aid in the Maintenance of Energy Level", employing one of these diagrams.

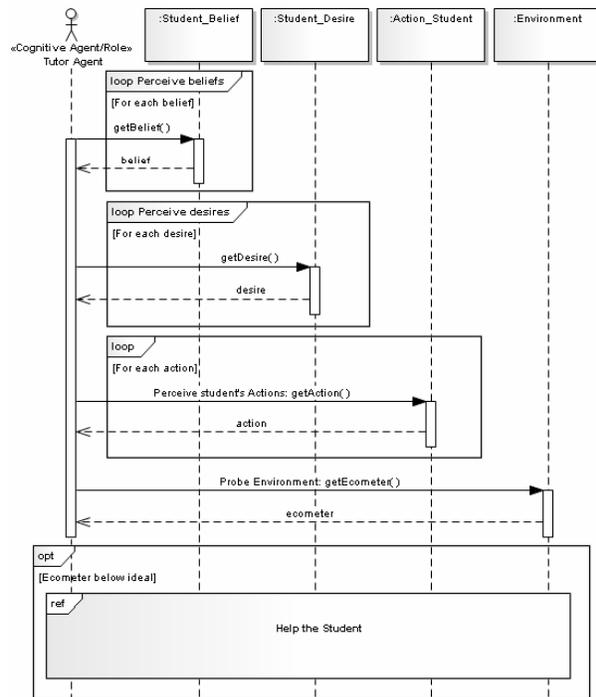


Figure 6. Detailing the Goal, "Aid in the Maintenance of Energy Level".

In this diagram we maintained the representation of the Agent, Tutor, as an agent/role actor. Actually this actor is an instance of the actor represented in the use cases diagram, just like the objects, Student-Belief,

Student-Desire, Action-Student and Environment are instances of classes defined by a class diagram named, "Student Knowledge Diagram", in which are stored, the mental states presented by the students, as well as data on the characters, the actions performed by such characters, and how much these actions can modify the lake's energy, plus those actions actually performed by each student and the level of energy contained in the environment, as can be seen in Figure 7.

Other than in previous approaches presented in this paper, the profile we are developing did not extend yet the metaclasses employed by either the class diagram or the sequence diagram, therefore all the classes and objects herein shown are normal ones.

Back to Figure 6, this diagram includes the Agent Tutor's perceptions, for these have been included into the Goal internal use case; thus, the Tutor must primarily needs perceiving the students' mental states, that is, their beliefs and desires. Thus the tutor enters a loop to retrieve all the students' beliefs and, soon after that, it enters a second loop to recover every desire expressed by the students. These loops are represented by loop-type composite fragments.

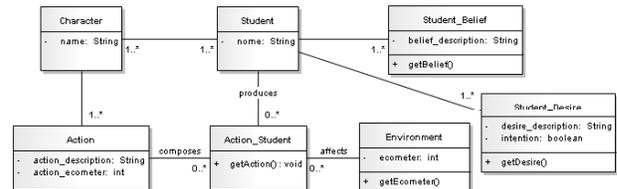


Figure 7. Student Knowledge Diagram

Next the tutor passes on to the second perception, in which becomes necessary for it to notice which are the actions performed by the students. Therefore, a third loop is executed to recover each action performed by the users. The last perception necessary is that of probing the environment so as to retrieve the lake's energy level, a task performed by triggering the method, getEcometer.

Finally we represented the Tutor's decision to trigger the plan, "Help the Student", by means of an Opt-type composite fragment, which represents an option whose contents will only be executed after some condition be satisfied. In this case, the condition is determined by the lake's energy level; whenever the ecometer shows a reading below the ideal one, the plan, "Help the Student", shall be executed. This plan is only referenced, as its detailing is presented into another sequence diagram, as shows Figure 8.

Here we represented once more the Tutor as an agent/role actor and the Student as a normal actor, while the objects were instantiated from the Tutor Knowledge Diagram, which is a class diagram whose function is that of storing beliefs, desires, and intentions, as well as the pedagogical strategies and tactics employed by the tutor, as can be seen in Figure 9.

Going back to Figure 8, the plan, "Help the Student", consists primarily into selecting a strategy followed by the selection of pedagogical tactics. These choices are based upon prior perceptions. Depending on the strategy and tactics chosen by the student and on the character

they decided to impersonate, the Tutor determines which message it is to send to the student in order to help him to improve the lake's energy level. Afterwards, the tutor registers the strategy and the tactics chosen in a historical repository so as to know which strategies and tactics have already been attempted.

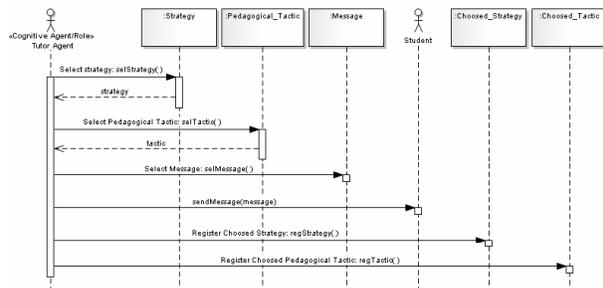


Figure 8. Detailing the Plan, "Help the Student".

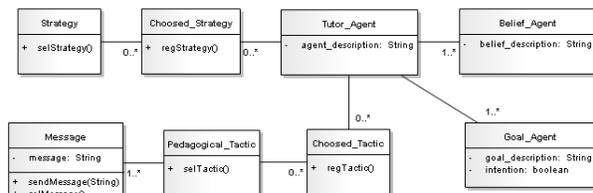


Figure 9. Tutor Knowledge Diagram

The messages sent by the Tutor agent to the student are simple pre-defined textual messages sent by the Tutor through the system to the students who, being human agents, are external to the system. If the students agents were software agents, we could employ an Agent Communication Language like FIPA, though in this case we might need to extend the sequence diagram in such a way as to model FIPA messages, like is proposed in the work of [8].

## 6. Conclusions

Throughout this paper we have presented a UML profile developed for the intelligent tutoring systems project oriented to the requirements collecting and analysing. We demonstrated the applicability of said profile by means of the MCOE System modeling and also that the stereotypes we have created can be used to model cognitive and reactive agents, and actions, perceptions, goals, and plans as well.

However, we intend to apply this profile on some other projects for intelligent tutoring systems, possibly of a more complex sort, so as to find out whether this profile is really adequate for projecting this type of systems and to pinpoint any weakness needing some improvement.

As demonstrated above, it is necessary to model structural issues, like beliefs and desires by means of class diagrams and the internal use cases detailing must be performed by the employment of a behavioral diagram. Until now we are directly utilizing these other diagrams, though we may presently extend other UML metaclasses so as to best adequate them to the intelligent tutoring systems project.

Although this profile has been developed for the intelligent tutoring systems modelling, namely for the modelling of the requirements on these systems, we believe it might be applied to other multi-agent systems projects oriented to other dominions.

Finally, we believe this profile can be adapted without difficulty to most UML already existing extensions created for the MultiAgent Systems projects and, since the studied extensions did not focus on the question of the applicability of the use cases diagram for requirements analyzing, we believe this profile can contribute to enhance them.

## 7. References

- [1] R.M. Vicari, J.C. Gluz, "An Intelligent Tutoring System (ITS) View on AOSE", International Journal of Agent-Oriented Software Engineering. (2007).
- [2] OMG - Object Management Group, "Unified Modeling Language: Infrastructure Specification - Version 2.1.1", OMG (2007), <http://www.omg.org>.
- [3] AUML Official Website, <http://www.auml.org>
- [4] I. Trencansky, R. Cervenka, "Agent Modeling Language (AML): A comprehensive approach to modeling MAS". Informatica, 29(4):391--400. (2005).
- [5] G. Wagner, "A UML Profile for External AOR Models". Third International Workshop on Agent-Oriented Software Engineering. (2002).
- [6] V. Silva, "MAS-ML: A Multi-Agent System Modeling Language". International Journal of Agent-Oriented Software Engineering, Special Issue on Modeling Languages for Agent Systems, Inderscience Publishers, vol.2, no.4. (2008).
- [7] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, J. Mylopoulos, "Tropos: An Agent-Oriented Software Development Methodology. Autonomous Agents and Multi-Agent Systems". 8, 203--236 (2004).
- [8] C. C. T. L. L. Silva, "Separating Crosscutting Concerns in Agent Oriented Detailed Design: The Social Patterns Case". Doctoral Thesis. Universidade Federal de Pernambuco. Recife (2007).
- [9] OMG - Object Management Group. "Unified Modeling Language: Superstructure Specification - Version 2.1.1". OMG (2007). <<http://www.omg.org>>.
- [10] B. Bauer, J. Odell, "UML 2.0 and Agents: How to Build Agent-based Systems with the new UML Standard.", Journal of Engineering Applications of AI. (2005).
- [11] S. Flake, C. Geiger, J. Küster, "Towards UML-based Analysis and Design of Multi-Agent Systems". Proceedings of ENAIS'2001, Dubai, March (2001).
- [12] L. Giraffa, M. Mora, and R.M. Vicari, Modelling the MCOE Tutor using a Computational Model, Lectures Notes on Artificial Intelligence - SBIA'98. (1998).

# Business-Object Oriented Requirements Analysis Framework for Data Warehouses

Anirban Sarkar<sup>1</sup>, Sankhayan Choudhury<sup>2</sup>, Nabendu Chaki<sup>2</sup>, Swapan Bhattacharya<sup>1</sup>

<sup>1</sup>National Institute of Technology, Durgapur, West Bengal, India

<sup>2</sup>Department of Computer Science, University of Calcutta, Kolkata, India

[Sarkar.anriban@gmail.com](mailto:Sarkar.anriban@gmail.com), [sankhayan@gmail.com](mailto:sankhayan@gmail.com), [nabendu@ieee.org](mailto:nabendu@ieee.org), [bswapan2000@yahoo.co.in](mailto:bswapan2000@yahoo.co.in)

## Abstract

Towards the design of successful Data Warehouse (DW) system, the requirements analysis specifications are used as the prime input for the conceptual level multidimensional data model. This paper proposed a Business Object based requirements analysis framework for DW system. Besides detail analysis of analytical requirements, the framework also provides mapping facility of requirements specifications into high level design components of graph semantic based object oriented multidimensional data model, using either Demand-driven or Mixed-driven approach for DW requirements analysis.

**Keywords:** Requirements analysis, Business objects, Conceptual Model, Data Warehouses.

## 1. Introduction

Data Warehouse (DW) and On Line Analytical Processing (OLAP) in conjunction with multidimensional database are typically used for complex, online and multidimensional analysis of data. DW projects are long-term projects. Thus the requirement analysis plays a key role within DW system development to reduce the risk of failure. Several studies [1, 2] indicate that improper analysis of user requirements or avoidance of requirements analysis phase leads to unsuccessful design of DW System. The primary deliverables of the DW requirement analysis phase is to identify the parameters towards building the conceptual level multidimensional data model schemata in design phase. Indeed, the approaches to DW requirements analysis are usually classified in *three* categories:

(a) *Supply-driven / Data-driven*: In these approaches [3, 4] the DW design starts from a detailed analysis of the operational data sources only.

(b) *Demand-driven*: These approaches [1, 5, 6] start from determining the information requirements of business users or stakeholders. In this case, conceptual level multidimensional data model design can be directly based on mapping of user requirements.

(c) *Mixed-Driven (Supply/Demand)*: In these approaches [5, 7, 8] user requirements analysis and operational data sources inspection are carried out in parallel. The Supply-driven and mixed framework is recommended when source schemata are well known.

Majority of these approaches does not addressed the mapping from requirements model to high level conceptual design. In [5] goal oriented approaches has been described to support both demand driven and mixed driven approaches. It comprehensively support early requirements analysis phase for DW system and provides a mechanism to map the requirements into the conceptual model.

In this paper, a Common Business Objects [9] based requirement analysis framework for DW system has been proposed, which consists of three phases, namely, (i) *Early Requirements Analysis Phase*, (ii) *Detailed Requirements Analysis Phase* and (iii) *Mapping Phase*. The first phase allows for modeling and analyzing the contextual setting of the business domain, in which the DW will operate. In *detailed requirements analysis phase*, the early requirements specifications are refined with the structural, functional and nonfunctional features of the domain. The refinement process is largely influenced by the concepts of Feature Oriented Domain Analysis (FODA) [11]. The *mapping phase* is used to map the DW requirement specifications to Graph Object Oriented Multidimensional Data (GOOMD) model [10]. The framework supports both demand-driven and mixed driven requirements analysis approaches for DW system.

## 2. GOOMD Model with Example

In this section we will summarize the basic concepts of GOOMD model [10]. The GOOMD model is a comprehensive object oriented model of a DW. It allows the entire multidimensional database to be viewed as a Graph (V, E) in layered organization. At the lowest layer, each vertex represents an occurrence of an attribute or measure. A set of vertices semantically related is grouped together to construct an *Elementary Semantic Group (ESG)*. On next, several related ESGs are group together to form a *Contextual Semantic Group (CSG)* – the next upper layer constructs to represent any context of business analysis. A set of vertices those determine the other vertices of the CSG, is called *Determinant Vertices*. This layered structure may be further organized by combination of two or more CSGs as well as ESGs to represent next upper level layers. From the topmost layer the entire database appears to be a graph with CSGs as vertices and edges between CSGs. *Dimensional Semantic Group (DSG)* is a type of CSG to represent a dimension

member. *Fact Semantic Group (FSG)* is a type of CSG to represent a fact and is an inheritance of all related DSGs and a set of ESG defined on measures. In order to materialize the *Cube*, one must ascribe values to various measures along all dimensions and can be created from FSG. Two types of edges have been used in GOOMD model, (i) *Link*: directed edges to represent the one – to – many associations between different CSGs and (ii) *Association*: undirected edges between constituent ESGs and determinant ESGs to represent the association within the members of any CSG.

Let consider an example, based on *Sales Application* with sales *Amount* as measure and with four dimensions – *Customer*, *Model*, *Time* and *Location*. *Model*, *Time* and *Location* dimensions have upper level hierarchies say *Product*, *QTR* and *Region* respectively. The schema from the topmost layer has shown in Fig 1.

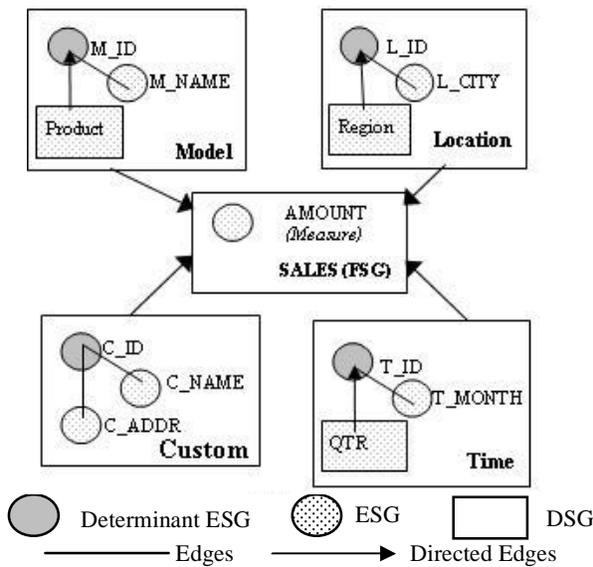


Fig 1: Schema for Sales Application in GOOMD Model

### 3. Proposed Requirements Analysis Framework

The core of the proposed requirements analysis framework for DW system is Business Object Model. A *business object* (BO) is a conceptual object that can directly describe and represent a business concept with a well-defined boundary and identity [9]. In the context of DW system, Business Object Modeling is an abstraction technique that consists of identifying the set of concepts and their contexts belong to some business domain, and representing its essential features for analytical purpose. Business Object Model is comprised of set of BOs to characterize the set of processes and activities of that specific business. The part of business object model taxonomies [9], relevant to DW system requirements analysis are as follows,

(a) **Business Object (BO)**: This is an abstraction that describes a concept of interest in the business and capable of being specialized through inheritance mechanism.

(b) **Entity BO**: This is a specialize form of BO that describes basic business concepts those are engaged in the conduct of business processes.

(c) **Process BO**: This is a specialized form of BO that describes a business process or workflow. It is comprised of Entity BOs, interactions and business events.

(d) **Event BO**: This is a specialized form of BO that describes a business event, which result from interactions between entity BOs in the context of a process BO.

Besides these taxonomies, DW context require some new taxonomies to be introduced and are as follows,

(a) **Measure Attribute**: This is an attribute that describes a quantitative aspect of business process that is relevant for decision making.

(b) **Fact BO**: This is a specialized form of Process BO to capture the concept of subject of analysis specific to some business process.

(c) **Dimension BO**: This is a specialized form of Entity BO to capture the concept of parameters over which the Fact BO will be analyzed.

The graphical notations for above taxonomies have been summarized in Table – 1.

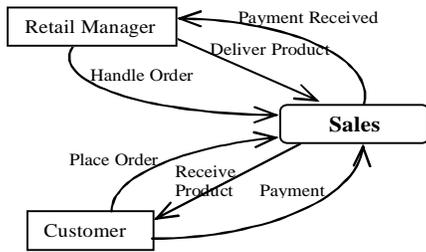
Table-1: Graphical Notations for BO based Requirements analysis Framework

Taxonomy	Graphical Notation
Entity BO	
Process BO	
Event BO	
Measure Attribute	
Fact BO	
Dimension BO	
Encapsulation	
Inheritance	
Collaboration	
Interaction Relation	

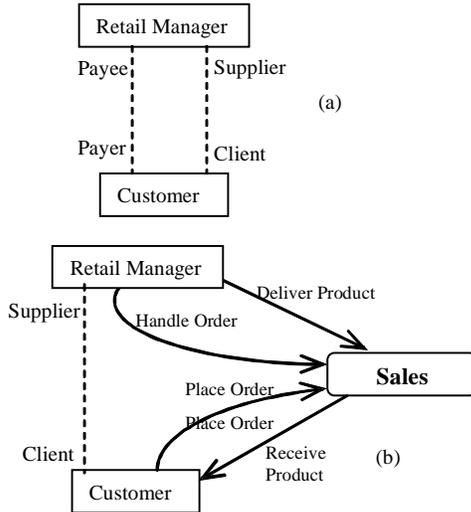
#### 3.1 Early Requirements Analysis Phase

The focus of early requirements analysis phase is to identify the business processes, different stakeholders, the interactions and collaborations between them, and the events relevant for analytical task with high level of abstraction. This phase is also the basis to identify the possible analytical requirements of decision maker or other stakeholders using business process driven approach. The phase consists of *five* steps as follows,

(a) **Identification of Process BOs, Entity BOs and interactions**: This step is to represent the relevant business processes with their context, related stakeholders and interactions between them. The business processes and stakeholders can be represented in the proposed framework using Process BO and Entity BO respectively. Interactions can be represented between Process BOs and Entity BOs. The step will result *Interactions Diagram* for each Process BO.



**Fig 2: Interaction Diagram**



**Fig 3: (a) Collaboration Diagram, (b) Collaboration Interaction Network for Supplier / Client Collaboration**

For example in a *Retail Organization* one important business process is *Sales* with two stakeholders namely, *Retail Manager* and *Customer*. The related Interaction Diagram has been shown in Fig. 2.

**(b) Identification of collaborations between Entity BOs:** In this step the collaborations are identified between Entity BOs in the context of some specific Process BO. Collaborations are occurred based on some specific roles played by participant Entity BOs. This step will result *Collaboration Diagram*. Further, for a specific collaboration we can devise the *Collaboration Interaction Network* The related diagrams for *Retail Organization* example are represented in Fig 3.

**(c) Identification of Measure Attributes:** Measure attributes related to each Process BO of interest are identified in this step. This step is the basis of *Fact BO* construction. Each Process BO relevant to analytical task can be specialized into a Fact BO by encapsulating the related measure attributes.

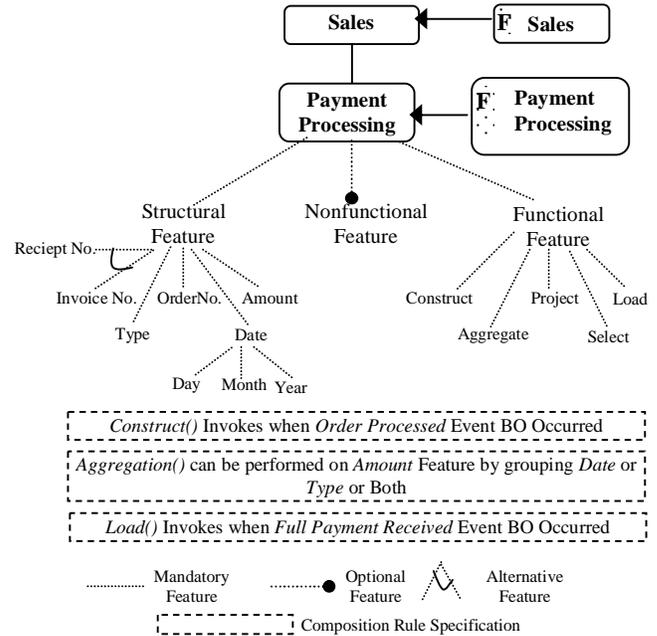
**(d) Identification of Dimension Objects:** In this step Entity BOs other than representing stakeholders are identified in the context of some specific Process BO. The set of Entity BOs identified in this step are in general captured the parameters over which the measure attributes are dependent and can be analyzed further.

**(e) Identification of Event Objects:** In this step the Event Objects are identified in the context of some specific Process BO and related Entity BOs. Identification of

Event BOs is important to understand the rationale for recording the process BO instances in the DW system for the analytical task.

### 3.2 Detailed Requirements Analysis Phase

The focus of detailed requirements analysis phase is to refine the Fact BOs and related Dimension BOs to satisfy the analytical requirements of decision makers or other stakeholders. This phase is also capable to implement the Supply-Driven part to our framework. This phase consists of *three* steps as follows,



**Fig 4: Feature Tree for the Process BO Payment Processing**

**(a) Refinement of Fact BOs:** This step consists of two levels. In the *first level*, Process BOs are refined as possible collection of related sub Process BOs. The Interaction Diagram concerned to the specific Process BO acquired from earlier phase is extended accordingly.

In the *second level* each refined Process BO can be further refined by adding the domain level feature like structural, functional and nonfunctional along with the constraint specifications. The features of some Process BO can be represented using *Feature Tree Diagram*. The feature tree exhibits the set of features using which the related Fact BO can be associated with concerned Dimension BOs for the formation of high level schema for the DW system. Further the feature tree may be accompanied with set of composition rules to express the existing semantics between the subset of features. An example Feature tree for the Process BO, *Payment Processing* has been shown in Fig. 4.

**(b) Refinement of Dimension BOs:** The set of Dimension BOs identified in the early requirements analysis phase are refined in this step. The refinement process involves *two tasks*. *Firstly*, the different levels of granularity required for the multidimensional information are identified for some specific business processes. This task facilitates the *vertical refinement* of Dimension BOs and

focus on the creation of dimension hierarchy. In the *second task*, the refined set of Dimension BOs are further refined using *Feature Tree* concept.

**(c) Compare with Operational Schemas:** This step is to implement the Supply-Driven part to our framework. The refined Process BOs and Dimension BOs resulted from the last two steps can further be modified by comparing with the existing operational schemas of the organization. To perform this step a prior knowledge of detailed analysis of operational schemas are required.

### 3.3 Mapping Phase

In this phase, the DW requirements specifications achieved from the early and detailed requirements analysis phase are mapped in the constructs and concepts of GOOMD model described in Section 2. This phase consists of two levels, namely, *Early Mapping* and *Detailed Mapping*.

Early mapping can be done just after the early requirements analysis phase for DW system. Each identified Fact BO, related Dimension BOs and encapsulated Measure Attributes will be mapped as *Fact Semantic Group (FSG, Dimension Semantic Group (DSG))* and *Elementary Semantic Group (ESG)* for measures respectively as described in GOOMD model concept. On next, each *DSG* need to be connected with the *FSG* using *Link*. The early mapping will yield the topmost layer of the GOOMD model Schema which exhibit high level abstraction.

In detailed mapping steps, firstly, the Dimension BOs of different granularity related to each top level Dimension BO are mapped as separate *DSGs* and are connected using *Link* to form the dimension hierarchies. On next, the basic structural features and identity from the Feature Tree of each Dimension BO are mapped as *ESG* and *Determinant ESG* respectively and are connected using Association. This step will exhibit all the inner layers of GOOMD model schema.

## 4. Features of Proposed Framework

**(a) Business Process Driven Approach:** The proposed framework starts with business process driven approach and is not biased from perspective of stakeholders' understanding about the system.

**(b) Object Oriented:** The proposed framework is based on Business Object Model which supports the general concepts and characteristics of object oriented paradigm.

**(c) Abstraction:** As the proposed framework is supported by object oriented paradigm, it is capable to represent the different business concepts and stakeholders' requirements in different level of abstraction.

**(d) Reusability:** The support of abstraction mechanism and feature oriented stepwise refinement of the BO based requirements descriptions enable the capability of reuse of different types of BOs in the proposed framework.

**(e) Support for both Demand-driven and Mixed-driven approach:** In detailed requirements analysis, the step for comparison with the operational schemas is to support supply-driven part in the proposed framework. By

omitting this step the framework is fully compatible with *Demand-Driven DW* requirements approach.

## 5. Conclusions

In this paper a Business Object based requirements analysis framework of DW system has been proposed. The framework is comprised of three phases namely, Early Requirements Analysis phase, Detailed Requirements Analysis phase and Mapping phase. We have described several requirements modeling elements like Fact BO, Dimension BO, Measure attributes, Interaction Diagram, Collaboration Diagram, Interaction Collaboration Network, Feature Tree etc. to express different business concepts of the domain, relevant to DW system. Finally the framework results the mapping of DW requirements descriptions in high level design components of conceptual level multidimensional data model. Moreover, the framework supports both demand-driven and mixed-driven approach.

Future work includes developing of a prototype tool in the support of the proposed framework.

## 6. References

- [1] R. Winter and B. Strauch, "A method for demand-driven information requirements analysis in data warehousing projects", *In Proc. HICSS*, pp 1359–1365, 2003.
- [2] Robert Winter, Bernhard Strauch, "Information requirements engineering for data warehouse systems", *ACM Symposium on Applied Computing*, 2004.
- [3] M. Golfarelli, D. Maio, and S. Rizzi. "The dimensional fact model: A conceptual model for data warehouses", *Intl' Jnl of Cooperative Information Systems*, Vol.7(2-3), pp 215–247, 1998.
- [4] D. Moody and M. Kortink, "From enterprise models to dimensional models: A methodology for data warehouse and data mart design", *2<sup>nd</sup> Int. Workshop on Design and Management of Data Warehouse*, 2000.
- [5] Paolo Giorgini, Stefano Rizzi, Maddalena Garzetti, "GRANd: A goal-oriented approach to requirement analysis in data warehouses", *Decision Support Systems*, Vol. 45 (1), pp 4-21, 2008.
- [6] N. Prakash and A. Gosain, "Requirements Driven Data Warehouse Development", *CAiSE Short Paper*, 2003.
- [7] Jose-Norberto Mazón, Juan Trujillo, Jens Lechtenböcker, "Reconciling Requirement-driven Data warehouses with Data Sources via Multidimensional Normal Forms", *Data & Knowledge Engineering*, Vol. 63(3), pp 725-751, 2007.
- [8] Jose-Norberto Mazón and Juan Trujillo, "An MDA approach for the development of data warehouses", *Decision Support Systems*, Vol. 45(1), pp 41-58, 2008.
- [9] OMG, Business Object DTF – Common Business Objects. OMG Document bom/97-11-11, <ftp://ftp.omg.org/pub/docs/bom/97-11-11.pdf>, 1997.
- [10] A. Sarkar, S. Choudhury, N. Chaki, S. Bhattacharya, "Conceptual Level Design of Object Oriented Data Warehouse: Graph Semantic Based Model", *INFOCOMP Journal of Computer Science*, Vol 8(4), pp 60 – 70, 2009.
- [11] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, A. Spencer Peterson, "Feature-Oriented Domain Analysis (FODA) Feasibility Study", Technical Report, Software Engineering Institute, Carnegie Mellon University (USA), 1990.

# Soft Systems in Requirements Engineering: A Case Study

Alejandra Yopez Lopez  
Computer Science and Engineering  
Mississippi State University  
Mississippi State, MS 39762  
ay68@msstate.edu

Nan Niu  
Computer Science and Engineering  
Mississippi State University  
Mississippi State, MS 39762  
niu@cse.msstate.edu

## Abstract

*Soft systems methodology should offer substantial benefits in managing expectations and requirements for a software-intensive system, but the benefits have not yet been examined empirically. This paper reports an exploratory case study investigating the hypothesis that “soft systems approach would identify all the flaws in requirements practices and suggest improvements suited to an organization’s context”. We analyzed problematic requirements practices in an ongoing software project, modeled potential changes, and asked the project team to assess the organizational fit of these changes. We conclude that soft systems methodology could indeed uncover a relatively complete set of flaws in requirements engineering, but not all changes were necessary.*

## 1. Introduction

Software requirements engineering (RE) is a “soft” human-centered activity concerned with identifying and communicating the purpose of a software-intensive system, and the context in which it will be used [6]. Soft systems methodology (SSM) aims to deal with “fuzzy” problem situations like RE, where multiple stakeholders’ diverse objectives exist [2]. In this context, requirements engineers must cooperate with the users in understanding the problem situation; however, users have yet to be fully empowered in deciding process and requirements and engineers are seen to deliver what is thought of as the users’ requirements with minimal reference to the users [1]. As a result, a large number of software projects failed. For instance, [4] estimated system failure was between 25% and 90% due to the following generic issues:

- *Correspondence failure* – the delivered system does not correspond to what was required.
- *Process failure* – a system is not forthcoming within time or resource constraints.

- *Interaction failure* – systems, as implemented, which fail to satisfy the users.
- *Expectation failure* – systems that are unable to meet stakeholders’ expectations.

RE addresses these issues more economically only if the problems can be identified. Thus, a key premise of applying SSM in RE is that holistic thinking would provide improvements to all problem issues. However, we are aware of no empirical studies that investigate this basic tenet, nor the scope of its applicability. To shorten this gap, we conducted an evaluation of applying SSM to an organization’s RE practices. Our goal was to explore what differences the use of SSM made to RE.

## 2. RE and SSM

The RE process is inherently “soft” since the needs that drive requirements are embedded in the relevant social, cultural, and organizational contexts [6]. Checkland’s SSM [2] favors human views of the problem, and has been used to study system characterized by the activities where the human behavior is a key factor that determines the result (satisfactory or not) of these activities, such as requirements elicitation and validation.

At the heart of soft systems thinking is the principle that whole entities exhibit emergent properties which are meaningful only when attributed to the whole, not to its parts [5]. Such a holistic view is particularly useful for uncovering flaws in current RE practices and enabling resolutions in an organized manner.

SSM involves users in software design [1], and [5] demonstrates the usefulness of applying SSM in the *technical* domain of designing software applications. In contrast, we explore the possibility of applying SSM in the *soft* domain of RE that identifies and communicates the purpose and the context of a software-intensive system.

### 3. Study context

The subject in our study is a research organization that specializes in scientific computing and government service. It is located in Starkville, Mississippi, and has approximately 50 employees as of 2009. To honor confidentiality agreements, we will call it SrvU. SrvU's mission is to plan and develop software to assist scientific communities and government agencies.

SrvU has a simple organizational structure where the project coordinator and the head manage the projects. The research scientists gather and analyze the information of the projects. The database administrator (DBA) manages the technological infrastructure and acts as an intermediary between the clients and SrvU's project team. The programmers develop the software system required for the project.

The case (unit of analysis) in our study is a traffic and transportation project, whose goal is to provide software support for analyzing and reporting daily activities the customer collects. Key project features include information visualization and predictive road traffic modeling. The project experienced failures in the past, mainly because the delivered product had not satisfied the user requirements. For this reason, the software has been redone several times and is currently within a new development cycle.

### 4. Methodology

We used an *exploratory case study* [8] as the basis for our research design. In our case, we were particularly interested in understanding *how* the use of SSM would affect the RE process.

We derived a central hypothesis to guide our study design: "SSM would find all the RE flaws and suggest improvements suitable for SrvU's context." To investigate this, we adapted Checkland's conventional model [2] to a more appropriate 4-stage SSM, as suggested in [5]. Figure 1 shows the adapted SSM stages.

Stage 1 examined SrvU's current RE practices. Stage 2 formulated root definitions via the mnemonic CATWOE, one of the best known SSM tools, used to define the Customers, Actors, Transformation, Weltanschauung (the worldview), Ownership, and Environmental constraints [2]. Stage 3 built conceptual models and identified potential organizational changes. Stage 4 allowed the stakeholders to debate the consequences of the proposed changes, so that the culturally feasible improvements to SrvU's RE practices could be applied.

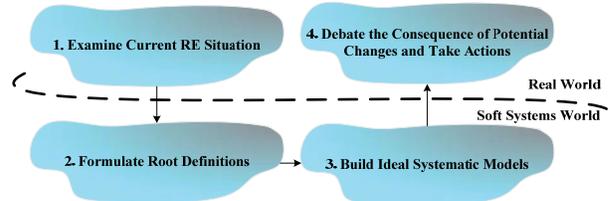


Figure 1. Adapted SSM stages

We tested our central hypothesis primarily via two evaluations, both of which were intentionally carried out in the "real world" (see Figure 1).

- *Objective assessment.* We used the four generic failure types [4] (correspondence, process, interaction, and expectation) as a baseline to check whether SSM's holistic thinking would identify a relatively complete set of RE flaws. This was done at stage 1 of Figure 1.
- *Subjective assessment.* We used questionnaires and semi-structured interviews to check whether SSM's conceptual modeling would generate RE improvements suitable for SrvU. This was done at stage 4 of Figure 1.

In our study, the data was collected mainly by means of interviews and questionnaires which involved project coordinator, DBA, and programmer. Note that we took the actual project duration into account when collecting data because some experiences and answers were only obvious in hindsight. In our evaluation, we used qualitative methods [7] to analyze the collected data. Qualitative research seeks to make sense of the way themes and meanings emerged and patterned in the data records built up from interviews and questionnaires. It is particularly suitable in our study since our collected data consisted of records of observation and interaction that were complex and contextualized.

### 5. Results

#### 5.1. Understanding current RE practices

One feature of the project under study is that an intermediary user representative participated in requirements meetings with SrvU's project team. The lack of communication with end users posed the risk that the delivered system was unable to meet their expectations [4]. SrvU's DBA occasionally had to guess the users' requirements, which would cause correspondence and interaction failures [4]. Every team member was mandated to attend the weekly meeting, coordinated by upper level personnel. This could result in software engineers' passive participation in the project, thereby causing a loss of

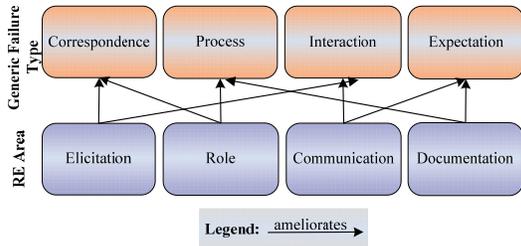


Figure 2. Addressing flaws

motivations to some extent [3]. The lack of requirements analyst’s skill set could contribute to the process failure [4].

As a result, the holistic thinking indeed uncovered a complete set of RE flaws, according to the baseline failure types (correspondence, process, interaction, and expectation) [4] employed in our study.

### 5.2. Conceptual modeling in soft systems world

We leveraged the CATWOE mnemonic [2] to formulate the SSM’s root definition, a precise description of the emergent properties of a system [5].

- *Customers (C)*: users of the software project.
- *Actors (A)*: SrvU’s project members.
- *Transformation (T)*: producing a software system fulfilling the users’ needs.
- *Weltanschauung (W)*: project members whose goals are to elicit, analyze, and validate requirements so that the high-quality software can be constructed and delivered.
- *Ownership (O)*: SrvU.
- *Environmental constraints (E)*: restrictions and standards applied to the use of information and technology framework used.

Teasing out CATWOE definitions provided a firm conceptual basis for modeling potential changes that SrvU might apply to improve its RE practices. We grouped the changes into four areas:

- *Communication*. A more direct communication with the users should adjust the expectations more properly, and greatly reduce the DBA’s guessing of what the users really want.
- *Elicitation*. The requirements should be gathered using multiple elicitation techniques [1], and reviewed by all stakeholders.
- *Role*. An explicit “requirements analyst” role should be created to better coordinate project meetings and to possibly replace the intermediary user representative.
- *Documentation*. Both SrvU and customers should agree on and maintain a formal document for contractual purposes.

Table 1. Summary of subjective evaluation

RE Area	Question	Answer	Result
Communication	Current time interval comm. with the user	All responded “ <i>more than twice a week</i> ”	Positive
	Ideal time interval communication	One said “ <i>twice a week</i> ” & others “ <i>as needed</i> ”	
Elicitation	Current elicitation techniques applied to the project	“ <i>meetings</i> ” and “ <i>personal knowledge and experience</i> ”	Positive
	Preference of other elicitation techniques	Meetings, modeling, prototyping, etc.	
Role	Having a requirement analyst role	It does not seem to be necessary	Negative
	Categorization of the role of the current intermediary user	Two responded “ <i>very effective</i> ” & one “ <i>somewhat effective</i> ”	
Documentation	Presence of requirements documentation	“ <i>(Yes,) under contract</i> ” & “ <i>(Yes,) in the initial (project) proposal</i> ”	Positive
	Is it useful?	All responded “ <i>Yes</i> ”	
	Opinion about the best way to document	Lightweight formats, e.g., e-mails, feature list	

The above areas for improvements were modeled by the first author of this paper. It is our belief that these changes stemmed from the soft systems world would overcome SrvU’s RE flaws in a culturally feasible way. Figure 2 shows how the RE areas could address different types of software failures [4].

### 5.3. Assessing changes

The suggested changes must be critically evaluated before SrvU’s adoption. We designed a questionnaire to guide our semi-structured interviews with SrvU’s project coordinator, DBA, and programmer. Each of the nine questions covered one area of the RE improvements, as shown in Table 1. The three interviews were conducted separately, ensuring non-interference with other team members’ opinions. Each interview lasted approximately 20 minutes.

We analyzed interview transcripts and completed questionnaires by coding (relating answer sections to proper subject matters under testing) and categorizing (classifying answers to be positive or negative) [7]. Table 1 summarizes the results. Note that SrvU’s project members’ quotes are represented *italic* and cited in double quotation marks (“”).

SrvU’s project team found changes in three areas to be acceptable. In particular, “*the communication with the users should be as needed just to make sure they are in the loop if we have questions.*” Although common elicitation techniques were meetings and personal knowledge, the project team realized the importance to incorporate other techniques like prototyping and modeling. Proper documentation was

crucial not only for contractual purposes, but also for “keeping everyone on the same page.” In this regard, lightweight formats like feature list were preferred.

To our surprise, the idea of setting up the “requirements analyst” role was not appealing to SrvU. According to some project members, SrvU enjoyed the culture of small project teams, supported by their flat organizational structure. They regarded the current division of labor, especially with the intermediary user representative, as effective. They were concerned that adding another role would increase latency and reduce the throughput of SrvU’s business processes.

Our work has provided SrvU much insight into their RE practices. Although we cannot claim that our results have had direct impacts on all SrvU’s RE processes, we can claim to have initiated changes in some areas. Already, SrvU has introduced rapid prototyping to the project’s current iteration, and also started exploring more flexible ways to document system features.

## 6. Threats to validity

Several factors can affect the validity of our exploratory case study. *Construct validity* concerns establishing correct operational measures for the concepts being studied [8]. The main constructs in our case study are “SSM”, “all the RE flaws” and “improvements suitable for SrvU’s context”. As for the first construct, our 4-stage design, shown in Figure 1, was in line with many SSM studies, e.g., [5]. As for the second construct, we do not feel that using the generic software failure types [4] as the baseline posed a serious limitation. As for the third construct, due to the lack of metrics for organizational fit, our best measure came from the subjective opinions of SrvU’s project members.

Regarding *internal validity* [8], a major limitation of our study design is the SSM modeling skills and experiences of the researcher, which compounds the problem of experimenter bias [8]. We plan to address the threat by involving multiple experts and stakeholders in developing conceptual models. Another likely confounding variable is the interview data. Participants in our current study may have omitted important facts when answering questions or we may have misinterpreted the data. This threat was mitigated by on-site ethnographic observation, as well as applying pre-defined qualitative data analysis methods (coding and categorizing) jointly by the two authors of this paper.

The results of this study might not generalize beyond SrvU’s organizational conditions and its traffic

and transportation project’s situational characteristics, a threat to *external validity* [8]. Nevertheless, the ongoing industrial-strength project, together with the participation of software professionals, provided a firm footing for applying SSM in RE. Finally, in terms of *reliability* [8], we expect replications of our study to offer results similar to ours. Of course, the experience of SSM modelers may differ, but the underlying trends and implications should remain unchanged.

## 7. Conclusions

This case study was set up to investigate the role of SSM in understanding RE practices. To that end, we discussed some fundamental aspects of soft systems approach in relation to RE. We then studied a socio-technical software system developed by a research organization. We found that SSM’s holistic thinking could indeed identify a relatively complete set of RE flaws. However, we were unable to confirm that all SSM changes would fit in the organization’s context.

From our experience, we feel that SSM has a rich value in scrutinizing and improving the human-centered RE activities. More in-depth empirical studies are needed to lend strength to the preliminary findings reported here. Our future work also includes helping SrvU to select a proper set of requirements elicitation techniques, analyze coordination patterns, and tackle requirements prioritization and evolution problems.

## References

- [1] I. Alexander, “Migrating towards co-operative requirements engineering”, *Computing & Control Engineering Journal*, 9(1): 17-22, 1999.
- [2] Checkland, P., *Systems Thinking, Systems Practice*, John Wiley & Sons Ltd., 1981.
- [3] T. Hall, N. Baddoo, S. Beecham, H. Robinson, and H. Sharp, “A Systematic Review of Theory Use in Studies Investigating the Motivations of Software Engineers”, *ACM Transactions on Software Engineering and Methodology*, 18(3): 1-29, 2009.
- [4] K. Lyytinen and R. Hirschheim, “Information systems failures – a survey and classification of the empirical literature”, *Oxford Surveys in Information Technology*, 4(1): 257-309, 1987.
- [5] L. Mathiassen, A. Munk-Madsen, P.A. Nielsen, and J. Stage, “Soft Systems in Software Design”, In: M.C. Jackson, G.J. Mansell, R.L. Flood, R.B. Blackham, and S.V. Probert (eds.), *Systems Thinking in Europe*, Basic Books, pp. 317-327, 1991.
- [6] B. Nuseibeh and S. Easterbrook, “Requirements Engineering: A Roadmap”, *Proc. of the Conference on the Future of Software Engineering*, pp. 35-46, 2000.
- [7] Richards, L., *Handling Qualitative Data: A Practical Guide*, Sage Publications, 2005.
- [8] Yin, R.K., *Case Study Research: Design and Methods*, 3rd ed. Sage Publications, 2003.

# Textual Software Requirements Specifications in the Context of Software Architecting

Matthias Galster  
University of Calgary  
Canada  
mgalster@ucalgary.ca

Armin Eberlein  
American University of Sharjah  
United Arab Emirates  
eberlein@ucalgary.ca

Mahmoud Moussavi  
University of Calgary  
Canada  
moussam@ucalgary.ca

## Abstract

*Textual software requirements can be specified in various ways. However, not all types of textual specifications might be an appropriate starting point for creating software architectures. Thus, this paper investigates different formats of textual software requirements and tries to identify a suitable input for creating architectures. Case studies involving human experts (practitioners) are performed to determine what type of specification is preferred in the context of software architectures. Four different software systems are used; each system is described in a different textual requirements format. Results show that not all types of textual specifications are considered suitable as input for the architecture process. Practitioners found atomic requirements and requirements that include design constraints as most appropriate. Thus, the architecture process should start with a cohesive set of individual requirements, rather than a plain, high-level system description.*

## 1. Introduction

Software architectures define the structure of a software system. In the context of this paper, an architecture is understood as a milestone in a software development project created after specifying requirements (and evolving as development progresses), rather than an entity that starts evolving when coding starts [1]. A good architecture is the foundation for software systems that implement all requirements within budget and at an acceptable level of quality (e.g., with regard to performance, maintainability, functionality). Moreover, software architectures are the foundation for subsequent development tasks, such as implementation, testing or maintenance.

Software architectures are created based on software requirements. Such requirements can be specified in many different ways, such as natural language, models (e.g., use case diagrams) or more formal approaches (e.g., the User Requirement Notation) [2]. Natural language requirements are widely used in practice and often supported by tools (e.g., Rational Doors®) because text can be used universally to convey information and to communicate between different stakeholders [3]. However, natural language requirements that are used as direct input for software architecting could be expressed in different ways, such as full text, brief paragraphs or atomic requirements (i.e., requirements which describe exactly one thing and that cannot be refined any further [4]). Therefore, this paper addresses the question of what textual requirements format is considered appropriate as input for an architecture process, from a practitioner's point of view. In detail, we are interested in two questions:

**Question 1:** Do practitioners prefer one type of textual requirements specification over another?

**Question 2:** If the answer to the previous question is positive, what type of specification is preferred by practitioners?

Answering these questions is helpful for developing practical and effective techniques, methods and tools for transforming requirements into architectures.

To answer the above two questions, we present four case studies with four software systems from literature. For these systems, we created different types of textual requirements specifications. Human experts from four organizations were asked to evaluate the specifications and to identify the requirements specification they prefer as input for software architecting.

Textual requirements specifications have known weaknesses (e.g., semantic vagueness, lack of automated processing). Moreover, formal or graphical requirements definitions are widely used in software engineering. However, this paper focuses on textual

specifications due to the following reasons: First, textual requirements are often used in requirements tools, such as Doors®. Second, textual requirements are widely used in practice and supported by requirements approaches, such as Volere, or in agile methods which use story cards to “document” requirements. Third, graphical requirements representations are often supplemented by textual descriptions. Moreover, natural language text is often the first way to document requirements and used to express requirements graphically later on.

The rest of this paper is organized as follows: Section 2 presents related work. Section 3 discusses the case study setting. In Section 4, we present the results of the study and a discussion. Limitations of our study are outlined in Section 5. Section 6 concludes the paper and presents future work.

## 2. Related work

In recent years, research has been conducted on relating software requirements and architectures [5]; however, we could not find any study that investigates the impact of textual requirements specifications on the architecture process.

Real-world influences on architectures have been investigated by Mustapic et al. [6]. Influences include reuse, legacy, technology, standards, the developing organization and the development process. According to the study, the development process, which includes requirements specification, has been found as being not a direct concern of architects [6].

Kral and Zemlicka investigated what strategy (e.g., data-orientation or object-orientation) for requirements specifications should be used under what conditions [7]. The study explored the impact of those strategies on the architecture process and concluded that strategies mainly impact the structure of requirements specifications. However, how to specify requirements from an architectural perspective was not investigated.

The impact of requirements specifications on project success was studied by Knauss et al. [8]. Rather than requirements specification types, this study investigated the formal quality of requirements (i.e., verbalization rules, such as grammar, rules of expression) and content-related requirements quality (such as completeness or redundancy).

Condori-Fernandez et al. investigated requirements specification approaches [9]. An empirical study was presented to understand what aspects of requirements specifications are evaluated and by which method. It was found that understandability of requirements is the most common aspect evaluated in software

requirements specifications. However, no relation to software architectures was made.

Requirements specification methods from an architecture perspective have been studied by Bass et al. [10]. The study investigated five methods for eliciting and specifying requirements, including requirements specification using natural language (other methods included use case analysis, Quality Attribute Workshop [11], global analysis [12]). According to the study, the benefit of natural language requirements is that they are understandable by a variety of stakeholders and therefore commonly used. The study only analyzed natural language requirements in general, rather than different types of specifying natural language requirements.

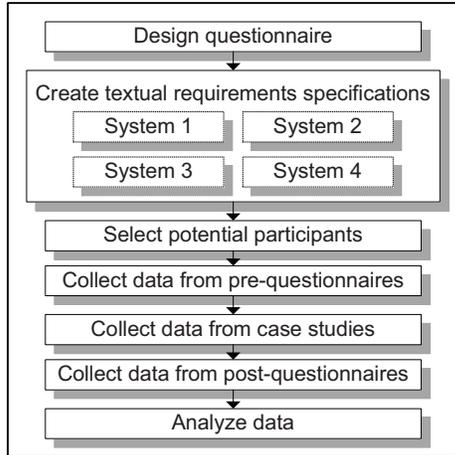
## 3. Case study setting

This paper describes four case studies with four different software systems. Eight participants representing four software development organizations were involved in the study. This allowed us to collect stronger evidence and acquire a broader understanding of the practical meaning of software requirements in the context of architectures, compared to the information we would get from a highly specialized single participant. Our study had one unit of analysis: the software requirements specification [13].

### 3.1. Preparation of study

The flow of the study is outlined in Figure 1. The first step for preparing the study was the design of the questionnaire used for data collection throughout all four case studies. Some questions were open-ended, whereas other questions provided possible answers with the choice to opt-out. The questionnaire was semi-structured. It was initially prepared by one author and reviewed by all authors. After test trials, reviews were carried out. Due to space limitations, we exclude details on the questionnaire.

The second step for preparing the study was creating textual requirements specifications for the four software systems (see Table 1). Each system was described in a different requirements format. The reason for having four different systems for four types of specifications was that we tried to avoid a learning effect when judging one specification after the other. Also, all systems had similar degrees of complexity to reduce the probability of participants preferring one specification because of the different complexity of the system described by this specification.



**Figure 1. Outline of study**

The initial problem statements of all four systems were taken from literature: System 1 in Table 1 is based on Feather et al. [14], Systems 2 to 4 are based on Shaw and Garlan [15]. We used these problem statements as basis for creating the requirements specifications in our study.

**Table 1. Requirements descriptions**

#	Description (case)	Textual requirements format
1	Meeting Scheduler	One page plain text
2	Mobile Robot System	Brief introductory paragraph + list of design constraints
3	Keyword in Context System	Brief description of system in one paragraph
4	Cruise Control System	Atomic requirements

Table 1 includes four types of textual requirements specifications. System 1 was described using one page of plain text. This type of specification is widely used especially in small software organizations with a low software process maturity level. System 2 was described using one brief paragraph introducing the system, together with a list of design constraints. This type of requirements specification is often used in organizations which start architecting without a dedicated software requirements engineering process but consider the software as a highly evolving entity throughout development. In our study, design constraints for System 2 included issues such as support for uncertainty in the Mobile Robot System. This constraint is based on the circumstances of the robot's operation which must ensure that the robot can act even with incomplete and unreliable information. System 3 used a brief description of the system in one paragraph. The motivation for this type of requirements specification is that architecture decisions are often considered as early decisions without detailed

information about the system. System 4 was described using atomic requirements, i.e., requirements which express only one thing. This type of requirements has been presented in literature in the context of software architecting [2].

### 3.2. Evidence chain

Participants of the study included eight experts from four organizations. A pre-questionnaire ensured that all participants had appropriate software engineering knowledge, in particular in requirements engineering and software architectures. Interestingly, the pre-questionnaire also revealed that all participants experienced problems with requirements and software architectures in previous projects. Participants were chosen from two academic organizations and two industrial firms (for confidentiality reasons, we do not include further details on the organizations). Participants from academia had at least one year of relevant industry experience.

All representatives from all four organizations evaluated the requirements specifications of all four systems. Participants were asked to evaluate the specifications and to determine the specification they preferred as starting point for architecting. We did not ask participants to rank the appropriateness of a specification. We wanted to simulate a real-world setting where one requirements format has to be chosen, rather than trying different formats.

To strengthen our understanding of textual requirements specifications and their appropriateness for software architecting, we also asked participants to fill out a post-questionnaire. This allowed participants to provide additional comments.

## 4. Discussion of results

This section discusses the results of our four case studies. This includes a discussion on the requirements processes used in the organizations of participating experts, and the relation of these processes to the requirements specification chosen by representatives of these organizations.

### 4.1. Quantitative analysis of results

It could have been expected that System 2 would be preferred by experts as it included design constraints. However, 50% of the experts (4 out of 8) chose System 2 and 50% chose the description with atomic requirements (System 4) as starting point for software architecting (note: there was always agreement between

experts that came from the same organization, even though participants individually evaluated specifications). This also means that there was 100% agreement that the specifications of Systems 1 and 3 are not an appropriate input for software architecting.

To determine the degree of agreement between study participants and to find out if this agreement was beyond chance, we applied Fleiss' kappa [16]. kappa is a measure for the chance-corrected agreement between participants, each of whom independently evaluated the specifications. The results of this analysis are summarized in Figure 2.<sup>1</sup>

Line	Output
1	Fleiss' (overall) kappa = 0.2381
2	kappa error = 0.0945
3	kappa C.I. (95%) = 0.1899 0.2863
4	Fair agreement
5	Reject null hypothesis: observed agreement is not accidental

**Figure 2. Output of reliability analysis**

The output of Figure 2 and the meaning of the lines are explained in Table 2. Figure 2 (lines 4 and 5) shows that the agreement is beyond chance. Thus, the results of our study can be considered as fairly reliable.

**Table 2. Explanation of the output in Figure 2**

Line	Description
1	Fleiss kappa calculated based on [16]
2	kappa standard error
3	kappa confidence interval
4	Agreement based on kappa benchmarks proposed by Landis and Koch [17]
5	z test (conclusion)

## 4.2. Case study 1: Meeting Scheduler

As discussed in Section 4.1, System 1 was not chosen by any participant or organization. According to some participants, the one page plain text requirements specification was too elaborate and contained too much information that would not be necessary for software architecting. However, this might depend on where developers draw the line between architecting and software design.

Case study 1 led to the observation that plain textual specifications are not a suitable input for creating software architectures.

<sup>1</sup> For the analysis we used a Matlab package available at <http://www.mathworks.com/matlabcentral/fileexchange/15426>.

## 4.3. Case study 2: Mobile Robot System

In our study, participants considered non-functional requirements as a critical factor in software architecting. This has also been shown in other studies [18, 19]. Participants stated that non-functional requirements are difficult to map on architecture elements. Moreover, it was stated that during software architecting, many decision variables have to be taken into consideration. Participants found that these concerns were addressed by the requirements specification of System 2, which emphasized design constraints. Thus, this specification provided a kind of “checklist” for critical success factors and focused on what is needed and what is important in an architecture. Moreover, according to some participants, System 2 allowed the use of design constraints as “keywords”, e.g., for selecting architecture styles based on the requirements specification.

Organizations 2 (large industrial organization) and 4 (small academic organization) had a less mature software process, following generic development practices. Interestingly, representatives from these two organizations preferred the specification of System 2.

Case study 2 led to the observation that brief summaries of design constraints can be a suitable input for software architecting, preferred by software development organizations with less mature processes.

## 4.4. Case study 3: Keyword in Context System

The brief one-paragraph requirements specification of System 3 was considered to be too short by many participants. Thus, no expert chose this specification as starting point for architecting. However, in general, this is debatable as it could be argued that software architectures include early design decisions which do not require detailed knowledge about the system.

Case study 3 led to the observation that brief descriptions of software systems are not an appropriate input for software architecting.

## 4.5. Case study 4: Cruise Control System

Several participants stated that the requirements specification of System 4 (atomic requirements) was clear and easy to understand. Thus, it met many of the classical requirements qualities (e.g., understandability, feasibility) described in literature [8].

Organization 1 was a large industrial organization which used the Volere requirements process in many projects [4]. Volere describes each requirement on a so-called Snow Card and prescribes a fit criterion for

each requirement. Such a fit criterion helps tighten and determine if a requirement is met. A fit criterion is applied to individual requirements. Therefore, atomic requirements are implicitly used by Volere. It is even stated by Volere that requirements specifications for design should contain detailed definitions of “non-decomposable” requirements. Thus, it is not surprising that participants from Organization 1 found atomic requirements as most appropriate for architecting.

Organization 3 was a medium-scale academic organization. It followed an agile software development process, utilizing mainly practices from Extreme Programming (XP) [20]. Again, atomic requirements were favored by this organization. Atomic requirements are in line with user stories and could easily be assigned to individual developers.

Case study 4 led to the observation that atomic requirements are considered as an appropriate input for software architecting, in particular by organizations with mature development practices.

#### 4.6. Summary of results

The results discussed in the previous section provided answers to the questions stated in the introduction: First, practitioners consider some requirements specifications as less suitable as input for the architecture process than others. Second, atomic requirements and a summary of design constraints are considered as most appropriate.

Case studies 2 and 4 raise the question why more mature organizations chose atomic requirements, whereas organization with less mature processes chose specifications which focus on design constraints. One reason could be that atomic requirements require a disciplined process that only exists in mature organizations. Moreover, atomic requirements might allow better project control which again is a criterion for mature development practices.

Our study could not conclusively answer why atomic requirements were preferred by practitioners or why requirements specifications were rejected. However, some anecdotal evidence suggests that benefits of atomic requirements stem from several areas: First, atomic requirements can directly be related to architecture elements (e.g., components, connectors) and thus can be traced easier compared to non-atomic requirements or free text which could spread across architecture elements. Second, as suggested elsewhere, atomic requirements can be replaced easier in specifications compared to non-atomic requirements [2]. Third, due to their semi-structured nature, atomic requirements address weaknesses of textual

requirements, such as semantic vagueness, bad readability or problems with automated processing.

### 5. Threats to the validity of the study

Internal and external validity as well as reliability are critical issues in empirical studies.

**Internal validity:** We focused on the requirements specification as the input for software architectures. We assumed that the format of the requirements specifications remains the same when requirements change. This means, if requirements are added or manipulated, they would be in the same format as the original specification. However, the change process or iterations were not a focus of our study.

**External validity:** We only used a small number of case studies but each one was of reasonable size. Also, due to the required expertise, only eight participants were involved. However, it is not uncommon to have this number of experts in studies that require special expertise. This is similar to Hickey and Davis, who used nine experts [21] or Liu et al. who used eight experts [13] to investigate requirements elicitation or requirements-architecture issues, respectively. Also, some experts expressed the opinion and experience of the team in the organization in which they develop software and not only their own experience.

**Reliability:** In particular with a small number of participants, the reliability of the study might be biased. However, by applying kappa in Section 4.1, we tried to determine that there was reasonable agreement beyond chance.

Moreover, we acknowledge that some discussions in Section 4 are subject to speculation (e.g., why do more mature organizations prefer atomic requirements). We believe that this is to be confirmed in further studies.

### 6. Conclusions and future work

This paper presented four case studies to investigate textual requirements specifications in the context of software architecting. It was found that practitioners do not consider all textual descriptions as equally suitable as input for software architecting. Moreover, atomic requirements were considered as most appropriate, in particular by more mature software development organizations. Thus, before starting the architecture process, proper requirements engineering should be performed in order to specify requirements in a way that supports the creation of architectures. In particular, providing a list of atomic requirements combined with a summary of high-level design constraints might be very useful input for the architecting process. Using the

evidence of our study is helpful for driving research and practice in software requirements engineering and software architectures as the right input for software architectures can facilitate the transition between requirements and architectures.

Our future work focuses on two areas. First, we plan to include more organizations in the study in order to get more statistically significant results. We hope to get more solid findings to understand why participants preferred the specifications they did, rather than providing anecdotal evidence as included in Section 4. Second, we plan to expand the types of requirement descriptions from textual descriptions to graphical notations.

## 7. Acknowledgements

The authors thank the participants of the study.

## 8. References

- [1] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*. Boston, MA: Addison-Wesley, 2003.
- [2] M. Galster, A. Eberlein, and M. Moussavi, "Atomic Requirements for Software Architecting," in *Proc. International Conference on Software Engineering and Applications (SEA'07)*, Cambridge, MA, 2007, pp. 143-148.
- [3] L. Mich, M. Franch, and P. Inverardi, "Market Research for Requirements Analysis Using Linguistic Tools," *Requirements Engineering*, vol. 9, pp. 40 - 56, February 2004.
- [4] S. Robertson and J. Robertson, *Managing the Requirements Process*. Harlow, England: ACM Press, 1999.
- [5] M. Galster, A. Eberlein, and M. Moussavi, "Comparing Methodologies for the Transition between Software Requirements and Architectures," in *Proc. IEEE International Conference on Systems, Man, and Cybernetics (SMC'09)*, San Antonio, TX, 2009, pp. 2454-2459.
- [6] G. Mustapic, A. Wall, C. Norstrom, I. Crnkovic, K. Sandstrom, J. Froberg, and J. Andersson, "Real World Influences on Software Architecture - Interviews with Industrial System Experts," in *Proc. Working IEEE/IFIP Conference on Software Architecture*, Amsterdam, Netherlands, 2004, pp. 101-111.
- [7] J. Kral and M. Zemlicka, "Requirements Specification: What Strategy Under What Conditions," in *Proc. 5th ACIS International Conference on Software Engineering Research, Management and Applications (SERA 2007)*, Busan, Korea, 2007, pp. 401-408.
- [8] E. Knauss, C. E. Boustani, and T. Flohr, "Investigating the Impact of Software Requirements Specification Quality on Project Success," in *Proc. 10th International Conference on Product-Focused Process Improvement (PROFES)*, Oulu, FI, 2009, pp. 28-42.
- [9] N. Condori-Fernandez, M. Daneva, K. Sikkel, R. Wieringa, O. Dieste, and O. Pastor, "Research Findings on Empirical Evaluation of Requirements Specification Approaches," in *Proc. Workshop on Requirements Engineering (WER'09)*, Valparaiso, Chile, 2009, pp. 121-128.
- [10] L. Bass, J. Bergey, P. Clements, P. Merson, I. Ozkaya, and R. Sangwan, "A Comparison of Requirements Specification Methods from a Software Architecture Perspective," SEI CMU, Pittsburgh, Technical Report CMU/SEI-2006-TR-013, 2006.
- [11] M. R. Barbacci, R. J. Ellison, A. J. Lattanze, J. A. Stafford, C. B. Weinstock, and W. G. Wood, "Quality Attribute Workshops (QAWs), Third Edition," SEI CMU, Pittsburgh, Technical Report CMU/SEI-2003-TR-016, 2003.
- [12] C. Hofmeister, R. Nord, and D. Soni, *Applied Software Architecture*. Reading, MA: Addison-Wesley, 2000.
- [13] W. Liu, C. L. Chen, V. Lakshminarayanan, and D. E. Perry, "A Design for Evidence-based Software Architecture Research," in *Proc. Workshop on Realising Evidence-based Software Engineering*, St. Louis, MO, 2005, pp. 1-7.
- [14] M. S. Feather, S. Fickas, A. Finkelstein, and A. van Lamsweerde, "Requirements and Specification Exemplars," *Automated Software Engineering*, vol. 4, pp. 419-438, October 1997.
- [15] M. Shaw and D. Garlan, *Software Architecture: Perspectives from an Emerging Discipline*. Upper Saddle River, NJ: Prentice Hall, 1996.
- [16] J. L. Fleiss, "Measuring Nominal Scale Agreement Among Many Raters," *Psychological Bulletin*, vol. 76, pp. 378-382, November 1971.
- [17] R. Landis and G. G. Koch, "The Measurement of Observer Agreement for Categorical Data," *Biometrics*, vol. 33, pp. 159-174, March 1977.
- [18] X. Franch and P. Botella, "Putting Non-Functional Requirements into Software Architecture," in *Proc. 9th International Workshop on Software Specification and Design*, Ise-Shima, Japan, 1998, pp. 60-67.
- [19] S. Kim, D.-K. Kim, L. Lu, and S. Park, "A Tactic-Based Approach to Embodying Non-functional Requirements Into Software Architectures," in *Proc. 12th International IEEE Enterprise Distributed Object Computing Conference (ECOC 2008)*, Munich, Germany, 2008, pp. 139-148.
- [20] K. Beck, *Extreme Programming Explained*. Boston, MA: Addison-Wesley, 1999.
- [21] A. M. Hickey and A. M. Davis, "Elicitation Technique Selection: How Do Experts Do It?," in *Proc. 11th IEEE International Requirements Engineering Conference*, Monterey Bay, CA, 2003, pp. 169-178.

# Conditions for ignoring failures based on a requirements model

João Pimentel<sup>1</sup>, Emanuel Santos<sup>1</sup>, Jaelson Castro<sup>1</sup>

<sup>1</sup>Universidade Federal de Pernambuco, Centro de Informática, Recife, Brazil  
{jhcp, ebs, jbc}@cin.ufpe.br

## Abstract

*Ideally, all system failures should be compensated. In fact, most failure-prone systems try to compensate all their failures. However, sometimes a compensation is not essential. Hence, diagnosing and compensating each and every one of their failures may be ineffective.*

*Thus, this work aims to increase the flexibility of failure handling in self-configuring systems, using tolerance policies based on requirements models. We allow the expression of conditions in which certain failures may be ignored – i.e., conditions in which a failure will not be compensated. Such policies may lead to reduced costs and performance improvement.*

*The FAST<sup>1</sup> framework consists of the definition of a tolerance policy, the mechanisms to evaluate this policy and a tool to aid the creation and maintenance of policies. We use a Smart Office system to show the several types of policy rules in action.*

## 1. Introduction

The increasingly complexity of software systems led to the proposal of Autonomic Computing, in which the systems are capable of maintaining its ideal behavior requiring only a minimal amount of human intervention, even in dynamic environments [1]. Autonomic systems present four basic characteristics: self-configuration, self-healing, self-protection and self-optimization [2][3]. In particular, self-configuration is seen as the main characteristic [4][5], partially because of the support it provides for implementing the other characteristics.

In this work we are considering a specific architecture for self-configurable systems [6], in which the system execution is monitored at the requirements level. This architecture performs a Monitoring – Diagnosing – Compensating (MDC) cycle. It monitors the execution of a system, diagnoses the failures that may happen and proposes reconfigurations in order to

avoid these failures. This monitoring and diagnosing is performed regarding the system requirements, expressed with goal models. The goals are modeled using the Tropos notation [7], which captures the social and intentional relationships in the system organizational environment, as well as quality attributes and functionalities of the system.

Ideally, all system failures should be compensated. However, sometimes compensation is not essential - it depends on the failure's criticality. For instance, let us consider that a research group has weekly meetings every Wednesday. If one of the meetings happens to coincide with a holiday, the group may just cancel that meeting and gather together in the following Wednesday. On the other hand, if there are three consecutive cancelations of the meeting (due to holidays or other motives), the time gap between one meeting and the next one would be too large. Hence, it would probably be better to reschedule some meetings to an alternative day of the week (eg. Thursdays). In this scenario, the failure – cancellation of a meeting - does not always need to be compensated. For example it is allowed to happen two times in sequence, but no more than that.

In contrast, the MDC cycle expects that each system failure will lead to compensation. Thus, this work aims to increase the flexibility of failure handling in self-configuring systems, allowing the expression of conditions in which certain failures may be ignored – i.e., conditions in which a failure will not be compensated. To discover the types of conditions, we performed an extensive analysis of goal models presented in the academic literature.

The concept of policies is used in Software Engineering to allow users or system administrators to control some characteristics of a system, without having to deal with implementation details [8]. In particular, this concept has often been used by the network community [9][10]. In this work we are defining a policy to enable the customization of the way that a system handles its failures.

This paper is organized as follows. Section 2 presents the background of this research, which is a

---

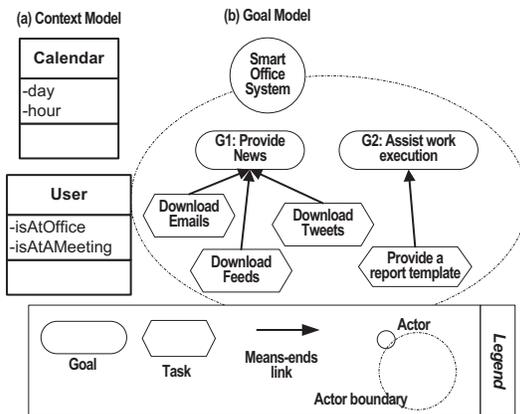
<sup>1</sup> Failure handling for Autonomic Systems

context-enriched Tropos notation. Section 3 presents our approach for expressing conditions in which a failure may be ignored – namely, the Tolerance Policy. The algorithm for processing this policy is presented in Section 4. Section 5 illustrates the use of our approach and the tools developed to support the policy. In section 6 we compare our research with related works. Finally, Section 7 summarizes our work and points out open issues.

## 2. Background

Our architecture is based on some previous work [6] which considers the requirements model as a goal model and a context model. With this information and the data provided during a system execution a self-configuration component is able to monitor and diagnose failures at runtime.

A goal model is a model that depicts the intentions of actors in a system, along with the means - tasks - to achieve these goals and the interdependencies among the actors. In particular, the self-reconfiguration architecture [6] adopted uses a Tropos [7] goal model consisting of actors, their goals, goals and/or-decompositions, tasks, means-end links (from a task to a goal) and dependency links (from an actor to another actor). Below (Figure 1) we describe the example to be used throughout the paper.



**Figure 1 - Goal and context model for a Smart Office System**

The example in Figure 1 (b), of a Smart Office system, shows some of these concepts. This system has two goals: provide news (to its user) and assist (a user) when performing his work. The *Download Emails*, *Download Feeds* and *Download Tweets* tasks are means to the *Provide News* end. Similarly, the *Provide a report template* task is a means to the *Assist work execution* end. We are considering that when downloading e-mail an Ethernet connection is used.

When this connection is down, the reconfiguration strategy is to connect to the Internet through a mobile phone - which is more expensive than using the Ethernet connection.

The context model defines the data that context sensors will have to monitor, in order to assess the tasks execution. In Figure 1 (a) we show the context model for the Smart Office system. In this example, the context sensors would need to know the day and current hour of the calendar and if the user is at his/her office or at a meeting.

## 3. Tolerance Policy

Given the overview of the models we are taking into consideration, in this section we are going to describe our tolerance policy. It is concerned with the definition of conditions for task failures to be ignored. A failure, in this case, is the unsuccessful execution of a task. By default, on the MDC cycle, all failures must be compensated through some reconfiguration - only those tasks explicitly mentioned in some rule of this policy will have its failures disregarded. Failures will be ignored depending on conditions that may be related to the system's context, to the system goals or to the amount of time elapsed since the occurrence of a failure. For each of these types of conditions, there is a specific rule type: *t.context*, *t.goal* and *t.limit*. The 't' in these type names stands for 'tolerance'. In the following sub-sections we describe each one of all types in detail.

### 3.1 Tolerance Rule Type 1 (*t.context*)

In order to express in which contexts the failure of certain tasks may be ignored we use *t.context* rules. It has the following structure:

```
tasksSet isAllowedToFailIf contextExpression
```

*tasksSet* is a set of tasks divided by a colon (:), and that has at least one task - i.e., it can not be an empty set. The *allTasks* reserved word may be used to refer to all the tasks of the goal model, without needing to name them one by one. This definition of *tasksSet* and of the *allTasks* reserved word is shared with all the remaining types.

*isAllowedToFailIf* is a fixed string to identify the rule type. *contextExpression* is a logic expression, with the following structure:

```
ContextEntity.AttributeName operator
    AttributeValue
```

*contextEntity* is any entity of the system's context model, and *AttributeName* is the name of an attribute of that entity. *operator* is a logic

comparator, among the following: equals (=), greater than (>), greater equals than (>=), lower than (<), lower equals than (<=) and different (<>). `AttributeValue` is any possible value that entity attribute may have. During the system execution, this value will be compared with the actual value of that attribute, in order to evaluate if this context applies or not.

A rule of the `t.context` type has the following meaning: if a task that is an element of the `tasksSet` fails and the `contextExpression` currently applies, then that failure will be ignored. In other words, no compensation will be performed for that failure.

Usual situations in which a failure can be ignored are those related to date and time, as in examples 1 and 2. Considering the Smart Office System, that periodically download the e-mails and feeds for its user, Example 1 states that the failure of the `downloadEmail` task will be ignored if it happens before 8a.m., as well as the failure of the `downloadFeeds` task. Example 2 defines that the failure of any task of the system will be ignored if it occurs on a Sunday. Example 3 illustrates that any context entity of the context model can be part of the context expression - the failure of the `downloadEmail` task will be ignored whenever the user is away from his office.

- Ex.1:** *downloadEmail:downloadFeeds  
isAllowedToFailIf calendar.hour<=8*
- Ex.2:** *allTasks isAllowedToFailIf  
calendar.day=Sunday*
- Ex.3:** *downloadEmail isAllowedToFailIf  
user.isAtOffice=false*

### 3.2 Tolerance Rule Type 2 (t.goal)

This type of rule uses the goal model to define when a task failure will be ignored. The status of a goal, or a set of goals will be examined. Its structure is similar to the structure of `t.goal`:

```
tasksSet isAllowedToFailWhen goalExpressions
```

`tasksSet` is defined similarly to the `t.context`. `goalExpressions` is a non-empty set of logic expressions, separated by a colon (:). Each expression is an equality comparison: `goalName=satisfied` or `goalName=unsatisfied`. In the first case, the expression will apply if the given goal is currently satisfied, and in the second case if the given goal is currently unsatisfied. The rule will apply if and only if all its `goalExpressions` apply.

The string `isAllowedToFailWhen` is the identifier for this rule type. The `t.goal` rule states that whenever a

task in the `tasksSet` fails, if all expressions of `goalExpressions` apply then this failure will be ignored.

In Example 4 we have that the failure of the `downloadFeeds` task will not be compensated if the system did not help the user in executing his/her work. In Example 5 the failure of the `downloadEmail` task will be ignored if both the `assistWorkExecution` goal is satisfied and the goal `provideNews` is satisfied.

**Ex.4:** *downloadFeeds isAllowedToFailWhen  
assistWorkExecution=unsatisfied*

**Ex.5:** *downloadEmail isAllowedToFailWhen  
assistWorkExecution=satisfied:provideNews=s  
atisfied*

### 3.3 Tolerance Rule Type 3 (t.limit)

In this rule type we are not concerned in defining specific conditions in which a failure will be ignored. Instead, the concern is to define a maximum number of times that some task will fail without being compensated. This type has the following structure:

```
tasksSet isAllowedToFailAtMost limit
```

`tasksSet` is defined similarly to the `t.context` and `t.goal` cases. The `isAllowedToFailAtMost` name uniquely identifies this rule type. `limit` is a positive integer number that indicates how many times the failures of each task of the `tasksSet` will be ignored, before a compensation is required.

A rule of this type means that each task of the `tasksSet` will have a `limit` number of failures ignored. The failure number `limit + 1` will be compensated, and the failure counting of that task will be reset.

Note that we do not define a limit of failures for a set of tasks, but the limit for each task of the `tasksSet`. For instance, in Example 6 the limit of 5 failures is not for the two tasks altogether, it is for each task separately (`downloadEmail` and `downloadFeeds`). The rule of the Example 6 can be split in other two rules (examples 7 and 8), keeping the same meaning.

**Ex.6:** *downloadEmail:downloadFeeds  
isAllowedToFailAtMost 5*

**Ex.7:** *downloadEmail isAllowedToFailAtMost 5*

**Ex.8:** *downloadFeeds isAllowedToFailAtMost 5*

## 4. Policy Processing

The goal of the Tolerance Policy processing is to define all failures that will be ignored. For that, the

procedure described in Figure 2 is used. Initially, there is a list of failed elements - i.e., the tasks that were not successfully completed. There is also a list of tolerance rules, extracted from the policy file, and a list of context entities, from which we can get the current attribute values of that entities. The result of this procedure is a list of failed elements without those which failure will be ignored.

```

Data: FE : FailedElement [], TR : ToleranceRule [], CE : ContextEntity []
1 foreach  $fe_i$  in FE do
2   if  $\exists tr_1 \in TR (tr_1.type = t1 \text{ or } tr_1.type = t2) \text{ and } tr_1.elementsSet.contains(fe_i)$ 
3     then
4       foreach  $tr_j$  in TR do
5         if  $tr_j.type = t1$  then
6           if EvaluateContext( $tr_j.expression$ , CE) then
7             FE.removeFailedElement( $fe_i$ )
8              $fe_i.status \leftarrow ignored$ 
9           end
10          else
11            if  $tr_j.type = t2$  then
12              if EvaluateGoals( $tr_j.expression$ , FE) then
13                FE.removeFailedElement( $fe_i$ )
14                 $fe_i.status \leftarrow ignored$ 
15              end
16            end
17          end
18        end
19      end
20    end
21    if ( $fe_i.status \neq ignored$ ) then
22      if  $\exists tr_2 \in TR tr_2.type = t3 \text{ and } tr_2.elementsSet.contains(fe_i)$  then
23        if  $fe_i.failureCounter < tr_2.limit$  then
24          FE.removeFailedElement( $fe_i$ )
25           $fe_i.status \leftarrow ignored$ 
26           $fe_i.failureCounter \leftarrow fe_i.failureCounter + 1$ 
27        else
28           $fe_i.failureCounter \leftarrow 0$ 
29        end
30      end
31    end
32  end
33 return FE

```

**Figure 2 - Algorithm for failure ignoring evaluation**

For each failed element (line 1), we check if there is a rule of the type t.context (t1) or t.goal (t2) which `tasksSet` contains that element (line 2). If there is such a rule, we are going to analyze each one of these rules (lines 3 and 4). If the rule is of the type t.context (t1) and its context expression applies, we will remove this element from the list of failed elements and mark that element as *ignored* (lines 5 to 9). If the rule is of the type t.goal (t2) and its goal expressions apply, we also remove this element from the list of failed elements and mark that element as *ignored* (lines 10 to 17). The analysis of the context expressions and of the goal expressions are performed, respectively, by the procedures EvaluateContext and EvaluateGoals. After analyzing all t.context and t.goal rules for the element,

if it is not yet marked as *ignored* (line 21), we will check if there is a rule of the type t.limit (t3) which `tasksSet` contains that element (line 22). If there is such a rule, we will check if the failure limit for that element was reached (line 23). If the limit was not reached yet, we will increase the failure counter of that element and mark it as *ignored* (lines 24 to 26). If the limit was reached, we will not ignore that failure - i.e., the compensation will be required - but we will reset the failure counter (line 28). As a result we return the list of failed elements (line 33), from which we removed all elements which failures were supposed to be ignored.

The EvaluateContext and EvaluateGoals procedures simply check if the rules conditions apply [14]. These procedures will not be detailed here for the sake of space.

In summary, the t.context and t.goal rules define conditions when the failure of a given task may be ignored, and t.limit rules define an amount of failures of a given task that will be ignored. However, the amount of failures defined with a t.limit rule does not take into account the failures already ignored by the t.context and t.goal rules.

In this sense, we can state that the rule types t.context and t.goal prevails upon the type t.limit. Given a t.context rule, the failure of a task in its `tasksSet` will always be ignored if its context expression is satisfied, despite how many times this failure had been ignored before. In a similar way, given a t.goal rule, the failure of a task in its `tasksSet` will always be ignored if its goal expressions hold.

The t.limit rules are concerned only with the failures that were not ignored during the evaluation of the t.context and t.goal rules. Note that the failures ignored due to a t.context or a t.goal rule will not change the failures counting of a task.

Rules can interact. For example three rule types, from examples 9 (a t.context rule), 10 (a t.goal rule) and 11 (a t.limit rule) and the failure log depicted in Table 1. That table shows a log of failures of the *downloadEmail* task, together with the number of the failure, the value of the *calendar.day* attribute and the status of the *assistWorkExecution* goal at the moment of the failure. It also indicates if the failure was ignored as well as the rationale (the rule used for ignoring the failure).

**Ex.9:** *downloadEmail isAllowedToFailIf calendar.day=sunday*

**Ex.10:** *downloadEmail isAllowedToFailWhen assistWorkExecution=satisfied*

**Ex.11:** *downloadEmail isAllowedToFailAtMost 3*

In this example, the failures for which the rule of the example 9 applies were ignored: 2 and 3. In the same way, the failures 1, 3 and 6 were ignored due to the rule of the example 10. These rules do not apply for failures 4, 5, 7, 8 and 9, so we may evaluate the rule of the example 11 for these failures. The failures 4, 5 and 7 were ignored, since they were below the limit of 3 failures expressed in the rule. The failure 8, being the fourth failure of that task that were not ignored by a t.context or t.goal rule, shall be compensated, and the failure counter for that task shall be reset. Since the failure counter was reset, the failure 9 was also ignored for being below the limit of three failures.

## 5. Application

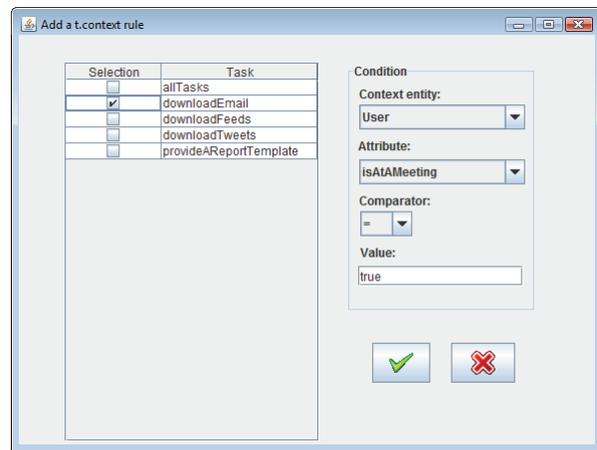
In order to use our approach we implemented all algorithms needed to evaluate the proposed policy. They were integrated with a simulator of our chosen self-configuration architecture [6]. We can provide a goal model, a context model and a log of context events, from which the simulator will run the Monitor - Diagnoses - Compensation cycle. We added to the simulator the ability to receive the Tolerance Policy as input as well.

We also developed wizards for making it easier to create the policy rules. Figure 3 shows an example of the creation of a t.context rule. The user selects tasks, which are extracted from the goal model, and then defines in which context that task can fail without compensation. As an example, the following rule is defined: *downloadEmail isAllowedToFailIf user.isAtAMeeting=true*. With these wizards we prevent some syntax errors that could otherwise occur.

We applied the policy rules in the example of a Smart Office system introduced in Section 2. In order

to avoid the cost of downloading e-mails through a mobile phone connection at times when the e-mails are unnecessary, we defined some tolerance rules, as follows.

Assuming that when the user is at a meeting he may not need to check his e-mails, we define the following t.context rule: *downloadEmail isAllowedToFailIf user.isAtAMeeting=true*. Assuming also that it is not required to have his e-mails updated when the system has already finished assisting the user in performing his work, we define this t.goal rule: *downloadEmail isAllowedToFailWhen assistWorkExecution=satisfied*. Finally, accepting that the e-mail downloading can fail at most three times consecutively, the following t.limit rule is stated: *downloadEmail isAllowedToFailAtMost 3*.



**Figure 3 - Screenshot of the wizard for creating a t.context rule**

We ran the simulator applying only one rule at a time, considering the scenario of one typical day of work, but on which the Ethernet connection was always down. The average result was a decrease of 46% on the number of required compensations. This result shows that, in some situations, the use of a tolerance policy can reduce the overall cost of using a

**Table 1 - Failures log of the task *downloadEmail***

# failure	calendar.day	assistWorkExecution	Ignore failure?	Rationale
1	Saturday	Satisfied	Yes	Ex. 10
2	Sunday	Not satisfied	Yes	Ex. 9
3	Sunday	Satisfied	Yes	Ex. 9, Ex. 10
4	Monday	Not satisfied	Yes	Ex. 11 (1 <sup>st</sup> failure)
5	Monday	Not satisfied	Yes	Ex. 11 (2 <sup>nd</sup> failure)
6	Monday	Satisfied	Yes	Ex. 10
7	Tuesday	Not satisfied	Yes	Ex. 11 (3 <sup>rd</sup> failure)
8	Tuesday	Not satisfied	No	
9	Tuesday	Not satisfied	Yes	Ex. 11 (1 <sup>st</sup> failure)

system, without a significant impact on the system behavior.

## 6. Related Work

In this work we applied the Tolerance Policy in connection with Dalpiaz architecture [6]. Despite the existence of a Tolerance Policy component in the original architecture, their work did not define a set of rule types, neither how they could be applied. Thus, in our work we have provided a more fine-grained control on the failure handling mechanism of that architecture, which results in a smaller amount of compensations to be performed during a system execution.

There are some other architectures for self-configuring, self-managing and autonomic systems [11][12][13]. However, for the best of our knowledge, none of them provide this level of failure control.

## 7. Conclusions

In this paper it was presented a tolerance policy that deals with failure occurrences. The objective was to increase the flexibility of failure handling in self-configuring systems, using tolerance policies based on requirements models. In particular we can express conditions in which a failure may be ignored. These conditions are related to requirements models – more specifically, a goal model and a context model. In order to make a proof of the concept, we defined algorithms and proposed a Policy Editor tool. This editor makes it easier for the user to create and maintain the rules of a policy. A simple example was used to explain our approach.

For the future, we plan to increase the expressiveness of the policy rules, allowing the usage of logic operators like AND, OR and XOR to create more complex conditions. Furthermore, we want to handle more complex rules, which can mix different types of a rule. We also need to apply these policies mechanisms in a real-world software system, analyzing the usefulness and the effectiveness of our approach. Lastly, we are interested in investigating how our policy could be used in different architectures, i.e. moving towards a more generic tolerance policy.

## 8. Acknowledgements

We are thankful to Fabiano Dalpiaz, Paolo Giorgini and John Mylopoulos, for inspiring this work and for their continuous feedback. This work was partially sponsored by FACEPE, CNPQ and CAPES.

## 9. References

- [1]Horn, P. Autonomic computing: IBM's Perspective on the State of Information Technology. [S.l.]: IBM, 2001.
- [2]Kephart, J. O.; Chess, D. M. The vision of autonomic computing. Computer, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 36, n. 1, p. 41-50, 2003. ISSN 0018-9162.
- [3]Müller, H. A.; O'Brien, L.; Klein, M.; Wood, B. Autonomic Computing. CMU/SEI-2006-TN-006. [S.l.], 2006.
- [4]Salehie, M.; Tahvildari, L. Autonomic computing: emerging trends and open problems. SIGSOFT Softw. Eng. Notes, ACM, New York, NY, USA, v. 30, n. 4, p. 1-7, Julho 2005. ISSN 0163-5948. Available in <http://dx.doi.org/10.1145/1082983.1083082>
- [5]Parekh, J.; Kaiser, G.; Gross, P.; Valetto, G. Retrofitting autonomic capabilities onto legacy systems. Cluster Computing, Kluwer Academic Publishers, Hingham, MA, USA, v. 9, n. 2, p. 141-159, 2006. ISSN 1386-7857.
- [6]Dalpiaz, F.; Giorgini, P.; Mylopoulos, J. An architecture for requirements-driven self-reconfiguration. In: ECK, P. van; GORDIJN, J.; WIERINGA, R. (Ed.). CAiSE. [S.l.]: Springer, 2009. (Lecture Notes in Computer Science, v. 5565), p. 246-260. ISBN 978-3-642-02143-5.
- [7]Giorgini, P.; Mylopoulos, J.; Perini, A.; Susi, A. The Tropos Metamodel and its Use. In: Informatica journal, 2005.
- [8]Damianou, N. Dulay, N.; Lupu, E.; Sloman, M.; Tonouchi, T. Tools for domain-based policy management of distributed systems. In: Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS 2002), pages 203–217, 2002.
- [9]Strassner, J.; Samudrala, S.; Cox, G.; Liu, Y.; Jiang, M.; Zhang, J.; Meer, S.; Foghl'u, M.; Donnelly, W. The design of a new context-aware policy model for autonomic networking. In ICAC '08: Proceedings of the 2008 International Conference on Autonomic Computing, pages 119–128, Washington, DC, USA, 2008. IEEE Computer Society.
- [10]Stone, G.; Lundy, B.; Xie, G. Network policy languages: A survey and a new approach. Technical report, Defense Technical Information Center OAI-PMH Repository, 2003.
- [11]Anthony, R.; Pelc, M.; Ward, P.; Hawthorne, J.; Pulnah, K. A run-time configurable software architecture for self-managing systems. Autonomic Computing, International Conference on, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 207-208, 2008.
- [12]Ouda, A.; Lutfiyya, H.; Bauer, M. Towards self-configuring policy-based management systems. In: POLICY '08: Proceedings of the 2008 IEEE Workshop on Policies for Distributed Systems and Networks. Washington, DC, USA: IEEE Computer Society, 2008. p. 215-218. ISBN 978-0-7695-3133-5.
- [13]Subramanian, L.; Katz, R. H. An architecture for building self-configurable systems. In: MobiHoc '00: Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing. Piscataway, NJ, USA: IEEE Press, 2000. p. 63-73. ISBN 0-7803-6534-8.
- [14]Pimentel, J.H.C. High Level Failure Treatment for Self-Configuring Systems: The FAST Approach (In Portuguese: Tratamento de Falhas de Alto-Nível para Sistemas Auto-Configuráveis: A abordagem FAST). MSc Dissertation. Federal University of Pernambuco, 2010.

# Ontology-Driven Enterprise Application Integration

G. Bucci, V. Sandrucci, E. Vicario

Dipartimento Sistemi e Informatica - Università di Firenze

{bucci, sandrucci, vicario}@dsi.unifi.it

## Abstract

*A major hurdle in Enterprise Application Integration (EAI) is the semantic heterogeneity of data and applications. Today's integration solutions focus mainly on the physical and syntactical aspect, providing little or no mechanism for semantic integration. No shared semantic concepts are explicitly used to define the semantics of different pieces of exchanged data, therefore the developer has to hard-code what to do with each data item from each application.*

*Ontologies are recognized as a most appropriate technology to overcome the limitations of current integration practices. Actually, they make possible the definition of a commonly agreed semantic model which can be reused and shared across applications.*

*We propose the use of an ontological model to capture both technical and semantic issues of a given integration project and to use that model to identify and validate integration components, as well as to validate the consistency of system configuration. Furthermore, since the ontological model can be accessed programmatically from conventional languages (e.g. Java), the contents of the messages exchanged among integration components are actually ontologies, guaranteeing the semantic consistency of exchanged data.*

## Keywords

*Ontologies, knowledge base, semantic integration, enterprise integration patterns.*

## 1 Introduction

The aim of Enterprise Application Integration (EAI) is the interconnection of multiple Enterprise Information Systems (EISs) to extend business processes throughout organizations.

Usually, EISs are made of heterogeneous, autonomous, and distributed systems [1], [2]. They are the result of a development occurred over time, in which individual application systems were developed independently, with little or no support for interoperability.

A consistent number of integration platforms, based on a variety of integration technologies (such as message brokers, Message Oriented Middlewares (MOMs), Web Services, etc.), are available of the shelf [3] [4], [3], [5], [6]. These solve the essential problem of information transport and point to point connectivity, but they do not address the problem of semantic heterogeneity. Therefore, the developer must know the meaning of the low-level data structures in order to implement semantically correct integration. As a result, mechanisms for interpreting and validating exchanged information are to be hard-coded [7], [8] and the correctness of the integration completely depends on the engineer's knowledge [1].

The advent of semantic web technologies like ontology languages (such as RDF, RDF/S and OWL) enables formal description of the semantics of the data structures exchanged between enterprise application systems, thus opening the way to automated validation and reconciliation of exchanged data. As such, semantics-based technologies are envisaged as an essential part of integration solutions in the near future [7] [8]. Frameworks like Protégé and Jena are already available to help the designer implement ontologies and use them via conventional programs.

An ontology-driven approach to integration, based on ontology mediation, is proposed in [2], where Web Services are chosen as the basic technology for integration. The resulting architecture is obtained augmenting SOA with a semantic layer that aims at enhancing service mediation in the context of EAI. Contrary to a static integration solution, where an integration scenario predefines every detail of any potentially connectable EIS, the approach supports also dynamic integration, in the sense that the binding services of target EISs can be performed at run time.

The potential use of ontologies and Semantic Web in systems and software engineering is the subject of a working draft of W3C [9].

The use of ontologies in a classical software engineering context is discussed in [10] and [11]. The goal of [10] was to show how ontologies can be defined that support the developer in creating new software or in running new components in a complex environment like an application server. Semantic metadata regarding software components

and APIs are described through an ontology. This may be queried for finding APIs with certain capabilities (development time support) or for pre-loading components that are required by other components (run time support). The ontology is also used to perform validity checks aimed at avoiding inconsistent system configurations.

An examination of several ontology-based approaches for data integration is in [12]. A thorough review of the potential benefits that software engineering can achieve by applying ontologies in the various stages of the software development life-cycle is [13].

The problem of developing software architectures, based on formal domain models (ontologies), consisting of conventional components written in languages such as Java, is treated in [14]. The point is the construction of the ontological domain model and the instantiation of the classes from the ontology into Java classes so as to be manipulated by conventional programs. This is made possible through the use of Protégé (see also [15] and [16]) and Jena library, the former as a rapid prototyping environment for building ontological domain models and to test how the system behaves in response to ontology changes, the latter as the API toward plain Java programs.

The transformation of an ontology in RDF Schema into a familiar, object-oriented Java API is described in [17]. The mapping of an OWL ontology into Java is also addressed in [18], where a set of Java interfaces and classes are created from an OWL ontology such that an instance of a Java class represents an instance of a single class of the ontology, with most of its properties, class relationships and restriction-definitions maintained. A framework that translates ontology constructs into Enterprise Java Beans and connects them to relational database persistence storage through the generation of Hibernate object-relational mappings is presented in [19].

In this paper we address the definition of an ontology capable of relating technical integration aspects to application-domain concepts, so as to augment integration components with explicit semantics.

Semantic integration is obtained through the resulting ontological knowledge-base, which is used: (a) during analysis and design stages to search for components with certain capabilities and/or to validate configurations; and (b) to provide the semantic domain model from which the programmer instantiates Java objects that are therefore consistent with their intended meaning.

Physical integration is obtained through a middleware built on top of a communication bus (e.g. JMS). The middleware is made of number of Enterprise Integration Components (EICs), designed to implement the functionalities required by the given integration target. EICs are modeled after Enterprise Integration Patterns (EIPs) [20], that is patterns of solutions to recurring integration problems.

As such, EICs are essentially based on asynchronous messaging and the resulting middleware leaves existing applications unchanged. Through ontologies and patterns, the exchange of consistent information between business functions occurs in a manner that appears to be seamless.

The proposed integration method has been applied in a pilot project carried out for the largest hospital of the Tuscan region, Italy.

The rest of the paper is organized in the following manner. Section 2 defines the structure of an ontology serving the purpose application integration. Section 3 discuss how EICs are represented in the ontology, while section 4 defines the integration architecture for the given integration problem. Sections 5 and 6 illustrate the construction of integration components. Conclusions are drawn in section 7.

## 2 Defining the ontology for integration

Before introducing the structure of our ontology, let us define the simple integration problem that will lead us in the sequel. The problem is as follows.

There are several applicative programs (called App1, App2, App3, ...) installed at distinct locations (Loc1, Loc2, Loc3, ..., respectively), all of which have their own database storing patients' medical data.

Assume App1 stores the following attributes for each patient: name, surname, date and place of birth, social security number, telephone number and home address, while App2 stores the same attributes as App1, with the exception of patient's home address which is replaced by patient's profession. Other applications, store the same sort of data with possible variations as above. Each application assigns a unique identifier to any patient in the local database, but there is no global identifier valid across all the databases.

The integration target is twofold:

- 1) to ensure consistency among the data bases when one application modifies some data in its local database;
- 2) to allow execution of distributed queries, that is queries involving several databases; that could be the case of a query generated by App3 for the data describing a given patient (this requires the union of data from App1 and App2).

We want to build an ontology which serves two purposes:

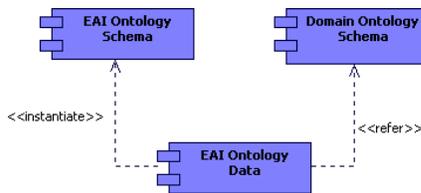
- to support "ontology-based specification" ([8], [10], [13]), aiding the integrator in: (a) finding software components and their properties; (b) validating consistency of component configuration; and (c) deriving further knowledge about the integration scenario;

- to support “common access to information” ([8], [10], [13]), allowing access the semantic model from conventional programs. This makes possible the construction of integration components which exchange ontologies rather than raw data, avoiding hard-coding of the mechanisms for data generation, interpretation and validation.

To satisfy the requirements of the previous two points, an ontology must be built around:

- concepts belonging to technical and architectural aspects, such as hardware component, software component, component allocation, etc.;
- concepts relative to the domain of interest such as patient, physician, therapy, etc.;
- individuals representing realizations of the concepts of the two previous points.

This leads to the structure of Figure 1 where EAI Ontology Schema is the ontological model pertaining to the technical aspects, Domain Ontology Schema is the ontological model pertaining to the domain of interest and EAI Ontology Data are the individuals instantiated from EAI Ontology Schema.



**Figure 1. General view of the ontological model supporting EAI.**

## 2.1 EAI Ontology Schema

The EAI Ontology Schema describes the concepts that are useful for integration. It is essentially composed of three classes, and their subclasses, representing respectively: (a) hardware components (i.e., machines); (b) software components; and (c) locations. Needless to say, the model also includes the associations among them.

As regards software, initially the integration scenario will only contain existing applications. In the course of integration, new sub-classes of class *Software*, are added. For instance, when a given type of EIC, say an *Adapter*, is identified, the corresponding class is added to the EAI Ontology Schema, while individuals of that class are instantiated in the EAI Ontology Data (see below) to represent the actual adapters entering in the integration solution. In this way the EAI Ontology Schema grows so as to describe any

software class, while EAI Ontology Data grows so as to describe any actual software component comprised in the integrated system.

## 2.2 Domain Ontology Schema

The Domain Ontology Schema contains domain concepts and domain predicates. It corresponds to the so-called “canonical data model” of [20], that is a data representation to/from which all the different data formats are converted. The essential difference with respect to [20] is that with the Domain Ontology Schema the developer obtains the payloads of exchanged messages by instantiating ontological classes in the Schema. This makes the system more resilient to the evolution of domain concepts. In fact, ontologies, provide a comprehensive and flexible conceptualization of domain data [21], which is inherently extensible and sharable across applications. This is the key concept to avoid hard-coding of data interpretation.

Referring to our integration target, the domain ontology must contain the class *Patient* and the following predicates: *hasName*, *hasSurname*, *hasPlaceOfBirth*, *hasDateOfBirth*, *hasSocSecN*, *hasTelephone*, *hasAddress*, *hasProfession* and *hasID*. These predicates correspond to the union of the sets of attributes managed by *App1* and *App2*. Furthermore, for each ID we must know the value of the local ID and the identity of the system to which that ID is related. This means that the rank of predicate *hasID* will be made of couples such as (*systemName*, *value*).

## 2.3 EAI Ontology Data

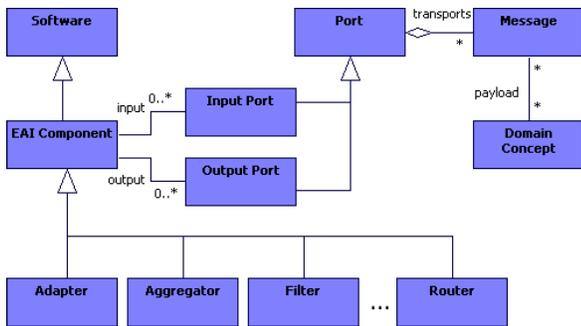
EAI Ontology Data contains individuals that are instantiated from EAI Ontology Schema. These individuals describe the actual (hardware and) software components; for instance, *App1*, *AdapterK*, etc.. Any instance is augmented with references to the domain concepts (e.g. *Patient*) classifying entities that are related to the component itself. In Figure 1, this is schematized through the dependency labeled <<refer>>.

EAI Ontology Data is populated in the course of integration activities. It is the duty of integrators and domain experts to establish the appropriate relations between the instantiated components and the concepts of the Domain Ontology Schema.

## 3 Integration Components

Each EICs is considered to be a black box with its own inputs and outputs. For each input (output) we specify the types of messages that are accepted (generated). Figure 2 shows that software components are related to exchanged

messages which, in turn, are associated to domain concepts (refer to the discussion of section 2.3). To avoid cluttering Figure 2, certain details have been omitted (for instance, communication channels associated with ports are not shown).



**Figure 2. Part of the model describing integration components as black boxes.**

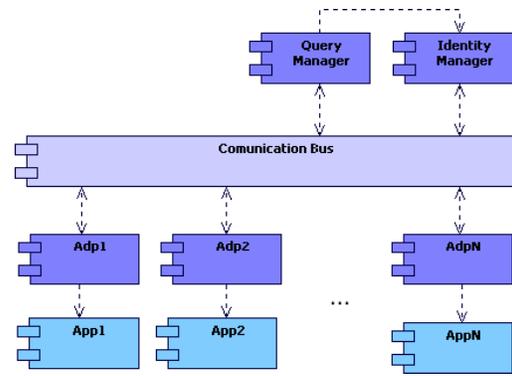
Integration components are defined adapting appropriate EIPs. The patterns collected in [20] permit to cope with almost any integration problem; therefore, the task of identifying the EICs to be put in place is greatly simplified.

The analyst can exploit the power of the knowledgebase. For instance, he can query the ontology to discover already defined components that suit specific needs, or he can infer the properties that a to-be-designed component must have in order to cooperate with the others.

#### 4 Defining the integration architecture

Referring to our integration target, we need the following components: (a) number of adapters between individual applications and the bus; (b) an identity manager to keep track of equivalences/replications of the patients described in the different databases and to maintain them aligned; and (c) a query manager to handle distributed queries. This leads to the architecture of Figure 3, where the depicted EICs have the following roles.

- *Adp1* translates the representation of *Patient* as of *App1* into the canonical representation and vice versa. *Adp2* does the same with respect to *App2*.
- *Identity Manager* maintains its own database to keep track of equivalences between the patients stored by different applications. It listens for the notification messages sent by adapters which inform that a *Patient* has been created, updated or deleted. On their arrival, the *Identity Manager* initiates a sequence of operations ending with a possible update to the database of equivalences. Along this sequence, the



**Figure 3. Identification of EICs over the application bus**

*Identity Manager* may query the adapters to obtain data relative to the *Patient*; it may also send commands to the adapters ordering them to update the representation of the patient.

In designing the *Identity Manager* we have been inspired by the concept of enterprise Master Patient Index (eMPI) as defined by IHE (Integrating the Healthcare Enterprise), an initiative designed to stimulate the integration of the information systems that support modern healthcare institutions [22].

- *Query Manager* is in charge of solving distributed queries, i.e. queries that involve the search of a given *Patient* in more than one database. Specifically, the *Query Manager*: (a) listens for a distributed query; (b) converts the distributed query into a query for the given *Patient*; (c) sends the query over the communication bus, so that all adapters can pass it to the associated data base; (d) collects the answers from adapters (i.e., from applications/databases); (e) uses the database maintained by the *Identity Manager* to eliminate duplications; (f) builds the body of the response to the distributed query; (g) sends the response over the communication bus.

Concerning point (e), the *Query Manager* and the *Identity Manager* interact on the basis of the integration pattern *Shared Database* [20]. The shared database is the database of equivalences maintained by the *Identity Manager*. (The dependency of the *Query Manager* on the *Identity Manager* is explicitly shown Figure 3.)

#### 5 Basic integration components

We distinguish between *basic EICs* and *composed EICs*. The former are those whose operations cannot be broken

into a number of simpler operations carried out by other components; the latter are those whose functions are obtained by appropriate composition of functions carried out by other components.

In this section we discuss the construction of a basic EIC (an adapter). In so doing we take the opportunity for explaining how ontologies help structuring and validating exchanged information. Next section will describe the construction of a composed EIC.

## 5.1 Component definition

In designing any component, the first step is the specification of its interface. For an *Adapter*, this is specified in terms of interactions with both the communication bus and the associated application. In Figure 4, the former are represented as solid arrows, the latter as a dashed (bidirectional) arrow. Specifically:

- `Patient Insert`, `Update` and `Delete Events` are notifications messages elicited by the local application. They respectively inform that something happened in the local database: (a) a new patient has been added; (b) a patient has been updated; and (c) a patient has been deleted. In our simple system, they are observed only by the *Identity Manager*, which initiates a sequence of operation that aligns the equivalence database maintained by the *Identity Manager* itself and all the databases containing data relative to the involved patient.
- `Patient Insert`, `Update` and `Delete Commands` determine execution of the corresponding operation against the data base managed by `App`. In the system of Figure 3 these messages arrive to the *Adapter* sent by the *Identity Manager*.
- `Patient Query Command` is converted to a query to the data base managed by `App`. In the system of Figure 3 this message is sent either by the *Identity Manager* or the *Query Manager*. The answer of the database is the used to build the `Patient Query Response`.
- Interactions with the associated application/data may take place in different ways, including: (a) use of services exposed by the application; (b) direct interaction with the local data base; and (c) use of conventional application interfaces.

An adapter is activated either by the associated application or by an appropriate message coming from the bus. The first case occurs when (a) the associated application needs to perform a query across the distributed system, or (b) the associated application has made a change to its local data that

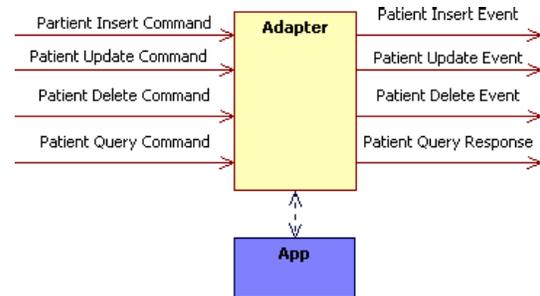


Figure 4. Specification of the basic component Adapter.

is to be communicated to the other application (change discovering is usually obtained via a trigger on the database). The second case is triggered by incoming command messages.

## 5.2 Ontology-supported component implementation

Consider the case in which the data relative to the representation of patient “Mario Rossi” is modified by `App1`. Then, `Adp1` will publish an appropriate notification. To set up the payload of the notification, the developer:

- uses the ontology to instantiate the patient. This is done with the support of library like Jena, by instantiating class `OntModel` and setting the due properties;
- converts the model to a transportable RDF string;
- pack the string in a notification message and sends it.

In coding a listener (e.g. the *Identity Manager*), the developer will:

- obtain the `OntModel` from the the RDF string constituting the payload of the received message. By construction the content of `OntModel` is consistent with its intended meaning. There is no need for hard coding specific validation mechanisms since they are automatically done by the supporting library;
- process the message content on the basis of on component’s specialization.

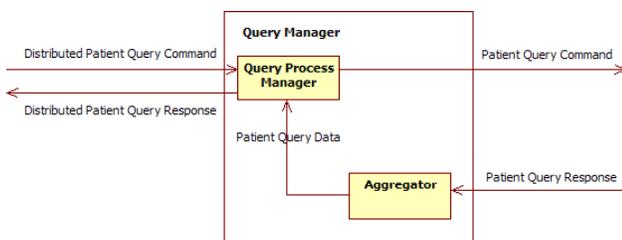
## 6 Composition of EICs

In the following, we refer to the *Query Manger* as an example of construction of a composed EIC.

The *Query Manager* recognizes a `Distributed Query Command` (generated by any application), interrogates the individual databases and recombines their answers

to build the response to the command. Of course, the *Query manager* could be programmed from scratch, though a better solution is to obtain it from the cooperation of simpler components. In Figure 5, the *Query Manager* results from cooperative work of two further basic components, i.e. the *Aggregator* and the *Query Process Manager*, both shaped as the EIPs [20] of the same name.

- *Aggregator* produces a message containing a list of patients obtained through aggregation of input messages.
- *Query Process Manager* is the adaption to our needs of the integration pattern *Process Manager* of [20]. This EIC has an internal state to keep track of the advancement of processing.



**Figure 5. How the *Query Manager* is obtained from simpler components.**

In building the response to a given distributed query, the *Query Manager* must use only the individual database responses that are relative to that query, avoiding to mix up responses that relative to other queries. This implies that messages must have a *correlation ID* so as to determine which message goes with which. The *Query Manager* behaves as follows:

- 1) recognizes a *Distributed Patient Query Command*; the message could say “send me the data relative to Mario Rossi”;
- 2) transforms the received message into a *Patient Query Command*. This message will be observed by any adapter which will obtain the data relative to the identified patient (i.e., Mario Rossi) from its local store, so as to answer with a *Patient Query Response*;
- 3) takes the *Patient Query Data* which are built by the *Aggregator* through composition of the *Patient Query Responses* coming from application adapters in response to the *Patient Query Command*;
- 4) analyzes the *Patient Query Data* eliminating possible duplicates so as to build the *Distributed*

*Patient Query Response* (this will be observed by the adapter of the application that issued the distributed query).

We must remark that the *Query Manager* of Figure 5 is a logical component rather than a real one, in the sense that its functionalities result from proper composition of the functionalities of the *Query Process Manager* and the *Aggregator*, which are the actual, concrete components. In other words, the *Query Manager* is obtained without writing a single line of code, the only requirement being that the installed individuals are properly configured so as to operate in concert.

Of course the process of composition can be iterated to build further, more complex components. For instance, we may want a component capable of returning all and only the patients that satisfy a given condition. This can be done combining the *Query Manager* with additional components capable of splitting and filtering the responses to *Patient Query Command* coming from individual applications.

In the stage of component definition, the ontological knowledge base is used to identify the existence of already defined components with certain capabilities. A reasoner can be used to validate the consistency of a given composition. For instance, we can verify whether messages *Patient Query Data* that the *Query Process Manager* accepts in input are consistent with those generated by the *Aggregator*. Before deployment, we can also validate whether each individual component has been properly configured, as well as validate overall configuration consistency.

## 7 Conclusions

We defined an ontological knowledge-base to describe both the technical and the semantic aspects of integration. Technical aspects are captured by an ontological model (EAI Ontology Schema) describing the overall hardware/software system. This includes the description of both applications to be integrated and integration components, as well as relations and dependencies among them. Starting from the initial representation of the application systems to be integrated, the ontology grows so as to include any EIC that is defined and validated.

Semantic aspects are captured by an ontological model (Domain Ontology Schema) describing concepts, relations and dependencies relative to the specific application domain. Instances of entities of the EAI Ontology Schema are associated with concepts of the Domain Ontology Schema giving rise to an ontology that relates software components to the domain concepts they deal with.

The developer uses the knowledge-base to examine the properties of the applications that are involved in a given integration target. This is followed by the definition of the

integration components that are required to achieve the specific target. In this process the developer resorts to Enterprise Integration Patterns, i.e. proved, structured solutions to almost any integration problem. In laying down the integration solution, he can consult the knowledge-base in search for the already defined integration components which possess (either partially or completely) the functionalities that are to be implemented. This is particularly useful in order to obtain new functionalities by aggregating existing integration components (either concrete or logical), so as to give rise to additional behaviors which correspond to new (logical) components. The same ontological model is the tool through which the semantics of EIC compositions, EIC configurations as well as system configurations is validated.

In programming EICs, the developer uses ontologies to derive the information contained in transmitted messages. As a result, the payloads of exchanged messages are actually ontologies, so that the receiving components can automatically rebuild information from those ontologies. This also guarantees that information circulating among integration components is always consistent with its intended meaning, avoiding a typical problem of today's enterprise integration platforms, that is the proliferation of configuration files.

## Acknowledgements

We are grateful to A. Tarocchi who helped defining and implementing the integration software. We are also grateful to C. Berdondini and L. Bartoli for their endorsement of the project within the Ospedale di Careggi, Florence, Italy.

## References

- [1] C. Bussler, "The role of semantic web technology in enterprise application integration," *Bulletin of the IEEE Comp. Soc. Tech. Committee on Data Engineering*, pp. 62–68, 2003.
- [2] S. Izza, L. Vincent, and P. Burlat, "A unified framework for application integration - an ontology-driven service-oriented approach," in *Proc. of ICEIS*, (Miami, Florida - USA), pp. 165–170, 2005.
- [3] M. Themistocleous and Z. Irani, "Towards a novel framework for the assessment of enterprise application integration packages," in *Proc. of HICSS'03*, (Hawaii, Usa), 2003.
- [4] T. Puschmann and R. Alt, "Enterprise application integration - the case of the robert bosch group," in *Proc. of HICSS'01*, (Washington, DC, USA), IEEE Computer Society, 2001.
- [5] K. Khoubati, M. Themistocleous, and Z. Irani, "Integration technology adoption in healthcare organisations: A case for enterprise application integration," in *Proc of HICSS'05*, (Washington, DC, USA), IEEE Comp. Soc., 2005.
- [6] R. Silveira and J. A. Pastor, "A model for enterprise application integration tools evaluation," in *Proc. of EMCIS 2006*, (Costa Blanca, Alicante, Spain), July 2006.
- [7] J. T. Pollock, "Integration's dirty little secret: It's a matter of semantics," whitepaper, Modulant, the Interoperability Company, February 2002.
- [8] M. Uschold and M. Gruninger, "Ontologies and semantics for seamless connectivity," *Sigmod Record*, vol. 33, pp. 58–64, December 2004.
- [9] P. Tetlow, J. Z. Pan, D. Oberle, E. Wallace, M. Uschold, and E. Kendall, "Ontology driven architectures and potential uses of the semantic web in systems and software engineering," tech. rep., W3C.
- [10] D. Oberle, A. Eberhart, S. Staab, and R. Volz, "Developing and managing software components in an ontology-based application server," in *Proc. of Middleware'04*, (New York, NY, USA), pp. 459–477, Springer-Verlag, 2004.
- [11] D. Oberle, S. Staab, and A. Eberhart, "Towards semantic middleware for web application development," *IEEE Distributed Systems Online*, 2008.
- [12] H. Wache, T. Voegelé, U. Visser, H. Stuckenschmidt, G. Schusterand, H. Neumann, and S. Hubner, "Ontology-based integration of information - a survey of existing approaches," in *Proc. of IJCAI-01*, (Seattle, WA, USA), pp. 108–117, 2001.
- [13] H. J. Happel and S. Seedorf, "Applications of ontologies in software engineering," in *Proc. of ISWC 2006*, (Athens, Georgia - USA), November 2006.
- [14] H. Knublauch, "Ontology-driven software development in the context of the semantic web: An example scenario with protege/owl," in *Proc. of MDSW2004*, 2004.
- [15] H. Knublauch, "An ai tool for the real world: Knowledge modeling with protege," *JavaWorld*, June 20, 2003., June 2003.
- [16] H. Knublauch, M. Horridge, M. Musen, A. Rector, R. Stevens, N. Drummond, P. Lord, N. F. Noy, J. Seidenberg, and H. Wang, "The protg owl experience," in *Workshop on OWL: Experiences and Directions*, (Galway, Ireland), 2005.
- [17] M. Voelkel and Y. Sure, "Rdfreacator - from ontologies to programmatic data access," in *Jena User Conference*, (Bristol, UK), May 2006.
- [18] A. Kalyanpur, D. J. Pastor, S. Battle, and J. Padget, "Automatic mapping of owl ontologies into java," in *Proc. of SEKE 2004*, (Banff, Canada), June 2004.
- [19] I. N. Athanasiadis, F. Villa, and A.-E. Rizzoli, "Ontologies, javabeans and relational databases for enabling semantic programming," in *Proc. of COMPSAC '07*, (Washington, DC, USA), pp. 341–346, IEEE Computer Society, 2007.
- [20] G. Hohpe and B. Woolf, *Enterprise Integration Patterns*. Addison Wesley, 2008.
- [21] G. Bucci, V. Sandrucci, and E. Vicario, "Potenzialità del paradigma ontologico nello sviluppo di applicazioni di e-government," *Informatica e Diritto*, no. 1-2, pp. 279–287, 2008. (in Italian).
- [22] IHE International, *IHE IT Infrastructure (ITI) Technical Framework, vol.1 Integration Profiles*, December 2008.

# Introduction of Time and Timing Variability in Usage Model based Testing

Sebastian Siegl  
University Erlangen-Nuremberg  
Martensstr. 3  
91058 Erlangen, Germany  
Email: sebastian.siegl@cs.fau.de

Reinhard German  
University Erlangen-Nuremberg  
Martensstr. 3  
91058 Erlangen, Germany  
Email: german@cs.fau.de

Kai-Steffen Hielscher  
University Erlangen-Nuremberg  
Martensstr. 3  
91058 Erlangen, Germany  
Email: hielscher@cs.fau.de@cs.fau.de

**Abstract**—In model-based testing Markov chain usage models (MCUM) are adopted to represent the possible usage of the system under test (SUT). The variability of operations of different users is represented by probabilities on transitions that have the stimuli assigned. Computations can be made that provide information about the model and the test cases that can be derived from it. However, all these figures give information about the elements of the abstraction, e.g. how they relate to each other, and not of the functionality that is described. The timing of usage and also variability in timing of usage is neither integrated nor considered systematically in the model, computations, and test case generation. We present in this paper enhanced usage models that are based on semi-Markov processes to integrate timing systematically in the model. In this way, it is possible to reflect time and the variance in timing in the usage model that serves as a basis for the whole method in testing of functional and non-functional requirements. Furthermore, we show how to compute the average test case duration and the testing time until a state, arc, or stimulus is covered in test cases, even before test case generation. In this way, it is possible to validate the suitability of the usage profile and to assess the testing effort before test case generation.

## I. INTRODUCTION

In many application domains, e.g. automotive and avionic systems, software becomes increasingly distributed and complex [1]. This development is intensified by the demand for environment friendly systems like hybrid and electric vehicles, in which software is necessary to provide functionality that was not necessary in previous systems or realized mechanically. In model-based testing Markov chain usage models (MCUM) are an established way to describe the possible usage of the system under test (SUT) and to systematically derive test cases from it. It serves as a basis for the whole engineering process to verify and validate the correct functionality of the SUT. A problem is that the timing in usage and the variability in timing of usage is not reflected by MCUMs. However, with the development of increasingly complex systems timing and variability in timing gains heavily of importance, because:

- system reactions depends and varies dependent on the timing of inputs applied.
- exhaustive testing is impossible and testing time is scarce in industry. As a consequence of this the test time and test benches should be used as efficiently as possible.
- test planning should be supported in advance by figures

that provide information about the test subject, e.g. the effort in time needed to meet a coverage criterion.

- test case generation can be optimized if it considers timing in usage to meet execution constraints.

This paper is structured as follows. In the next section related work is presented and Markov chain usage models are introduced. The characteristics and requirements of the testing routine in the automotive industry are presented. Next, our enhanced model, the *Timed Usage Model (TUM)*, is introduced. It is presented how computations and test management information can be derived. Algorithms to generate test cases from TUMs are presented. We present the results of a case study that we have conducted on a safety functionality in modern automobiles. Before the conclusion we give information about our future work.

## II. TESTING AND MCUMS

### A. Related Work

There has been a broad spectrum of research and practical experience [2] in the appliance of MCUMs in model-based testing. The MCUM as a stochastic source can be analyzed and provides help in the decision when to stop testing [3]. It is known how to generate transition probabilities and to guide the testing by usage profiles [4]. There exist proven computations and standard notations for usage models [5]. In the field of the modeling and performance analysis of systems with non-exponential timings semi-Markov processes are well understood and profound research and theory exists [6].

Yet in the field of usage modeling and test case generation from usage models no research is known to the authors that deals with the question how non-exponential timing and variability in timing can be incorporated systematically into usage models, computations, and test case generation.

We present Timed Usage Models on the basis of semi-Markov processes that address these issues and make it possible to describe the usage of a system in a more realistic way. In this way, the whole testing process and engineering can be supported by this additional information at hand.

### B. MCUM

A MCUM is usually described by its graphical abstraction that consists of:

- A set of *states*  $S = \{s_1, \dots, s_n\}$ , that represent possible states of usage.
- A set of *arcs*  $A$ , representing state transitions. An arc from state  $s_i$  to state  $s_j$  is denoted by  $a_{ij}$ , multiple arcs between  $s_i$  and  $s_j$  are not allowed.
- A set of *stimuli*  $Y$  on the SUT. A stimulus  $y_j$  is assigned to each arc.
- The *transition probability* from state  $i$  to state  $j$ , denoted by  $p_{ij}$  for an existing arc  $a_{ij}$ . Otherwise the transition probability  $p_{ij} = 0$ .

In Figure 1 an example of a MCUM as a directed graph is presented.

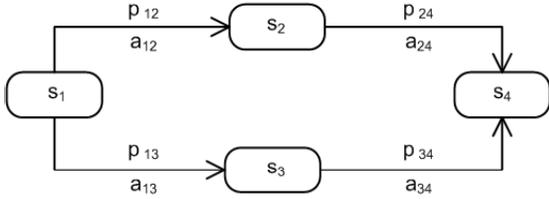


Fig. 1. Classic Markov Chain Usage Model

The values  $p_{ij}$  can be stored in a *state-transition-matrix*  $P$ , that is called a *usage profile*. Element describes  $p_{ij}$  the probability of the transition from state  $s_i$  to state  $s_j$ .

All paths from the start to the final state are a valid test cases.

The traversal of such a model is guided by two stochastic processes. The first, also referred to as the *Embedded Markov Chain (EMC)*, determines the change of states according to  $p_{ij}$ , the second the sojourn time in states with  $F_i(t)$ .

### C. Testing in automotive domain

Testing and test constraints in the automotive domain comprise:

- test subject. Usually the SUT specification forms the basis, from which the testing requirements are derived.
- test process and automation. It determines the steps and how they have to be taken, either manually or automated.
- test bench, that directly determines under which constraints the SUT can be tested.

In the following, test subjects characteristics, the test automation and process *Extended Automation Method (EXAM)*, applied within the Volkswagen AG, and the test benches employed in the automotive domain are briefly presented.

a) *Test Subject*: Functionality in the automotive domain is increasingly based on distributed real-time embedded systems by means of software. Due to this development the complexity and time dependence of functionality increases as well as the potential usage scenarios.

b) *EXAM Testing Process*: Test automation in the scope of EXAM means the automated generation of platform dependent code and the execution of the derived test suite without human interaction. The EXAM testing methodology [7], as it is used by the Volkswagen AG, defines the process, roles, and tools used to

- 1) model test cases graphically and platform independently in UML.
- 2) generate platform dependent test scripts automatically from the formal description in UML.
- 3) automatically execute the test cases and evaluate the results.

The majority of test cases is automatically executed on Hardware-in-the-loop (HIL) systems.

c) *HIL testing*: HIL-testing has become the de facto standard in the automotive industry. Hardware-in-the-loop (HIL) simulation provides an environment for testing embedded systems composed of Electronic Control Units (ECUs) under conditions so that the ECUs *feel* like being in a real car.

HIL systems have high acquisition costs. Due to this car manufacturers do not dispose of a large number of HIL systems and HIL systems are being shared by different departments. So testing time on HIL systems is scarce.

Summing up the testing routine in automotive industry one can say that automated solutions to execute and evaluate test cases exist. Yet testing time is scarce and a method is needed for the systematic and automated generation of test cases. This method should incorporate knowledge of the timing behavior of the SUT itself and allow test case generation that takes into account test time constraints.

## III. TIMED USAGE MODEL

In this section, our extension to classic MCUMs is presented.

### A. Semi-Markov Extension

A semi-Markov process (SMP) is a generalization of the Markov process as it allows a random amount of time between changes of states. The next state is still chosen dependent on  $p_{ij}$ , as it is in the Markov chain. But the time until the transition from state  $s_i$  to state  $s_j$  occurs, i.e. the time  $X_i$  until state  $s_i$  is left is determined with the distribution  $F_i(t)$ . In Figure 2, the elements of a semi-Markov process are presented in a directed graph structure.

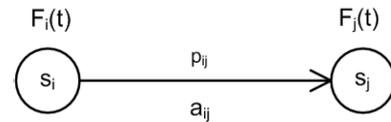


Fig. 2. Semi-Markov Process

### B. Modeling elements

The new attributes that allow the realistic integration of time and timing variability in the model are as follows:

- A probability density function (pdf)  $t_i$  to reflect the sojourn time is assigned to each state  $s_i$ .
- A pdf of the stimulus time  $t_{ij}$  is assigned to each arc  $a_{ij}$ . This pdf describes the duration of the execution of a stimulus on the SUT. The concept provides the possibility to characterize the stimuli by its typical variation in time,

that can be fix or variable and vary from very small to large values.

The classic transition probabilities  $p_{ij}$  as well as the values of the new attributes  $t_i$  and  $t_{ij}$  can be stored and exchanged by means of a matrix  $P$ . This extension provides the possibility to consider time aspects where needed. The extensions introduced are highlighted in Figure 3.

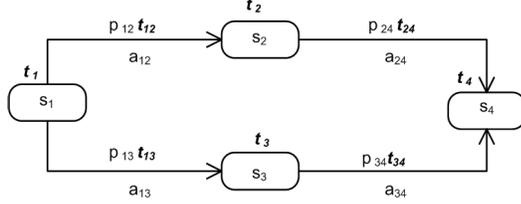


Fig. 3. Timed Usage Model

### C. Mapping to semi-Markov Processes

The modeling elements have to be mapped on a semi-Markov process so that computations and analysis is possible. The mapping comprises the following steps:

- Each transition  $a_{ij}$  is mapped to a pseudo state  $s'_{ij}$ .
- For each new pseudo state  $s'_{ij}$  two transitions  $a_{iij}$  and  $a_{ijj}$  are introduced. The first leads from the originating state  $s_i$  to  $s'_{ij}$  and has the probability  $p_{ij}$  assigned. The second transition leads from  $s'_{ij}$  to  $s_j$ . The probability of  $p_{ijj}$  is one.
- The sojourn time  $X_{ij}$  in the pseudo state  $s'_{ij}$  is sampled from  $F_{ij}(t)$ . In the TUM  $F_{ij}(t)$  is denoted by  $t_{ij}$ .
- The sojourn time  $X_i$  in states  $s_i$  is described by  $F_i(t) := t_i$ .

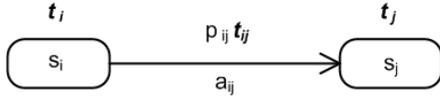


Fig. 4. Timed Usage Model Elements

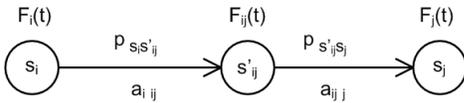


Fig. 5. Semi-Markov Process of Timed Usage Model

### D. Computations

1) *Classic Markov Computations:* Markov computations are used to analyze the model. Classic Markov computations comprise [5]:

- Number of occurrences of a state, arc, or stimulus in a test case
- Long-run state, arc, or stimulus probabilities
- Probability of occurrence for states, arcs, or stimuli
- Expected test case length

Computations give figures about the elements of the graph abstraction and how the elements relate to each other. For example the measure *average test case length* is a number in terms of states and transitions. Clearly this figure depends strongly on how the designer decided to build the model and less on the testing subject. Average test case length in time, i.e. the average test case duration, could provide a more valuable measure for test management and the determination of test cases to be generated.

### E. Timed Usage Model Computations

The classic Markov computations stated in section III-D can be applied to the EMC of the new model. Moreover, we present indicators that can be computed making use of the timing information.

The following steps form the basis for our computations.

To simplify the notation of the computations we renumber the elements. All composite indices are mapped to single ones. The states are enumerated by  $s_i \in S$  and  $i$  ranges from 1 to  $m$ , at which  $m = |S'|$  and the index  $m$  indicates the final state of the Timed Usage Model. The indices of the distributions  $F'_{ij}(t)$  are mapped to single indices in accordance with the state indices. The new matrix  $P'$  is made up according to the new state indices.

The variables  $c_{s_i}$  are introduced, that store the expectation value  $E[F_i(t)]$  for each state  $s_i$ . The values  $c_{s_i}$  are stored in the matrix  $C$ . The entries are defined as:

$$c_{ij} = \begin{cases} E[F_i(t)] & i = j \\ 0 & i \neq j \end{cases} \quad (1)$$

a) *Expected number of occurrences of a state and transition:* The EMC is used to compute the expected number of occurrences of a state in a test case. The transition probabilities of the EMC are given by  $P'$ . Let  $n_{ij}$  denote the number of occurrences of state  $s_j$ , given that one starts in  $s_i$ . For the expectation value for the number of occurrences  $E[n_{ij}]$  the following relation holds [5]:

$$E[n_{ij}] = \delta_{ij} + \sum_{k \in S_0} p_{ik} \cdot E[n_{kj}] \quad (2)$$

whereas  $\delta_{ij}$  is the Kronecker delta:

$$\delta_{ij} = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases} \quad (3)$$

Writing  $E_{i,j}$  in matrix notation as  $N = [E[n_{ij}]]$  allows us to formulate equation 2 in matrix form:

$$N = I + P'N \quad (4)$$

$$N - P'N = I \quad (5)$$

$$(I - P')N = I \quad (6)$$

$$N = (I - P')^{-1} \quad (7)$$

For the next steps only the first row of  $N$ , i.e.  $E[n_{1j}]$ , is of interest, as we always start in the initial state  $s_1$ . So  $E[n_{1j}]$  provides us the expected number of each state of the EMC in a test case. As we have mapped all transitions  $a_{ij}$  of the

original model to states we can directly obtain the expected number of occurrences of a state as well as of a transition in a test case.

b) *Expected residence time of a state and transition:* The matrix  $C$ , which provides the expectation value of  $E[F_i(t)]$  for each state  $s_i$ , is needed to compute the expected residence time of a state or transition. The expected residence time of each state and transition  $E_r[s_i]$  in a test case can be computed straightforward as follows:

$$E_r[s_i] = E[n_{1i}] \cdot C_{ii} \quad (8)$$

c) *Expected execution time of stimulus:* To compute the expected execution time of a stimulus  $E_r[y_j]$ , one has to take into account that different pdfs can be assigned at each occurrence of the stimulus. Therefore  $E_r[y_j]$  can be computed by:

$$E_r[y_j] = \sum_{j \in A_j} E[n_{1j}] \cdot C_{jj} \quad (9)$$

d) *Average test case duration:* The presented solutions are based on the concepts for system performance analysis of communication systems [6]. First, the steady state distribution, i.e. the long-run state probabilities of the states of the Embedded Markov Chain (EMC), has to be computed. For this it is necessary that the EMC is strongly connected, i.e., each state  $s_i$  can be reached from any other state  $s_j$  in a finite number of state transitions. We achieve this by adding a recurrence loop from the final state  $s_m$  to the source state  $s_1$  with probability one and  $F_m(t) = Det(1)$ . In the matrix  $P'$  this is achieved with the entry  $p_{m1} = 1$ . Hence  $P'$  is an irreducible matrix.

Let  $\Psi = [\psi_{s_1}, \dots, \psi_{s_m}]$  be the vector for the long-run probabilities of the  $m$  states. This vector can be computed by solving the eigenvector equation:

$$\Psi \cdot P' = \Psi, \quad \Psi \cdot e = 1 \quad (10)$$

The steady state distribution of the SMP can be computed by:

$$\Pi = \frac{\Psi \cdot C}{\Psi \cdot C \cdot e} \quad (11)$$

The probability of being in state  $s_j$  at a point in time  $t$  is denoted by  $\Psi_j(t)$ . At time zero the distribution  $\Psi_j(0)$  is one for the start state  $s_1 = 1$ , all other states have probability zero.

With our next step, we switch from stationary to transient analysis. So in our next computations the recurrent loop from state  $s_m$  to  $s_1$  is removed. The set of transient states is then  $S_0 = \{s_1, \dots, s_{m-1}\}$  and the final absorbing state  $s_m$  constitutes the set of absorbing states  $S_1$ . In Figure 6, the set of transient and absorbing states are illustrated.

The mean time to absorption  $\sigma_i$  in each state  $s_i$  in the EMC can be computed with the help of:

$$\sigma_j - \psi_j(0) = \sum_{i=1}^{m-1} \sigma_i \cdot p'_{ij} \quad , \quad \forall j = 0, \dots, m-1 \quad (12)$$

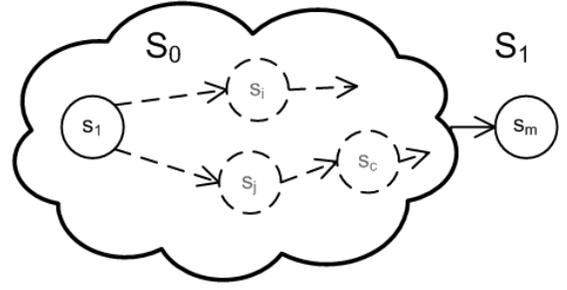


Fig. 6. Transient and absorbing states

For the SMP, the mean time to absorption can be computed by the definition of

$$\chi_i = \sigma_i \cdot c_{s_i} \quad (13)$$

Then we obtain:

$$\chi_j - \psi_j(0) = \sum_{i=1}^{m-1} \chi_i \cdot p'_{ij} \quad , \quad \forall j = 0, \dots, m-1 \quad (14)$$

In this way, we can compute the expected test case duration  $E[d]$  by summing up the mean time to absorption for all states before the final state:

$$E[d] = \sum_{j=1}^{m-1} \chi_j \quad (15)$$

e) *Testing time until a state or transition appears in a test case:* Next, a solution is presented how the time can be computed until a state, a transition or a stimulus is covered or invoked, respectively. We present the solution how to compute the expected testing time using the example of a state. Clearly this solution can easily be adapted to a transition, by considering the state that corresponds to a transition. The computations to determine the expected test case duration (cf. III-E0d) can be used straightforward if we change the structure of the Timed Usage Model. First, the model is made recurrent again by introducing the loop from the final state  $s_m$  to the initial state  $s_1$ . Then the probability of all outgoing transitions of the state under consideration  $s_c$  is set to zero. In this way, the state under consideration constitutes the new set of absorbing states  $S_1 = \{s_c\}$ . The set of transient states is then  $S_0 = \{s_i\}, \forall i \neq c$ .

Having changed the model in this way, the computations introduced in section III-E to compute the expected test case duration can be applied straightforward to determine the expected testing time until a state or transition is covered.

### F. Test Case Generation

The algorithms for the generation of test cases from classic MCUMs can still be used for the Timed Usage Model on the basis of the EMC, however, in this way the time information is not used. An important information for the test management is the execution time of each test case and of a set of test cases. It can now be easily derived by adding up the sampled

durations during test case generation. The summation of the estimated time is not explicitly stated in each algorithm. Moreover, the timing information should be considered in test case generation, as it can be used on the one hand to improve the testing of non-functional requirements. In the presented algorithm, we obtain the values  $p_{ij}, t_i, t_{ij}$  from the usage profile  $P$  that is assigned to the model (cf. section III).

f) *Random Walk*: Random walk is the standard sampling technique to derive test cases from MCUMs. At each state the next transition is chosen randomly based on the probabilities from the assigned usage profile  $P$ . Different usage profiles can be applied to guide the test case generation by random walk.

Random walk for TUMs extends this algorithm, as not only the next transition is chosen randomly, but also the duration of a state and the execution of a stimuli in a test case is sampled from the usage profile. In this way, the generated test cases vary in the stimuli themselves and in the time between and of the execution of stimuli. The random walk for TUM is specified in Figure 1.

**Data:**

Timed Usage Model;  
 $s_c$  state under consideration;

**Initialization:**

$s_c \leftarrow s_1;$

**while**  $s_m$  not reached **do**

    Get  $X_c$  by sampling  $t_c$ ;  
    Get  $s_j$  by sampling over  $\forall p_{cj}$ ;  
    Get  $X_{cj}$  by sampling  $t_{cj}$ ;  
     $s_c \leftarrow s_j$ ;

**end**

**Algorithm 1:** Random walk for Timed Usage Models

g) *Timed Usage Model of Electronic Stability Program*:

We evaluated our approach with a case study from the automotive domain. We conducted it within the research project with AUDI AG. We chose the operational concept of the Electronic Stability Program (ESP) in modern cars. Different functionalities are invoked dependent on the state of the car and the pressing duration and frequency of the ESP-switch [8]. Some of these functionalities directly represent the desired reaction by the user and some are assumed to represent the reaction desired by the user. One of these is the so called handbag switching mechanism. In some cars, the ESP switch is placed in the center console, where car passengers often deposit their hand bag. This can lead to frequent misactivations of this switch and as a consequence to undesired deactivation and activation of the ESP. On account of this, the handbag switching mechanism is now integrated in the cars. This mechanism activates ESP irrevocably for the trip if the ESP Switch is pressed for longer than 15 seconds, thus preventing the undesired deactivation of ESP if an object is placed on top of the ESP switch.

In figure 7 one can see different equivalence classes of

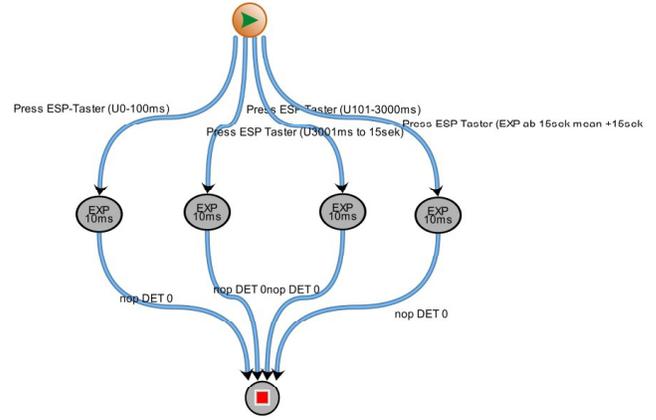


Fig. 7. Submodel ESP Switch Activation

activation durations of the ESP switch. The presented models are reused at different levels of the whole usage model.

In this way, we described possible usage scenarios concerning the ESP switch through all drive scenes systematically.

h) *Computations*: Computations have been made for the model both in the classic way (cf. section III-D) and new computations for TUM as introduced in this paper. Results of classic computations are presented in table I. The states  $S1$  and  $S9999$  have the same values for stationary distribution 0.03844 and the expected number of visits is in both cases 1.0000. State  $S4$  has a value of the stationary distribution of 0.07686 and an expected number of visits of 2.0000. State  $S234$  has a comparatively low value of the stationary distribution 0.01443 and expected number of visits 0.37538. Taking these figures as indicators for the test case generation, one would not attach to state  $S234$  as much importance as to state  $S4$ , regarding their expected number of visits. So if one would like to have a balanced occurrence of states in test cases, one would adjust the usage profile so that states  $S1$  and  $S234$  get lower indicators for the expected number of visits and the expected number of visits for state  $S234$  is higher. In this way, one would also expect that the testing effort is equally distributed among the states. Yet this procedure does not take into consideration the time consumed in testing by each state as well as the overall testing time that would result from this figures. Furthermore, if one considers testing of non-functional requirements, the equal distribution among all states is only useful if each state can be associated with one non-functional requirement.

Results of new computations are presented in table I. Figures from the classic computations are included for direct comparison. The new computations take the time aspects into consideration and give valuable information to decide if the usage profile is suitable for test case generation. One can see that, although the stationary distribution and expected number of visits of state  $S234$  is comparatively low (0.01443 and 0.37538), the expected residence time in a test case is 3.75380. So taking testing time into consideration this state

has to be given more attention that e.g. state  $S_{9998}$ , that has a higher expected number of visits (1.00000) but zero expected residence time. Furthermore, the new computations show, that states  $S_1$  and  $S_{9999}$ , though having the classic indicators for expected number of visits and stationary distribution in common, should discriminated for test case generation. The expected residence time of state  $S_1$  in a test case is 9.00000, whereas of state  $S_{9999}$  it is 0.00000. So test case  $S_{9999}$  is obviously a state in which none or no time consuming action happens, so from testing time aspects it can be neglected. This means, even though it would have a much higher expected number of visits and occurrences in test cases, this would not increase the testing time significantly. State  $S_1$  has to be regarded quite different. The expected residence time in a test case is 9.00000, so many traversals of this state in test cases have a larger impact on testing time.

State ID	Mean residence time	Expected number of visits	Visiting probab.	Expected residence time
S1	9.00000	1.00000	1.00000	9.00000
S4	0.01000	2.00000	1.00000	0.02000
S201	0.40000	1.50200	0.40000	0.60080
S234	10.00000	0.37538	0.22232	3.75380
S9998	0.00000	1.00000	1.00000	0.00000
S9999	0.00000	1.00000	1.00000	0.00000

TABLE I  
NEW AND CLASSIC COMPUTATIONS FOR CASE STUDY ESP

*i) Test Case Generation:* Test case have been generated with the help of the algorithms described in section III-F. The most probable test case described a typical use case of the ESP functionality: After a drive, the ESP switch is pushed so that it leads to a deactivation of the ESP functionality. Later, the ESP switch is pressed once, but the duration is too small so that it is not activated again. This test case covers 31.43% of the states of the usage model and 14.06% of all stimuli are invoked. The sojourn time of the states and execution duration of the stimuli is the expectation value of the assigned transition. A test suite with 50 test cases was generated with the help of the *random walk algorithm*. The 50 test cases cover all states of the modell and 98.44% of the stimuli are invoked. The duration of the execution of stimuli now varies according to the distributions assigned, thus increasing the ability of automatically derived test cases to validate non-functional requirements and to discover failures.

*j) Results:* The case study ESP demonstrated the usefulness of our extension to classic MCUMs. The quality of the model is improved, as time and timing aspects are systematically described. The indicators for the test planning profit, as computations are now possible that give information about testing time, that is finally reflected in testing effort. These information are hardly derivable from classic MCUMs. Also the test case generation and the generated test cases profit, as the time information helps to develop algorithms to derive test cases that, one the one hand, have a higher ability to discover failures and on the other hand take into account

testing time and thus effort.

#### IV. FUTURE WORK

Our future work engages in the development of test generation strategies that make optimized use of the time information. This concerns algorithms to generate test suites for the validation of requirements and the efficient use of the test bench, which are in our case mainly HIL systems. Therefore, we develop a method how non-functional requirements can be associated with the model. The test strategies should incorporate this knowledge and support the decision on what can be tested under given time constraints.

#### V. CONCLUSIONS

In this paper, Timed Usage Models (TUM) were presented. TUM allow the systematic integration of time and variability in timing in the test model. Thus, indicators for the test planning and test case generation profit from this extension. Indicators that give information about the test subject and not about the graph abstraction of the model, can be automatically derived from the model. We evaluated our approach on a functionality in the automotive domain. The results show that substantial advantages arise from our approach. The time information can be used both for the test suite execution planning and for the test case generation. In the first case, indicators such as the number of occurrences of a state in a test case is not an abstract number, but provides the test designer with information about the expected duration of a test case. Additionally, this information is used by the algorithms for the generation of test cases. It is possible to generate test cases that do not only reflect the variability of stimuli, but also the variability in timing of stimuli. On account of this, TUMs are currently drawn for energy-management and air conditioning functionality of vehicles.

#### REFERENCES

- [1] F. van Meel, G.-P. Duba, T. Bock, and B. Strasser, "Developing properties for driver assistance systems by means of an inovative and constant development process," *FISITA 2008 World Automotive Congress - Springer Automotive Media*, vol. II, 2008.
- [2] T. Bauer, F. Bohr, D. Landmann, T. Beletski, R. Eschbach, and J. Poore, "From requirements to statistical testing of embedded systems," in *SEAS '07: Proceedings of the 4th International Workshop on Software Engineering for Automotive Systems*. Washington, DC, USA: IEEE Computer Society, 2007, p. 3.
- [3] S. J. Prowell, "A stopping criterion for statistical testing," in *In Proceedings of the 37th Hawaii International Conference on Systems Sciences (HICSS37)*, 2004.
- [4] P. A. Brooks and A. M. Memon, "Automated gui testing guided by usage profiles," in *ASE '07: Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering*. New York, NY, USA: ACM, 2007, pp. 333–342.
- [5] S. J. Prowell, "Computations for markov chain usage models," Software Engineering Institute, Carnegie-Mellon University, Pittsburgh, USA, Tech. Rep., 2000, uT-CS-03-505.
- [6] R. German, *Performance Analysis of Communication Systems with Non-Markovian Stochastic Petri Nets*. New York, NY, USA: John Wiley & Sons, Inc., 2000.
- [7] G. Kiffe, *EXtended Automation Method (EXAM) Konzeptpapier Version 3.0*, Audi AG, Ingolstadt, Mai 2009.
- [8] AUDI AG, *Spezifikation ESP Anzeige- und Warnkonzept*, 85057 Ingolstadt, Germany, 2005.

# FLAT — A Fast Lattice-Based Algorithm for Test Suite Reduction

Ahmed Raafat Abuzeid  
College of Engineering  
Arab Academy for Science  
Cairo, Egypt  
ahmed.r.abuzeid@gmail.com

Haitham S. Hamza  
Dept. of Information Technology  
Cairo University  
Giza, Egypt 12613  
hshamza@acm.org

Ismail Abdel Hamid Taha  
College of Management  
Arab Academy for Science  
Cairo, Egypt  
ismail\_taha@aast.edu

## Abstract

*Software testing is a critical part of software development that consumes a considerable amount of the development time and effort. One of the main factors that greatly affects the testing cost is the construction and execution time of test suites. Several research efforts have been focusing on developing minimization algorithms to reduce the size of test suites, and hence the overall testing time. This paper proposes a new minimization algorithm, namely, the fast lattice-based (FLAT) algorithm that can construct a minimized test suite in a smaller time as compared to that needed by the known Delayed-Greedy (DG) algorithm. Evaluation results show that, for some test suites, the proposed algorithm can reduce the construction time of the minimized test suites by more than 80% as compared to that of the DG algorithm.*

## 1. Introduction

Testing is an important process that is performed to support quality assurance. Testing activities consist of designing test cases, executing those test cases, and examining the results produced by those executions. Studies indicate that more than fifty percent of the cost of software development is devoted to testing, with the percentage for testing critical software being even higher [1].

As the software evolves, its test suites also evolve: new test cases are added to exercise new functionality or to maintain test adequacy. As a result, test suite size increases, and so does the cost of managing and using these test suites. Consequently, researchers have investigated the notion of test suite minimization. Effectively a test suite can be reduced to a smaller suite that guarantees equivalent coverage.

The motivation for test suite minimization is straightforward: by reducing test suite size we reduce the costs of executing, validating, and managing those test suites over

future releases of the software. Another motivation in the minimization process is the construction of the minimized suite and how to reduce the time needed to construct the minimized test suite. This paper proposes a new minimization algorithm, namely, the fast lattice-based (FLAT) algorithm that can construct a minimized test suite in a smaller time as compared to that needed by the known Delayed-Greedy (DG) algorithm. The proposed algorithm exploits the features of Formal Concept Analysis (FCA) [2] to achieve minimization. FCA is a mathematical framework that can be used in data analysis and knowledge representation.

The rest of this paper is structured as follows. Section 2 provides the problem statement. Section 3 reviews background and related work. In Section 4 and Section 5, the proposed algorithm is presented and evaluated, respectively. Conclusions are presented in Section 6.

## 2. Problem Statement

As software is developed, test engineers create test cases to detect defects in the software. Engineers may employ several test case generation techniques to create large numbers of test cases that are potentially beneficial in terms of their defect detection ability. As software is modified, test cases are added to cover its new and modified features. At some point in the software development life cycle, the time it takes to run the entire test suite against a modified version of the software may become excessive, leading to an increased cost in the overall development life-cycle.

Since test cases may be redundant with respect to the statements, functions, paths, or other program elements they execute, the problem of test suite minimization (reduction) was investigated, which focuses on reducing the test suite to obtain a subset that yields equivalent coverage with respect to some criterion. The goal of test suite minimization is to generate a test suite that is smaller (and therefore cheaper to execute and maintain) but that still retains much of its original ability to detect faults.

The test suite minimization problem may be stated as follows [3]:

**Given:** Test suite TS, a set of test case requirements  $r_1, r_2, \dots, r_n$  that must be satisfied to provide the desired test coverage of the program, and subsets of TS,  $T_1, T_2, \dots, T_n$ , one associated with each of the  $r_i$ s such that any one of the test cases  $t_j$  belonging to  $T_i$  can be used to test  $r_i$ .

**Problem:** Find a representative set of test cases from TS that satisfies all of the  $r_i$ s. The  $r_i$ s in the foregoing statement can represent various test case requirements, such as source statements, decisions, definition-use associations, or specification items. A representative set of test cases that satisfies all of the  $r_i$ s must contain at least one test case from each  $T_i$ ; such a set is called a hitting set of the group of sets  $T_1, T_2, \dots, T_n$ .

To achieve a maximum reduction, it is necessary to find the smallest representative set of test cases. However, this subset of the test suite is the minimum cardinality hitting set of the  $r_i$ s, and the problem of finding such a set is NP-complete [4]. Thus, minimization techniques resort to heuristics.

### 3. Background and Related Work

In this section, we review the basic concepts of Formal Concept Analysis (FCA) and some key related work in the context of test suite reduction.

#### 3.1. Formal Concept Analysis (FCA)

Formal concept analysis (FCA) is a branch of mathematical lattice theory that provides means to identify meaningful groupings of objects that share common attributes as well as provides a theoretical model to analyze hierarchies of these groupings, [5].

A main component in FCA is the formal context. A formal context consists of a set of objects  $O$ , a set of attributes  $A$ , and a binary relation  $R \subseteq O \times A$  between objects and attributes, indicating which attributes are possessed by each object. Formally, it can be defined as  $C = (A, O, R)$ . From the formal context, FCA generates a set of concepts where every concept is a maximal collection of objects that possess common attributes.

A concept  $C$  is defined as a pair of sets  $(X, Y)$  such that:

$$X = \{o \in O \mid \forall a \in Y : (o, a) \in R\} \quad (1)$$

$$Y = \{a \in A \mid \forall o \in X : (o, a) \in R\} \quad (2)$$

Where  $X$  is said to be the extent of the concept  $C$  and  $Y$  is said to be its intent.

The set of all concepts of a formal context and the partial ordering can be represented graphically using a concept lattice. A concept lattice consists of nodes that represent the

concepts and edges connecting these nodes. The concept lattice is the basis for further data analysis.

Table 1 shows an example of a context with a set of four objects and a set of five attributes. The set of concepts deduced from relation  $R$  is given in Table 2.

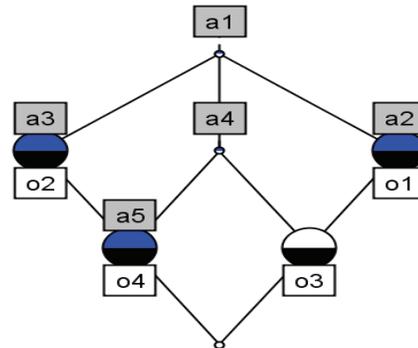
**Table 1. Sample context table**

$R$	$a1$	$a2$	$a3$	$a4$	$a5$
$o1$	x	x			
$o2$	x		x		
$o3$	x	x		x	
$o4$	x		x	x	x

**Table 2. List of concepts**

Concept	Extent	Intent
$C1$	$\{o1, o2, o3, o4\}$	$a1$
$C2$	$\{o1, o3\}$	$a1, a2$
$C3$	$\{o2, o4\}$	$a1, a3$
$C4$	$\{o3, o4\}$	$a1, a4$
$C5$	$\{o3\}$	$a1, a2, a4$
$C6$	$\{o4\}$	$a1, a3, a4, a5$
$C7$	$\{\}$	$a1, a2, a3, a4, a5$

Figure 1 shows the concept lattice of the formal context given in Table 1



**Figure 1.** Concept Lattice for the Formal Context in Table 1

#### 3.2. Related Work

There have been many prior studies of test suite reduction while keeping some coverage requirement constant. The main concept in all these studies work on the selection of a subset of test suite that can test all the requirements should be tested by this test suite [6] [7] [8].

Many prior studies [9][10][11] investigated the effects of test suite minimization on the fault detection capabilities of the reduced test suites. These studies show that the fault-detection capabilities of test suites can be severely compromised by test-suite reduction.

A classical approximation algorithm [12] [13] for the minimum set-cover problem uses a simple greedy heuristic. This heuristic picks the set that covers the most points, throws out all the points covered by the selected set, and repeats the process until all the points are covered.

Formal Concept Analysis was also used in [14] for reducing a test suite for web applications. They consider the URLs used in a web session as the attributes and each web session as a test case. In this work, one test case from each of the strongest concept in the concept lattice is selected to generate a reduced test suite to cover all the URLs covered by the unreduced suite.

The approach presented in [15] used concept analysis iteratively to exploit the implications among the coverage requirements (attribute reductions) and the implications among the test cases (object reductions), in addition to the owner reductions, to derive a reduced suite and applied the greedy heuristic only when needed. This is in contrast to the classical greedy algorithm which applied the greedy heuristic at every step. Thus, this Delayed-Greedy algorithm is guaranteed to generate reduced suites that are of the same size or smaller size than those generated by the classical greedy algorithm.

#### 4. The Proposed FLAT Algorithm

This paper introduces a fast lattice-based algorithm (FLAT) to construct a minimized test suite in a fast manner. The proposed FLAT algorithm exploits the lattice structure and properties to reduce the time needed to construct a minimized suite.

The key point for minimization here is to discard test cases that are redundant with respect to the specified coverage criterion.

##### 4.1. Solution Approach

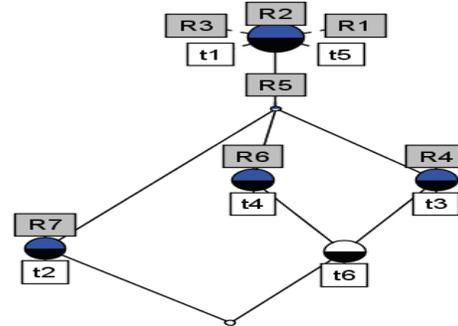
In order to apply FCA we need to map the concept of objects and attributes to the terminology of software testing. We use objects to represent the test cases and attributes to represent the requirements of the software being tested.

Lattices generated from concept analysis contexts for test suite reduction will exhibit several interesting properties and relationships, we attempted to analyze these properties and implications in the sample context shown in Table 3.

Interpreting the sample concept lattice in Figure 2, a test case  $t$  covers all requirements at or above the concept

**Table 3. Properties context**

$R$	$R1$	$R2$	$R3$	$R4$	$R5$	$R6$	$R7$
$t1$	x	x	x				
$t2$	x	x	x		x		x
$t3$	x	x	x	x	x		
$t4$	x	x	x		x	x	
$t5$	x	x	x				
$t6$	x	x	x	x	x	x	



**Figure 2.** Properties Concept lattice

uniquely labeled by  $t$  in the lattice. For example, the test case  $t3$  covers  $R1, R2, R3, R4$  and  $R5$ .

Similarly, all test cases at or below the concept uniquely labeled by requirement  $R$  covers this requirement  $R$ . In Figure 2, test cases  $t2, t3, t4,$  and  $t6$  cover the requirement  $R5$ .

A node labeled with a test case  $t$  and no attributes indicates that  $t$  covers no unique requirement. For example,  $t6$  doesn't cover a unique requirement.

The top of the lattice denotes the requirements that are covered by all of the test cases in the lattice. In our example,  $R1, R2$  and  $R3$  are covered by all of the test cases in our test suite.

The bottom of the lattice denotes the test cases that cover all the requirements in the context. Here, lattice bottom is not labeled with any test case, denoting that no test case covers all the requirements in the context.

To determine the common requirements covered by two separate test cases  $t1$  and  $t2$ , the closest common node  $c$  toward the lattice top starting at the nodes labeled with  $t1$  and  $t2$  is identified.  $t1$  and  $t2$  both cover the requirements at or above  $c$ . For example, test cases  $t3$  and  $t4$  both cover the requirements  $R1, R2, R3,$  and  $R5$ .

Similarly, to identify the test cases that jointly cover two requirements  $R1$  and  $R2$ , the closest common node  $d$  toward lattice bottom starting at the nodes uniquely labeled by  $R1$  and  $R2$  is determined. All test cases at or below  $d$

cover both  $R1$  and  $R2$ . For example, test cases  $t3$  and  $t6$  cover both requirements  $R4$  and  $R5$ .

## 4.2. Proposed Algorithm

The proposed FLAT algorithm is composed of four steps. The input for this algorithm is the context table of the test suite. The actual steps of the algorithm executes on the concept lattice itself. The outline for the algorithm is given in Figure 3.

**Step 1:** Formulate the concept lattice from the initial context.

**Step 2:** If the set of test cases of the node at the bottom level of the lattice is not empty then select one test case from it. [If there is an output from this step then don't continue the rest of steps and this test case is the minimized test suite for this context].

**Step 3:** Select test cases from all the nodes in the level above the bottom level of the concept lattice, and if a node has more than one test case select only one from them.

**Step 4:** From the set of test cases in step 3; if the node of any test case uniquely covers a specific requirement; then select these test cases and check if these test cases together cover all requirements or not. If yes then these test cases are the minimized test suite for this context; otherwise, the set of test cases in step 3 is the minimized test suite for this context.

## 4.3. Illustrative Example

The following is a demonstration of executing the proposed algorithm on the formal context shown in Table 4.

**Table 4. Case 1 context**

$R$	$R1$	$R2$	$R3$	$R4$	$R5$
$t1$	x	x			
$t2$		x	x		
$t3$	x		x		
$t4$		x	x	x	
$t5$	x				x

**Step 1:** Formulate the concept lattice from the input context as given in Figure 4.

**Step 2:** The set of test cases of the node at the bottom level of the lattice is empty.

**Step 3:** Here we select test cases from all the nodes in the level above the bottom level of the concept lattice, then the selected test cases from this step are  $t1, t3, t4$  and  $t5$ .

**Step 4:** From the set of test cases in step 3; we can find that the node of  $t4$  and  $t5$  uniquely covers  $R4$  and  $R5$  respectively; and also we note that  $t4$  and  $t5$  together covers all the requirements. Then the minimized test suite contains only  $t4$  and  $t5$ .

```

1: INPUT: Context table for given test suite  $T$ .
2: OUTPUT: Set of test cases in minimized test suite  $T_{\text{minimized}}$ .
3: PROCEDURE
4:  $T_{\text{minimized}} = \text{empty}$ ;
5:  $T_{\text{bottom}} = \text{empty}$ ; {Set of test cases at the bottom level of lattice}
6:  $T_{\text{above\_bottom}} = \text{empty}$ ; {Set of test cases from all nodes in the level above the bottom level of lattice}
7:  $T_{\text{unique}} = \text{empty}$ ; {Set of test cases which its node uniquely covers a specific requirement}
8: Formulate the concept lattice from the initial context;
9: if  $T_{\text{bottom}} \neq \text{empty}$  then
10:    $T_{\text{minimized}} = T_{\text{bottom}}\{t\}$ ; {Select one test case from  $T_{\text{bottom}}$ }
11: else
12:    $T_{\text{above\_bottom}} = \text{Select test cases from all nodes in the level above the bottom level of the lattice and if a node has more than one test case select only one from them}$ ;
13:   for all  $t$  in  $T_{\text{above\_bottom}}$  do
14:     if node of test case  $t$  uniquely covers a specific requirement then
15:        $T_{\text{unique}} = T_{\text{unique}} + t$ ;
16:     end if
17:   end for
18:   if test cases in  $T_{\text{unique}}$  together cover all requirements then
19:      $T_{\text{minimized}} = T_{\text{unique}}$ ;
20:   else
21:      $T_{\text{minimized}} = T_{\text{above\_bottom}}$ ;
22:   end if
23: end if
24: return  $T_{\text{minimized}}$ 
25: END PROCEDURE

```

**Figure 3.** Proposed FLAT algorithm.

## 4.4. Tool Support for FLAT

The Concept Explorer is an open source software that implements the basic functionalities of the FCA [16]. In the proposed work, the Concept Explorer was extended to implement the DG algorithm and the proposed algorithm. Figure 5 gives a snapshot for the extended tool.

## 5. Algorithm Assessment

### 5.1. Assessment Approach

We evaluate the proposed FLAT algorithm in terms of the time needed to generate the minimized test suite. We

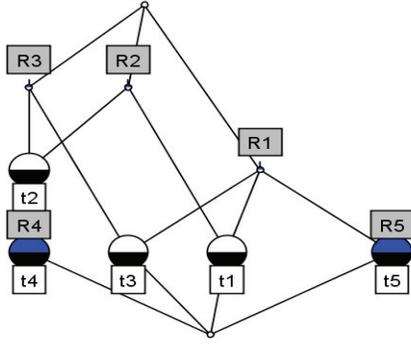


Figure 4. Concept Lattice for Context in Table 4

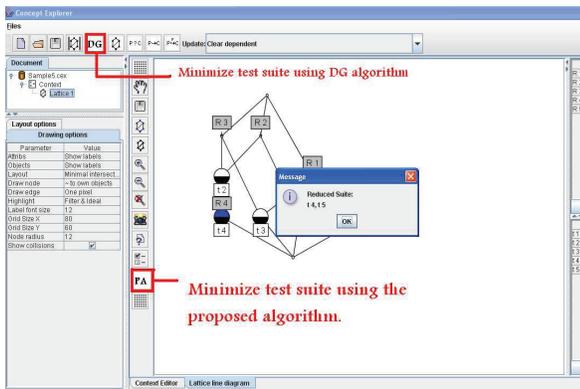


Figure 5. Tool support for the FLAT algorithm.

record the time consumed to generate the minimized test suite for the FLAT algorithm and compared to that required by the Delayed Greedy algorithm.

The FLAT algorithm was evaluated using three different case studies as follows:

**Case 1:** Composed of 5 requirements and 5 test cases.

**Case 2:** Composed of 8 requirements and 8 test cases.

**Case 3:** Composed of 94 requirements and 8 test cases, obtained from the repository in [17].

We have chosen the *Delayed Greedy* (DG) algorithm [15] as a baseline for comparison because it also uses FCA in the minimization process. Although both algorithms make use of FCA, however, our algorithm utilizes the concept lattice graph instead of using the context table directly as in the DG algorithm. We show that using the lattice graph can significantly reduce the time needed for minimizing a test suite.

## 5.2. Assessment Results

Table 5 summarizes the experimental results for applying the proposed FLAT algorithm and the Delayed Greedy

algorithm on the example context presented in section 4.3.

Table 5. Results for Case 1.

	Proposed FLAT	Delayed-Greedy
Initial Suite	$t1, t2, t3, t4, t5$	
Reduced Suite	$t4, t5$	$t4, t5$
Time (ms)	62	157

Table 7 gives the results for applying the proposed FLAT algorithm and the Delayed Greedy algorithm on the example context presented in Table 6.

Table 6. Formal context for Case 2.

$R$	$R1$	$R2$	$R3$	$R4$	$R5$	$R6$	$R7$	$R8$
$t1$	x	x	x		x	x	x	
$t2$		x		x		x		
$t3$		x	x	x				
$t4$				x	x	x	x	x
$t5$		x			x		x	
$t6$		x		x				
$t7$			x		x		x	
$t8$					x	x	x	

Table 7. Results for Case 2.

	Proposed FLAT	Delayed-Greedy
Initial Suite	$t1, t2, t3, t4, t5, t6, t7, t8$	
Reduced Suite	$t1, t4$	$t1, t4$
Time(ms)	78	296

Table 8 shows the results for applying the proposed FLAT algorithm and the DG algorithm on the context of Case 3.

As we can see in Figure 6, for the three cases under consideration, the proposed FLAT constructs the same minimized test suite obtained by the DG algorithm but in a reduced time. In particular, the FLAT algorithm reduces the construction time of the minimized test suites by 61%, 74%, and 90% for Case 1, Case 2, and Case 3; respectively.

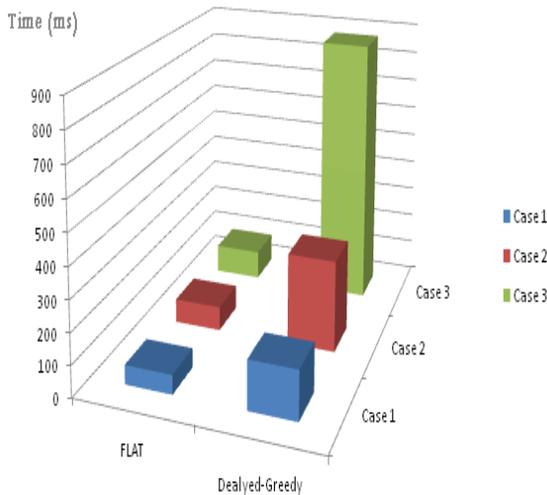
It is worth noting that, as the number of test cases and /or tested requirements increases, the FLAT algorithm exhibits better performance as compared to the DG algorithm. This evident by the results obtained for Case 3 above. This result holds for larger contexts as well. The reason for this is simple: the DG algorithm consumes much time in processing large context tables, whereas, our algorithm process only one level in the lattice regardless the size of this lattice.

## 6. Conclusion

In this paper we addressed the problem of constructing a minimized test suite in a reduced time. Formal Concept

**Table 8. Results for Case 3.**

	Proposed FLAT	Delayed-Greedy
Initial Suite	$t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8$	
Reduced Suite	$t_1, t_3, t_6, t_7, t_8$	$t_1, t_3, t_6, t_7, t_8$
Time(ms)	94	875

**Figure 6.** Assessment Results.

Analysis was applied to support the new algorithm due to its great support and contributions in the field of software engineering. A comparison between the proposed algorithm and the related work was held to introduce some sort of improvement in minimization process time consumption. A tool that supports the implementation of the proposed algorithm was also introduced to demonstrate the usability of the algorithm.

## References

- [1] E. Kit. Software Testing in the Real World: Improving the Process, Addison-Wesley, USA, 1995.
- [2] G. Birkhoff. Lattice Theory, volume 5, American Mathematical Soc. Colloquium Publications, 1940.
- [3] M. J. Harrold, R. Gupta and M. L. Soffa. A methodology for controlling the size of a test suite. ACM Trans. on Softw. Eng. and Methodology, 2(3):270285, July 1993.
- [4] M. Garey and D. Johnson. Computers and Intractability. W.H. Freeman, New York, 1979.
- [5] B. Ganter and R. Wille. Formal Concept Analysis: Mathematical Foundations. Springer Verlag, 1999.
- [6] Dennis Jeffrey and Neelam Gupta. Test Suite Reduction with Selective Redundancy. icsm, pp. 549-558, 21st IEEE International Conference on Software Maintenance (ICSM'05), 2005
- [7] Adenildo da Silva Simao, Rodrigo Fernandes de Mello and Luciano Jose Senger. A Technique to Reduce the Test Case Suites for Regression Testing Based on a Self-Organizing Neural Network Architecture. Proc. 30th Int. Computer Software and Applications Conf. (COMPSAC'06), pp. 93-96, 2006.
- [8] Jennifer Black, Emanuel Melachrinoudis and David Kaeli. Bi-Criteria Models for All-Uses Test Suite Reduction. icse, pp. 106-115, 26th International Conference on Software Engineering (ICSE'04), 2004
- [9] G. Rothermel, M. J. Harrold, J. von Ronne, C. Hong, and J. Ostrin. Experiments to Assess the Cost-Benefits of Test-Suite Reduction. Technical Report 99-60-09, Oregon State University, September 1999.
- [10] G. Rothermel, M. J. Harrold, J. V. Ronne and C. Hong. Empirical Studies of Test-Suite Reduction. J. of Software Testing, Verification, and Reliability, 4(2), Dec 2002.
- [11] G. Rothermel, M. J. Harrold, J. Ostrin and C. Hong. An empirical study of the effects of minimization on the fault detection capabilities of test suites. Int. Conf. on Software Maintenance, pp. 34-43, Nov. 1998.
- [12] V. Chvatal. A Greedy Heuristic for the Set-Covering Problem. Mathematics of Operations Research. 4(3), August 1979.
- [13] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein. Introduction to Algorithms. MIT Press, Second Edt, 2001.
- [14] Sreedevi Sampath, Sara Sprenkle, Emily Gibson, Lori Pollock and Amie Souter Greenwald. Applying Concept Analysis to User-Session-Based Testing of Web Applications. IEEE Tran. on Software Eng., vol. 33, no. 10, pp. 643-658, Oct. 2007.
- [15] S. Tallam and N. Gupta. A Concept Analysis Inspired Greedy Algorithm for Test Suite Minimization. Proc. Sixth ACM SIGPLAN-SIGSOFT Workshop Program Analysis for Software Tools and Eng., pp. 35-42, 2005.
- [16] Serhiy A. Yevtushenko. System of data analysis "Concept Explorer". In the Proc. of the 7th Conf. on Artificial Intelligence KII-2000, pp. 127-134, 2000.
- [17] <http://sir.unl.edu/portal/index.html>

# Semantic Document Architecture for Desktop Data Integration and Management

Saša Nešić<sup>1</sup>, Dragan Gašević<sup>2</sup>, Mehdi Jazayeri<sup>1</sup>

<sup>1</sup>Faculty of Informatics, University of Lugano, Lugano, Switzerland

<sup>2</sup>School of Computing and Information Systems, Athabasca University, Athabasca, Canada

<sup>1</sup>{sasa.nesic, mehdi.jazayeri}@usi.ch, <sup>2</sup>dgasevic@acm.org

## Abstract

*The paper presents a novel desktop document architecture, namely SDArch, which attempts to integrate data from heterogeneous desktop applications into a unified desktop information space. To achieve this, SDArch introduces a new document representation model, which establishes explicit semantic links between fine-grained units of document data based on the conceptualization of their semantics. The SDArch semantic search and navigation services, which run on this unified desktop information space, aim to improve the search and navigation within desktop data, thus improving the effectiveness and efficiency of desktop users in carrying out their daily tasks. We report on a usability evaluation of the SDArch prototype and an experimental evaluation of the proposed semantic search. The evaluation results are promising. We present an analysis of these results.*

## 1. Introduction

Over the last decade, personal desktops have faced the problem of information overload due to increasing computational power, easy access to the Web and cheap data storage. Moreover, an increasing number of diverse end-user applications have led to the problem of information fragmentation. Accordingly, discovering and accessing information stored on the desktop represent the main challenges for desktop users carrying out their daily activities. Several research projects in the field of the Semantic Desktop [6, 10, 7, 4] have attempted to solve these problems by applying Semantic Web technologies [1] on local personal desktops. They attempt to set up an additional semantic layer on top of the existing desktop infrastructure. On this layer, desktop resources are identified by their URIs and described by RDF descriptions using agreed upon vocabularies (ontologies) [1]. However, the desktop information space differs from the Web in many aspects, which hamper the full potential of the applied Semantic Web technologies. A key problem with the use of RDF in the context of the desktop is the modeling of the relationship between a

resource URI from the semantic layer containing resource descriptions and the resource itself (i.e., the content of the resource). While on the open Web it is acceptable that links can become broken, on the desktop, the integrity of resources and their descriptions must be ensured [6]. Accordingly, the existing semantic desktop approaches need synchronization mechanisms to propagate modifications to the RDF layer. Due to the complexity of the current desktop infrastructures, the design and implementation of such mechanisms represent the main challenge. The other problem of having a separate semantic layer is the limited ability to identify and describe fine-grained, self-contained information units stored inside conventional desktop documents [5]. This is due to existing conventional document formats, which provide limited mechanisms for structuring and identifying the actual document content.

In this paper we present a novel desktop document architecture, called the Semantic Document Architecture or SDArch, which should solve the problems described above and facilitate desktop data management. With SDArch, desktop users are able to search and navigate across their desktop data more efficiently, thus completing their tasks more efficiently. SDArch introduces a new representation model called the Semantic Document Model (SDM), which integrates the semantic layer into the core of the document data representation structures. The model is inline with the Linked Data principles[3], representing documents as RDF linked data sets, in which document units are identified by globally unique URIs, annotated by an arbitrary number of ontological concepts, interlinked with other document units via explicit semantic links, and connected by the content stream to their binary content. While semantic documents (i.e., instances of SDM) can be browsed and edited in a number of application specific formats, each semantic document has only one permanent instance which is integrated into a unified desktop data graph and stored into a desktop RDF repository. On top of the desktop RDF repository, we have built a service-oriented middleware with a set of services that allow application software to interact with semantic documents. Moreover, the repository exposes a publicly-available SPARQL endpoint, thus enabling remote access to

the desktop data and its integration into the global Linked Open Data (LOD) cloud [3]. Together, the RDF repository, the middleware and the application software that utilizes the middleware services compose SDArch as a three-tier service-oriented architecture.

The remainder of the paper is structured as follows. Section 2 outlines the main characteristics of SDM. In Section 3, we describe the design principles of SDArch. In Section 4, we give a more detailed description of the SDArch services, which implement semantic annotation, indexing and linking of document units from semantic documents. Then, in Section 5, we describe the SDArch search and navigation services and show an example of their integration into application software. Section 6 presents the results of the preliminary evaluation of the proposed architecture. We have a look at some related efforts in Section 7, before concluding the paper with some final remarks in Section 8.

## 2. Semantic Document Model

We have created the semantic document model (SDM) [8], aiming to provide the infrastructure for the integration of semantically related data units, which are fragmented into diverse desktop application formats, thus unifying the desktop information space and improving personal information management [10]. Moreover, the model intends to bridge the gap between the desktop information space and the global LOD cloud, so that desktop users can directly access and integrate information from the LOD cloud as well as publish their desktop data to the LOD cloud.

SDM represents document units as RDF nodes, each of which being uniquely identified by means of an HTTP dereferencable URI [3]. In order to enable semantic linking of document units over shared semantics, SDM provides mechanisms to conceptualize the semantics of document units and to link them to the document units' RDF nodes as semantic annotations. The conceptual representation of a document unit's semantics is achieved by a combination of an arbitrary number of ontological concepts along with their relevance weights for the document unit. By having semantic annotations linked to document units' RDF nodes, SDM establishes semantic links between nodes which share the same annotation concepts.

SDM is formally described by the *smd ontology* [8], which specifies possible types of document units (e.g., `sdm:paragraph`, `sdm:section`, `sdm:table` and `sdm:illustration`), types of hierarchical and structural relationships between document units (e.g., `sdm:hasPart`, `sdm:isPartOf` and `sdm:belongsTo`), and the semantic annotation and semantic linking interfaces. The semantic annotation interface consists of the `sdm:Annotation` entity and its two properties: the `sdm:annotationConcept`, which holds a reference to the concept from an annotation ontol-

ogy and the `sdm:conceptWeight` that determines the relevance of the annotation concept for the document unit it annotates. The semantic linking interface consists of the `sdm:SemanticLink` entity and the following properties: the `sdm:unitOne` and `sdm:unitTwo`, which refer to the document units to be linked; the `sdm:linkConcept` that holds the reference to the ontological concept that annotates both units and determines the type of the semantic relationship; and the `sdm:linkStrength` property whose value determines the strength of the semantic relationship between the units. In addition, every RDF node, representing a document unit, has one content stream connecting the binary content of the document unit, stored into the binary content repository, to the document unit's RDF node.

## 3. Architecture Design

We have designed SDArch as a three-tier architecture composed of a data layer, service-oriented middleware and presentation layer. The data layer contains the desktop RDF repository, the binary data repository, the concept index and the text index. The RDF repository stores semantic documents, that is, RDF instances of SDM, while the binary content of document units are placed into the binary content repository. SDArch maintains one single concept index that enables semantic search (Section 5) over RDF data and a single text index that enables content-based search over the binary content of document units. The middleware provides a set of services that implement functionalities of SDArch, defines communication protocols among the services and provides a service registry for registering or unregistering services from the middleware. Two main usage scenarios we focussed on while designing the current prototype of the SDArch middleware are: 1) semantic document authoring and publishing by transforming conventional documents into semantic documents and 2) semantic search and navigation across semantically integrated collections of desktop documents. New scenarios can be easily enabled by adding new services to the SDArch middleware and making them communicate with the existing services. Figure 2 shows an illustration of SDArch depicting internal processes and relationships among services for the two above-mentioned usage scenarios. In the following two sections we describe in more detail the services and processes related to both scenarios. For the prototype development of the SDArch middleware, we used Windows Communication Foundation (WCF) .NET framework. As an example of the user interface of the SDArch services, we have developed a set of tools called 'SemanticDoc' and integrated them into MS Office. Implementation details, snapshots, and demos of the use of the SDArch prototype and SemanticDoc tools can be found on our project Web page<sup>1</sup>.

<sup>1</sup><http://www.semanticdoc.org>

## 4 Semantic Document Authoring

We distinguish between two possible scenarios for authoring semantic documents. The first one is to transform documents of conventional document formats (e.g., MS Word, MS PowerPoint or PDF documents) into semantic documents. The second one is to use application software developed specifically for authoring semantic documents from scratch. In order to let users continue to work in familiar environments while taking advantage of the new architecture as well as to enable conversion of existing documents into semantic documents, in the current SDArch prototype we implemented the first authoring scenario. Moreover, it is also inline with our hypothesis that semantic documents can be browsed and edited in a number of application specific formats, that is, there can be many possible document views but there is only one permanent semantic document copy integrated into the desktop data graph [8].

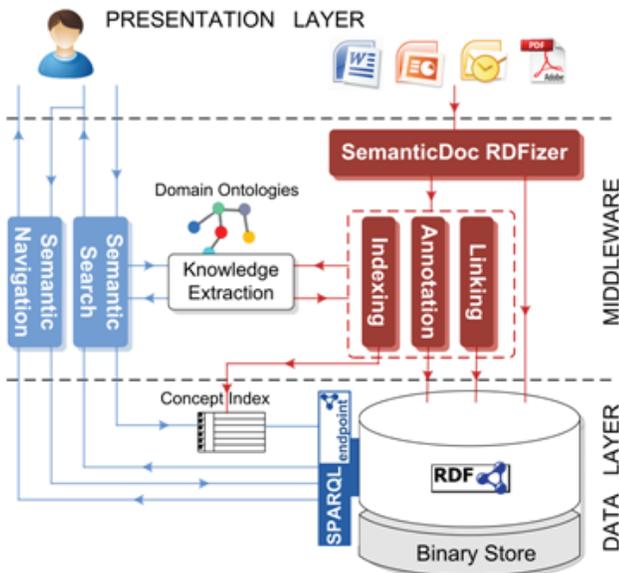


Figure 1: SDArch layers - illustration

The transformation process starts with the *SemanticDoc RDFizer* service which scans a document to be transformed, recognizes document units (e.g., sections, paragraphs, tables, illustrations and slides), generates RDF nodes and URIs of document units, generates RDF links which represent structural and hierarchical relationships between the units and stores the generated RDF data into the desktop RDF repository. For each document unit, the *SemanticDoc RDFizer* calls the *annotation*, *indexing* and *linking* services. These three services all rely on the discovery of concepts from domain ontologies which conceptualize human-consumable information stored in document units. The concept discovery is realized by the *knowledge extraction service*, which utilizes domain ontologies and applies the concept exploration algorithm (CEA)[9], to discover ontological concepts of the document units and to calculate their

relevance weight to the document units. As domain ontologies, the service uses existing ontologies from the Semantic Web. There already exist a number of widely used, shared ontologies, many of which are applicable for desktop data. By using shared ontologies we enable different applications, either on the same or networked desktops, to interpret desktop data in a semantically correct way.

The *annotation service* uses the discovered ontological concepts and their corresponding weights to form the semantic annotations, which are then linked to the RDF nodes of the document units, via the annotation interface (Section 2). The *indexing service* adds an entry for each document unit to the concept index. The entry contains a weighted concept vector [9] of the document unit's annotation concepts. After the semantic annotation and indexing, the next step is to link semantically related document units by establishing semantic links between their RDF nodes. If two document units are annotated by the same ontological concept, then the two document units share some semantics, which determine the semantic relationship between them. The *linking service* computes the strength of the semantic relationship between the units over the annotation concept as a product of the concept's weights of each unit. Then, the service generates an RDF instance of the semantic link interface (Section 2), thus creating the semantic link between the document units. By setting up semantic links between document units originating from heterogeneous, application-specific formats, desktop data becomes connected into an integrated, desktop-data graph stored into the desktop RDF repository. Moreover, by exposing the SPARQL endpoint to the desktop RDF repository, we contribute to the LOD cloud [3] by adding data from desktop documents.

## 5 Semantic Search and Navigation

In the traditional hypertext Web, browsing and searching are often seen as the two dominant modes of interaction [1]. While browsers provide the mechanisms for navigating the information space, search engines are the place where navigation begins. In order to bring this paradigm to the desktop information space, SDArch provides the *semantic search* and *navigation* services. We now describe the main features of the two services and present some examples of their integration into an existing, well-established document authoring environment such as MS Office.

The search process normally starts with users constructing a query that reflects their information needs. As the initial form of the user query, the semantic search service takes a free-text query. The service then models the semantic meaning of the query by forming a weighted query concept vector [9] that we refer to as a semantic query. To do this, the search service actually employs the knowledge extraction service, which discovers query concepts and calculates their relevance weights for the query. The search

service forms semantic queries in the same way the indexing service represents document units in the concept index. This results in both the document units and the user query being represented by the corresponding weighted concept vectors. The rest of the search process is as follows. From the concept index, the search service discovers document units which are indexed by some of the concepts from the semantic query. Then, the service calculates the similarity between the discovered document units and the semantic query, by calculating the cosine similarity between the document units' concept vectors and the query's concept vector. Based on the calculated similarity, the service ranks the document units and forms the retrieval list. Finally, for each document unit from the list, the service queries the RDF repository to acquire the details of the document unit and sends them to the user. Figure 2 a) shows the interface of the 'SemanticDoc' search add-in that we have developed to enable MS Office users to get access to the search service.

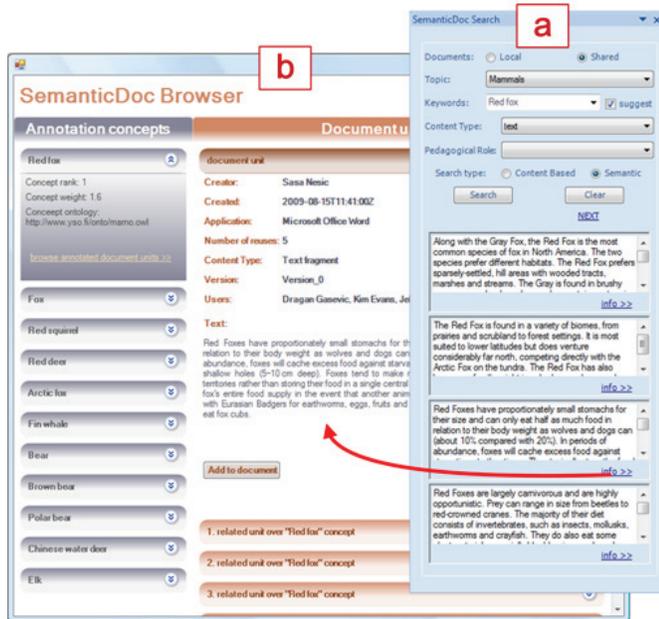


Figure 2: SemanticDoc: a) search interface b) browser

The semantic navigation service enables users to traverse semantic documents by navigating along the semantic links. The navigation process assumes the existence of an exploratory interface through which the users interact with the semantic navigation service. Figure 2 b) shows the 'SemanticDoc' browser that we have developed and integrated into MS Office as a supplement to the 'SemanticDoc' search add-in. The navigation process starts by the user browsing a document unit retrieved by the search service and clicking on a label one of the unit's annotation concepts, which represents a potential semantic link. This user action activates the navigation service, which takes the URIs of the document unit of the clicked concept, forms a corresponding SPARQL query (Figure 3), and executes it against the

RDF repository. The document unit can be linked to many document units via the same semantic link determined by the clicked concept, thus the query can return multiple document units. The query orders the returned document units by the strength (Section 4) of the semantic link between them and the initial document unit. Finally, the service acquires the details of the document units, and sends them to the user.

```

1 PREFIX sdm:<http://semanticdoc.org/sdm.owl#>
2 PREFIX uri:<http://semanticdoc.org/URI#>
3 SELECT ?targetUnit ?strenght
4 WHERE {
5   ?link sdm:linkConcept uri:concept_a32d254
6   ?link sdm:unitOne uri:unit_b42c177
7   ?link sdm:unitTwo ?targetUnit
8   ?link sdm:linkStrength ?strength }
9 ORDER BY ?strength

```

Figure 3: SPARQL query executed by the navigation service

## 6. Preliminary Evaluation

As a preliminary evaluation of SDArch we have conducted two evaluation studies: 1) the usability evaluation of the SDArch prototype and the SemanticDoc tools, and 2) the experimental evaluation of the proposed semantic annotation, indexing and retrieval.

### 6.1 The Usability Evaluation

The usability evaluation has been designed to consider user *effectiveness*, *efficiency* and *satisfaction* in using the SDArch prototype and SemanticDoc tools. It consisted of a set of tasks related to authoring of course material, that is, preparing a PowerPoint presentation for the 'Software Design Patterns' topic. Recent research in learning content authoring has shown that most authors reuse and modify existing content, available in their own archives or on the Web [2], rather than authoring from scratch. Therefore, the efficient exploration of data on user desktops is the main prerequisite to efficient authoring. In the evaluation, 12 participants, from three universities, took part. All participants were volunteers and were familiar with software design patterns. The evaluation document set consisted of more than 70 topic's related documents. The documents were available in their original formats (i.e., Word, PowerPoint and PDF) and as semantic documents generated by SDArch and stored into desktop RDF repository. All participants were asked to perform a set of seven tasks, designed so that their successful completion results in creation of an overall presentation of an appropriate quality level. Moreover, we asked the participants to perform the tasks two times arranged in two continuous but unlimited time sessions. One half of the participants first used the system with conven-

tional desktop tools and then the system with SDArch, and the other half used the systems in the opposite order. The evaluation session consisted of two phases, namely observation and feedback. In the observation phase, we were observing the participants and tracking their behavior using screen recordings while they were conducting the evaluation tasks. All participants performed the tasks on our PC with remote access control.

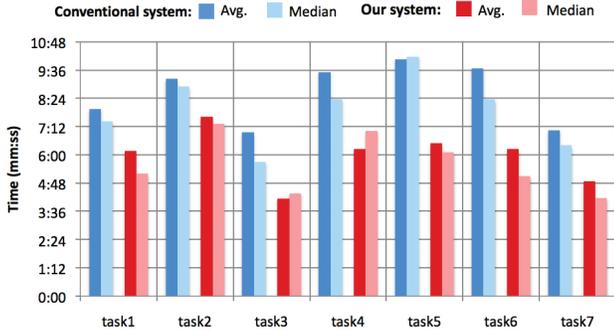


Figure 4: Average and median task execution times

With respect to user effectiveness, we tracked how many and which tasks participants completed successfully. Results showed that all participants successfully completed all tasks using both systems, which was mostly because of unlimited time sessions and the genuine motivation of the participants. With respect to user efficiency we measured the execution times, the amount of mouse clicks and the number of window switches during the tasks’ executions. Figure 4 presents the average and median execution times, for each task for both systems. From the figure we can see that, regarding task execution time, our system outperforms the conventional system for all the evaluation tasks.

Task	Rel. performance for (mouse clicks)		Rel. performance for (window switches)	
	Average	Median	Average	Median
1	87.1%	86.9%	21.5%	33.3%
2	79.6%	86.4%	35.8%	26.6%
3	75.3%	68.8%	24.0%	29.4%
4	78.9%	83.8%	28.4%	27.7%
5	51.1%	53.3%	25.6%	21.0%
6	57.1%	56.6%	28.1%	26.3%
7	80.4%	86.2%	21.6%	28.5%

Table 1: Relative user performance when using the SDArch with respect to conventional desktop

Table 1 shows the relative performance of the participants when using our system with respect to the use of the conventional system. For example, a value of 87.1% indicates that participants using our system needed 87.1% of the mouse clicks that participants using the conventional system needed. All values from Table 1 are less than 100%, which means that the performance of the participants regarding both number of mouse clicks and window switches

are better when using our system for all tasks. The measured execution times, number of mouse clicks and number of window switches showed that the efficiency of the participants when using our system is better than their efficiency when using the conventional system, with respect to the measured criteria.

The participants provided us their subjective feedback about the usefulness and ease-of-use of our system by filling a questionnaire at the end of the evaluation session. The questionnaire contained 9 statements, first 5 statements (S1-S5) designed to gather a subjective evaluation of the system usefulness, and the following 4 statements (S6 - S9) designed to gather subjective evaluation of the ease-of-use of the system. Statements S5 and S9, which express the overall satisfaction regarding the usefulness and ease-of-use respectively, were the two highest-rated statements with an average rating of 4.8 out of 5. The other statements were also rated as highly positive with average ratings ranging from 4.1 to 4.7.

## 6.2 The Experimental Evaluation

The first goal of the experimental evaluation was to show if our semantic annotation approach can produce a substantial amount of semantic annotations, which is crucial for the semantic linking and integration of desktop data. The second goal was to show how accurate the generated semantic annotations are and to which extent they can improve search on the desktop. The detailed discussion with complete evaluation results are presented in [9]. In this paper we give just a brief overview.

We have designed the experimental evaluation as a proof of concept and it was not meant to address issues of scalability or efficiency. The document set that we used was composed of 350 Word documents containing records for steel, aluminum, copper, titanium, and other metals. As the annotation ontology we used the METALS ontology, which contains over 3,500 concepts about metals and their applications. Both the document set and the ontology, we obtained from KEY-to-METALS<sup>2</sup> company, which maintains one of the world’s most comprehensive metals database. We have transformed the evaluation document set into five collections of semantic documents, by applying five different concept discovery strategies:  $S_1$  - simple syntactic matching,  $S_2$  - lexically expanded syntactic matching, and  $S_3, S_4, S_5$  - lexically expanded syntactic matching and the semantic matching. The first two strategies are present in most existing semantic annotation approaches. The last three strategies comprise all the features of the concept exploration algorithm [9] implemented by the SDArch knowledge extraction service (Section 4), and only differ in the values of some of the algorithm’s parameters. For each strategy, Table 2 shows the total number and the average weight of

<sup>2</sup><http://www.keytometals.com/>

the syntactic and semantic matches. These results indicate that our annotation model (i.e.,  $S_3$ ,  $S_4$  and  $S_5$ ) has potential to produce substantial amount of semantic annotations. To evaluate the performance of the proposed semantic search we formed ten queries related to the data of the evaluation document collection and asked three KEY-to-METALS engineers to assess the relevance of document units (i.e., paragraphs) of the collection to the queries. The queries were then executed against each of the five semantic document collections. The measured values of the precision and recall for each of the five semantic document collections showed that the proposed semantic search improves retrieval in terms of both precision and recall.

Strategy	Syntactic matching		Semantic Matching	
	Num.	Avg, Weight	Num.	Avg Weight
$S_1$	1524	2.56	-	-
$S_2$	3182	3.62	-	-
$S_3$	3182	3.62	6714	2.43
$S_4$	3182	3.62	11102	1.12
$S_5$	3182	3.62	23716	0.27

**Table 2:** Concept discovery results for strategies ( $S_1$ - $S_5$ )

## 7. Related Work

The idea of using Semantic Web technologies to enhance data interoperability and information management on the personal desktops has been widely researched. A number of Semantic Desktop projects such as [6, 10, 7, 4] have the goal of providing a semantic infrastructure that covers all applications and integrates information sources that users operate on. All of these projects attempt to enhance the existing desktop infrastructures by adding an additional semantic layer, providing semantic descriptions that refer to actual resources. In such scenarios, the semantic integration of desktop resources should happen at the semantic layer by interlinking descriptions of semantically related resources instead of linking actual resources. The main problem here is the propagation of modifications to resources and their relationships to the semantic layer. Moreover, the existing Semantic Desktop approaches also pay little attention to the identification and annotation of small, self-contained data units from conventional desktop documents. They rather consider whole documents as identifiable resources. Linked Data [3] could address this problem, but while there are more and more tools available for publishing Linked Data on the Web, there is still a lack of such tools for local desktops. To the best of our knowledge, only the approach presented in [11] offers the model that integrates the semantic layer into the actual desktop file system, enabling explicit semantic links between desktop files. However, the model recognizes only entire files as identifiable and linkable resources. With respect to semantic annotation of desktop documents, many approaches exist[12], but none of them

offer a solution for interlinking data from different documents, which is annotated by the same semantic annotations.

## 8. Conclusions

In this paper we present a novel desktop document architecture, called SDArch, which attempts to semantically integrate the desktop information space and provides the set of services which aim to improve the effectiveness and efficiency of desktop users in completing their daily tasks. The results of the usability and experimental studies conducted with the architecture prototype were promising. In future work we plan to continue to evaluate the usability of SDArch services as well as the performances of the proposed semantic annotation, indexing and search and to provide further statistical analysis of the collected data.

## References

- [1] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific America*, 2001.
- [2] C. Betty and S. Allard. Technology and human issues in reusing learning objects. *Journal of Interactive Media in Education*, 4(1):1–25, 2004.
- [3] C. Bizer, T. Heath, and T. Berners-Lee. Linked Data - The story so far. *Int. Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.
- [4] L. Blunski and et al. A dataspace odyssey: The imemex personal dataspace management system (demo). In *CIDR*, pages 114–119, 2007.
- [5] H. Eriksson. The semantic-document approach to combining documents and ontologies. *Int. Journal of Human-Computer Studies*, 65(7):836–840, 2007.
- [6] T. Groza and et. al. The NEPOMUK Project - On the Way to the Social Semantic Desktop. In *the I-Semantics*, pages 2010–211, 2007.
- [7] D. R. Karger, K. Bakshi, D. Huynh, D. Quan, and V. Sinha. Haystack: A general-purpose information management tool for end users based on semistructured data. In *CIDR*, pages 13–26, 2005.
- [8] S. Nešić. Semantic Document Model to Enhance Data and Knowledge Interoperability. *Annals of Information Systems, Springer*, 6:135–162, 2009.
- [9] S. Nešić, M. Jazayeri, F. Crestani and D. Gašević. Concept-Based Semantic Annotation, Indexing and Retrieval of Office-Like Document Units. Tech. Rep., 2010/01 University of Lugano. <http://www.inf.usi.ch/phd/nesic/tech-rep.pdf>, 2010.
- [10] L. Sauermaann and D. Heim. Evaluating long-term use of the gnosis semantic desktop for pim. In *International Semantic Web Conference*, pages 467–482, 2008.
- [11] B. Schandl and B. Haslhofer. The sile model - a semantic file system infrastructure for the desktop. In *ESWC*, pages 51–65, 2009.
- [12] V. Uren and et al. Semantic Annotation for Knowledge Management: Requirements and a Survey of the State of the Art. *J. Web Sem.*, 4(1):14–28, 2006.

# Using the whole structure of ontology for semantic relatedness measurement

Ehsan KhounSiavash  
Computer Engineering Department  
University of Isfahan  
Isfahan, Iran  
KhounSiavash@comp.ui.ac.ir

Ahmad Baraani-Dastjerdi  
Computer Engineering Department  
University of Isfahan  
Isfahan, Iran  
AhmadB@eng.ui.ac.ir

**Abstract**— Semantic relatedness measurement is one of the key issues in many different fields of AI. The methods of semantic relatedness measurement are very susceptible to inconsistencies of the ontology structure. The reason is that they do not use the whole structure of the domain to compare concepts. On the other hand, semantic relationship has wide span; it can contain similarity, opposition, inclusion, etc. In semantic relatedness measurement it is very important to take into account most available semantic relationship between concepts. In this paper a model for representing the position of concepts based on all defined relationships and other concepts in the domain has been proposed. Through this model concepts can be compared by taking into account all their implicit and explicit relationships with the other concepts of the domain. We represent each concept by a vector consisting of paths from the concept to the other concepts of the domain. The experimental results, using WordNet, show that not only our approach operates as well as other methods; it also resolves most inconsistencies in the other path-based ones.

**Keywords:** *Semantic Relatedness Measurement; Semantic Space; Ontology; Path;*

## I. INTRODUCTION

Measuring the semantic relatedness of two concepts is one of the most important problems in ontology engineering. Human's brain may recognize two semantically related items, due to wide variety of Semantic links between them. The most familiar types of semantic relationships consist of similarity, inclusion, interaction, bilaterality or even opposition. For example, taxi and car are semantically related because of their common applications, car and wheel because wheel is part of car, car and fuel because car uses fuel or even heat and cold because they are opposite.

**Motivation.** Semantic relatedness measurement plays a key role in knowledge management and artificial intelligence [1]. It is directly used by many parts of information retrieval systems like word sense disambiguation, query expansion, semantic annotation, and text summarization [2]. Some of the important approaches in data mining especially for text classification and clustering rely on ontology. In knowledge

management and ontology engineering semantic relatedness measurement is used for ontology learning, ontology merging, etc.

**Research Challenges.** Considerable amount of semantic relatedness (or similarity) measurement methods like [3, 4] are based on is-a taxonomy. The main purpose of is-a taxonomy is indicating common features, and the information it provides is more suitable for similarity measurement. Therefore, it is expected that these methods are unable to reflect other types of semantic relationships as well. The methods using other kinds of relationships have a lot of inconsistencies caused by faults in ontologies. So, they try to avoid these inconsistencies using wide variety of heuristics like weighting relations and paths, specifying patterns for meaningful paths, etc.

Studying current methods of semantic relatedness (or similarity) measurement carefully, and considering their weakness and strength lead us to the following conclusion: Available ontologies have faults that seriously bias semantic relatedness (or similarity) measurement methods. The following are the most important ones:

- Inappropriate organization. For example, the basic taxonomy of WordNet instead of being subjective is based on word's part of speech. Furthermore, there is no link among members of different parts of speech.
- Limited diversity of defined relations. For example WordNet as a general ontology has only about 10 types of relationship.
- Boolean relations. Defined relations conform 0-1 logic i.e. all of them have the same significance [4].
- Unbalanced distribution of relations. The noun category constitutes the main part of WordNet. On the other hand, the is-a taxonomy is the most populated part of WordNet [5]. Furthermore density of relations in the is-a taxonomy is not equal in different parts [3].
- Rational inconsistency. For example in the is-a taxonomy of WordNet, taxi and bus have been placed in two different and also far categories.

**Contributions.** In this paper, a method for semantic relatedness measurement is proposed. The method uses the position of concepts in the domain to compare them. In

order to do this, each concept of the domain will be represented by a vector in the semantic space of the domain. The vector represents the position of the concept in the domain; it uses all implicit and explicit relationships of the concept with the other concepts. To constitute these vectors we use the length of path between concepts and try to take into account all defined relations. Therefore, since all available relations are being used in building vectors, the most available features of semantic relationship would be used. On the other hand, all the other concepts of the domain with their relations to a concept participate in the vector of the concept. Thus, the final result would be less affected by the inconsistencies of the ontology structure.

In the next section, we will discuss existing methods of semantic relatedness measurement using ontology. In section 3, at first, some of the inconsistencies of WordNet and their bad effects on path-based methods will be illustrated by a brief example. Then our method for modeling the position of concepts in the semantic space of the domain will be explained. In section 4, the advantages and disadvantages of the model in operating on a common dataset will be discussed and then it will be compared with the other methods. In section 5, we will conclude and explain remarkable future works.

## II. RELATED WORK

There are two different trends in semantic relatedness measurement; one of them use lexical resources like ontology, Roget's Treasures, etc. The other one, e.g. [6], is based on the contents of documents and their relations like.

The classic taxonomy of semantic relatedness measurement methods which use lexical resources, consists of information path-based, theoretic-based, and gloss-based approaches [7].

In path-based methods, the semantic relatedness of concepts is determined using their distance in the graph of the ontology. Different formula using different factors are used to estimate the semantic distance between concepts. However, the underlying idea of all such methods is the same; the shorter is the length of path between two concepts, the stronger the semantic relationship between them will be. The main differences among these methods include the way they find the shortest path between two concepts and the parameters they use for measuring the semantic relationship conveyed by the path. For example [8, 9] select paths that conform with some heuristic patterns. Or [10] uses the shortest path which just includes is-a relations and traverses the most common ancestor of the concepts. The other heuristic of path-based methods is assigning weight to relations. The commonly used factors for weighting include type and depth of relations [10-12]. These methods are seriously affected by inconsistencies of the ontology structure mentioned in the previous chapter.

The information theoretic based methods use information contents of concepts to measure their semantic relatedness. Therefore, the main challenge of this approach is estimating the information content of concepts. Reference [4] used the amount of shared information content by two

concepts to determine their semantic similarity. It defines the shared information content as the information content of the most common ancestor of two concepts. The information content of a concept is defined as  $-\log [p(c)]$  by [4]. So that  $p(c)$  is the probability by which the concept  $c$  is used in a corpus. This definition for information content implies that the amount of information content should increase steadily by ascending in the taxonomy of is-a. This definition is widely used to measure semantic similarity of concepts. The main drawback of this approach is its dependence on external resources to find information content of concepts. It motivated [5, 14] to estimate information contents independent of external resources.

References [15] uses the shared information content to show concepts in the semantic space of the domain by means of semantic vectors. Thus, the semantic similarity of concepts is measured using the cousin similarity of their vectors.

The gloss-based methods are motivated to utilize the unused parts of information provided by ontology. First time [16] used overlaps of glosses of words for word sense disambiguation. Pedersen *et al.* expanded this idea in different ways [17-20]. In [17], they augmented the gloss of a concept by adding the gloss of its neighbor concepts. They also assumed that longer phrases are more important than shorter ones. In [19], the concepts of the domain are shown by vectors in a space that consists of words used in glosses of the concepts.

There are also hybrid methods that try to combine different approaches in order to remove one's disadvantages by the other one's advantages.

Reference [3] is considered as an improved version of [4] which uses the length of path as the correcting factor.

Reference [21] uses nonlinear relations to take into account the effect of length of path, depth of most common ancestor, and information content of engaged concepts.

In [2], the length of path between concepts, the depth of their most common ancestor, and the intersection of their glosses are used to estimate their semantic relatedness.

## III. PURPOSED METHOD

One of the methods of semantic relatedness measurement is comparing their positions i.e. their relationships with the other concepts in the domain. This comparison is equal to compare two nodes of a graph, whereas this is not a standard operation in graph theory. At the first look, using the distance between two concepts can be a good criterion to compare the positions of the concepts. But experiments show that this idea, at least due to faults of ontologies, suffers from a lot of inconsistencies.

### A. A Brief Example

Fig. 1 illustrates part of is-a taxonomy of WordNet. The judgments of some path-based methods of semantic relatedness measurement for some pairs of Fig. 1 are shown in Table 1.

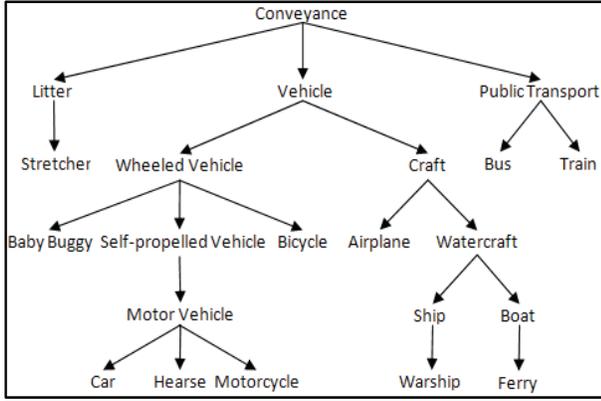


Figure 1. Part of is-a structure of WordNet.

TABLE 1. THE VALUES OF SEMANTIC RELATEDNESS FOR SOME PAIRS OF FIG. 1 [23]

Pairs	Hirst and St-Onge	Wu and Palmer	Leacock and Chodorow
Car-Bus	5	0.66	1.60
Car-Ship	5	0.69	1.49
Car-Train	3	0.69	1.60
Car-Airplane	3	0.72	1.60
Car-Stretcher	0	0.69	1.60

As shown, the values of semantic relatedness for these pairs are completely inconsistent. For example according to [8] car and bus are as related as car and ship. Reference [10] supposes that car and airplane are the most semantically related concepts. Or [22] assigns the same values for four of them. Considering uneven structure of WordNet and the main idea of these methods (just using one path), this problem seems completely natural and expected.

We suppose that finding a way to use the whole structure of ontology for comparing concepts will remove most of these inconsistencies. As in this way, the noisy data, obtained from the inconsistent parts of ontology, would constitute a negligible portion of the entire data.

### B. Compare Concepts Using the Whole Structure

Suppose that the graph of an ontology is shown by adjacency matrix. This matrix just shows explicit relationships among the concepts of the domain. We name this matrix as “elementary semantic adjacency matrix”. But in addition to explicit relationships to adjacent concepts, each concept has implicit semantic relationships with the other concepts of the domain. These implicit relationships are shown by explicit relationships of the adjacent concepts to the other concepts of the domain.

The “semantic adjacency matrix” of an ontology is a non-boolean matrix which includes both explicit and implicit semantic relationships. We name this matrix as Semantic Space Matrix (SSM). Every row (or column) of this matrix represents a concept of the domain in the  $n$ -dimensional space of the ontology (1). So that  $n$  is the number of concepts in the domain.

$$\overrightarrow{concept} = SSM[concept][1..n] \rightarrow \quad (1)$$

$$SSM[concept][i] = SemRel(concept, concept_i)$$

Thus, using the inner production (cousin similarity), we are able to compare two concepts as their position in the ontology point of view (2).

$$SemanticRelatedness(c_1, c_2) = \frac{\overrightarrow{c_1} * \overrightarrow{c_2}}{|\overrightarrow{c_1}| * |\overrightarrow{c_2}|} \quad (2)$$

The remaining problem is building the Semantic Space Matrix.

### C. Transforming Adjacency Matrix to Semantic Space Matrix

To show implicit relationships among concepts and constitute the semantic space matrix, we need to measure semantic relatedness of concepts. During solving a problem, we need the answer of the same problem. It seems that we have arrived to the starting point. But there are two important points. First, in this phase the semantic relatedness of two concepts just indicates one dimension of its semantic vector in the semantic space. Considering numerous dimensions of semantic vectors, it is not expected that imprecise values for some of dimensions can mainly bias the final result. Second, the values are being calculated in this phase, at last, will be compared to each other.

The underlying principles of using distance between concepts to measure their semantic relatedness are: 1) The distance of concepts in the graph of ontology represents the semantic distance between them; 2) Semantic distance is in inverse relation with semantic relationship. Therefore, all methods which use the length of path between concepts to measure semantic relatedness can be used to transform adjacency matrix to semantic space matrix. Some of such methods assign different weights to relations and have different criteria to find meaningful paths between concepts. Thus, the semantic distance of adjacent concepts would be the weight of their relation and for non-adjacent concept would be the weight of path between them.

### D. The Brief Example (Continue)

As a preliminary test of the mentioned ideas, the semantic vectors for samples of Table 1 were built as the following circumstances. The value of  $n$  in (1) is the number of noun Synsets in WordNet 2.0. The  $SemRel(concept, concept_i)$  is defined as (3), so that the maximum allowable length of path is limited by  $\lambda$  i.e. the value of  $SemRel$  for two Synset with length of the shortest path greater than  $\lambda$  is 0. The variables  $\alpha$  and  $\beta$  are used for tanning.

$$SemRel(concept, concept_i) = \frac{\alpha}{distance(concept, concept_i)^{-\beta}} \quad (3)$$

All available relations in WordNet 2.0 were used to find the shortest path between concepts and paths are rough mixture of different relations. Assuming the same value

(one) for all types of relations, best results were found by  $\lambda = 5$ ,  $\alpha = 0.5$ , and  $\beta = 2$ .

The inner productions for semantic vectors of the former example are shown in Table 2. Although considering different contexts and applications, these values also have inconsistencies. But in conditions that we can judge about semantic relatedness of concepts regardless of context, the results of our method are completely according to the expectation.

#### IV. EVALUATION

Rubenstein and Goodenough (1965) asked 51 persons to judge about the semantic similarity of 65 pairs of words ranging from “highly synonymous” to “semantically unrelated”. Miller and Charles (1991) found similar results in a similar study using 30 pairs of Rubenstein and Goodenough’s and 31 subject [24]. The results of their experiments made a baseline for the evaluation of semantic relatedness measurement methods.

The Rubenstein and Goodenough’s dataset was chosen due to its variety in comparison with Miller and Charles’. In order to evaluate our idea purely, we did not take into consideration any correcting factors of path-based methods like depth, relation weighting, avoiding meaningless path, etc. The results of previous experiment using this dataset and same value for parameters  $\alpha$ ,  $\beta$ , and  $\lambda$  are depicted in Fig. 2 and Table 3. These results are worthy of discussing from two sides. First, they show important points of weakness and strength of our method in measuring semantic relatedness of different pairs. Second, the overall result (correlation) shows congruence of our method with the others.

##### A. IntraMethod Discussion

As it is shown in Fig. 2, the values of semantic relatedness for noticeable number of cases are underestimated by our method. We suppose that some kinds of important relationships have not been taken into account in our method. For example, the instances flagged by \* in Fig. 2 are related by is-a relationship (with or without intermediate relations). This kind of relationship indicates very strong similarity and implicates strong semantic

relatedness. However, semantic relatedness for these cases were measured less than expected. There is the same scenario for instances flagged by #. Although they are not directly related by is-a relations, but according to is-a taxonomy of WordNet they are obviously very similar and having strong semantic relationship is expected for them.

It seems this defect is the direct consequence of using (3) as an imprecise method for measuring semantic relatedness in the phase of semantic space matrix constitution. It is expected that improving (3) or using existing path-based methods instead, could effectively improve our results.

##### B. InterMethod Discussion

The correlations of some prominent methods of semantic relatedness measurement along with ours are listed in Table 3. Although, the correlation of 0.838 is not an excellent score in comparison with the others, but some points are worthy of attention.

First, we were going to resolve some of consistencies caused by the structure of ontology. The acceptable correlation of 0.838 and the example mentioned in section 3 show that our method not only operates as well as other methods but also it is less affected by inconsistencies of the ontology structure.

Second, in Rubenstein and Goodenough’s experiment and similar studies, subjects were asked to specify semantic similarity of pairs and not their semantic relatedness. So, considering all types of semantic relationships, semantic relatedness of concepts does not conform to their similarity.

TABLE 2. THE VALUES OF SEMANTIC RELATEDNESS FOR INSTANCES OF TABLE 1 WHICH CALCULATED BY OUR METHOD

Pairs	$\cos(\vec{c}_1, \vec{c}_2)$
Car-Taxi	0.76
Car-Bus	0.53
Car-Motorcycle	0.50
Car-Train	0.46
Car-Airplane	0.43
Car-Ship	0.29
Car-Stretcher	0.07

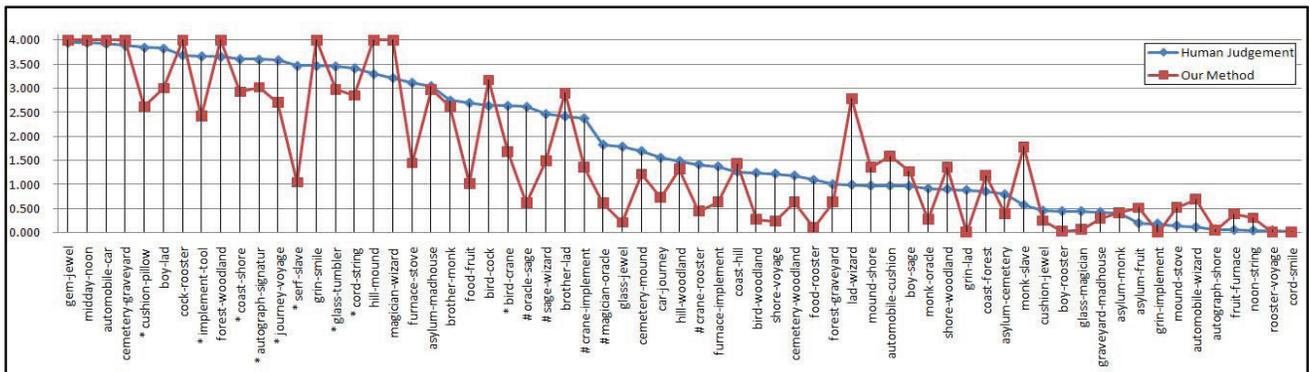


Figure 2. The values of semantic relatedness for instances of Rubenstein and Goodenough (1965).

TABLE 3. THE CORRELATIONS OF SOME OF SEMANTIC RELATEDNESS (OR SIMILARITY) MEASUREMENT METHODS [2]

Method	R&G	Path	IC	Gloss
Resnik	0.779	N	Y	N
Hirst and St-Onge	0.786	Y	N	N
Leacock and Chodorow	0.838	Y	N	N
Lin	0.819	N	Y	N
Jiang and Conrath	0.842	Y	Y	N
Li et al.	0.891	Y	Y	N
Patwardhan and Pedersen	0.900	N	N	Y
Yang and Powers	0.910	Y	N	N
Alvarez and Lim	0.900	Y	N	Y
Pirró	0.908	N	Y	N
Ours	0.838	Y	N	N

Third, realizing the intended meaning of a polysemous word is merely possible through its application in the context. For this reason, some methods like [17] in spite of having good results in application based evaluation, are not very successful in evaluation by Rubenstein-Goodenough or Miller-Charles.

Therefore, we hope that using a more accurate method in building semantic space matrix, on one hand, and application-based evaluation, on the other hand, show the real value of measuring semantic relatedness using the whole structure of ontology.

## V. CONCLUSION AND FUTURE WORKS

In this paper we introduced a robust method for semantic relatedness measurement. The method models the concepts as semantic vectors, using all available relationships and other concepts of the domain, so that it would be able to compare them based on their position in the domain.

The results of experiments show that, using the whole structure of ontology for modeling the position of concepts in their domain, our method is able to tolerate noisy data and correct inconsistencies caused by them. They also indicate that similarity constitutes an important portion of semantic relationship. Therefore it is necessary to devote special effort on recognizing and reflecting it in semantic relatedness measurement.

The comparison with other methods shows that our method operates as precise as the other methods and it is expected to outperform them in application based experiments.

Our future works include two parts. One part is improvement of measuring semantic relatedness in constituting semantic space matrix phase. To do so we should use a more precise method as *SemRel(concept, concept<sub>i</sub>)* so that it differentiate among relations according to their types, depth, etc. The other part is evaluating our method in one of the applications of semantic relatedness measurement like information retrieval.

## REFERENCES

[1] D. N. Le and A. E. S. Goh, "Current Practices in Measuring Ontological Concept Similarity," 2007, pp. 266-269.

[2] M. A. Alvarez and S. J. Lim, "A Graph Modeling of Semantic Similarity between Words," 2007, pp. 355-362.

[3] J. J. Jiang and D. W. Conrath, "Semantic similarity based on corpus statistics and lexical taxonomy," in *International Conference Research on Computational Linguistics (ROCLING X)*, 1997.

[4] P. Resnik, "Using information content to evaluate semantic similarity in a taxonomy," 1995, pp. 448-453.

[5] G. Pirró, "A semantic similarity metric combining features and intrinsic information content," *Data & Knowledge Engineering*, vol. 68, pp. 1289-1308, 2009.

[6] A. G. Maguitman, F. Menczer, H. Roinestad, and A. Vespignani, "Algorithmic detection of semantic similarity," 2005, p. 116.

[7] G. Varelas, E. Voutsakis, P. Raftopoulou, E. G. M. Petrakis, and E. E. Milios, "Semantic similarity methods in wordNet and their application to information retrieval on the web," 2005, p. 16.

[8] G. Hirst and D. St-Onge, "WordNet: An electronic lexical database, chapter Lexical chains as representations of context for the detection and correction of malapropisms," The MIT Press, Cambridge, MA, 1998.

[9] L. Mazuel and N. Sabouret, "Semantic relatedness measure using object properties in an ontology," 2008, p. 681.

[10] Z. Wu and M. Palmer, "Verbs semantics and lexical selection," 1994, pp. 133-138.

[11] M. Sussna, "Word sense disambiguation for free-text indexing using a massive semantic network," 1993, pp. 67-74.

[12] D. Yang and D. M. W. Powers, "Measuring semantic similarity in the taxonomy of WordNet," 2005, p. 322.

[13] D. Lin, "An information-theoretic definition of similarity," 1998, pp. 296-304.

[14] E. Blanchard, P. Kuntz, M. Harzallah, and H. Briand, "A tree-based similarity for evaluating concept proximities in an ontology," 2006, pp. 3-11.

[15] J. W. Kim and K. Candan, "CP/CV: concept similarity mining without frequency information from domain describing taxonomies," 2006, p. 492.

[16] M. Lesk, "Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone," 1986, pp. 24-26.

[17] S. Banerjee and T. Pedersen, "Extended gloss overlaps as a measure of semantic relatedness," 2003, pp. 805-810.

[18] S. Patwardhan, S. Banerjee, and T. Pedersen, "Using measures of semantic relatedness for word sense disambiguation," *Lecture Notes in Computer Science*, pp. 241-257, 2003.

[19] S. Patwardhan and T. Pedersen, "Using WordNet-based context vectors to estimate the semantic relatedness of concepts," 2006.

[20] T. Pedersen, S. Banerjee, and S. Patwardhan, "Maximizing semantic relatedness to perform word sense disambiguation," *Supercomputing institute research report umsi*, vol. 25, 2005.

[21] Y. Li, Z. A. Bandar, and D. McLean, "An approach for measuring semantic similarity between words using multiple information sources," *IEEE Transactions on knowledge and data engineering*, pp. 871-882, 2003.

[22] C. Leacock and M. Chodorow, "Combining local context and WordNet similarity for word sense identification," *WordNet: An electronic lexical database*, vol. 49, pp. 265-283, 1998.

[23] T. Pedersen, S. Patwardhan, and J. Michelizzi, "Wordnet::similarity-measuring the relatedness of concepts," 2004, pp. 1024-1025.

[24] A. Budanitsky and G. Hirst, "Evaluating wordnet-based measures of lexical semantic relatedness," *Computational Linguistics*, vol. 32, pp. 13-47, 2006.

# Composer-Science: A Semantic Service Based framework for Workflow Composition in e-Science Projects.

Laryssa Machado da Silva<sup>1</sup>

<sup>1</sup> Master Program in Computational Modeling  
Federal University of Juiz de Fora -  
Juiz de Fora – Minas Gerais – Brazil  
laravit@gmail.com

Regina Braga<sup>1,2</sup>, Fernanda Campos<sup>1,2</sup>

<sup>2</sup> Software Quality Research Group  
Federal University of Juiz de Fora -  
Juiz de Fora – Minas Gerais – Brazil  
(regina.braga, fernanda.campos)@ufjf.edu.br

**Abstract**—An important element in e-Science research is the scientific workflow that, in general, is very long, composed of many computations, and represents a scientific process. Generally, a scientific workflow is very large what turns it difficult to define. One possibility to help its definition is using tools that aggregate semantics to assist in the composition. In this context, this paper presents a proposal that aims to make possible the composition of scientific workflows, considering semantic search of web services and incorporating them in the workflow definition

**Keywords:** Workflow composition, Web Service, Semantic Web, Ontologies

## I. INTRODUCTION

The Computer Science revolutionized scientific research and today is recognized as the third pillar together with theory and experimentation [1]. The use of computer resources in the development of scientific research benefits the work of scientific communities facilitating the sharing of data and computing services, and contributing to build an infrastructure of data and a distributed research community [2]. This context, in which the computer becomes essential to the successful of scientific research from different areas, is the e-Science context [3], where science is performed with computer support and becomes more efficient.

The e-Science activities are growing around the world, accompanied by a proliferation of data and tools. This brings new challenges, for example, how to understand and organize these resources, how to share and reuse successful experiments (tools and data), and how to provide interoperability among data and tools from different locations and used by people with different profiles. In the e-Science context, the technology of workflow provides room for the resolution of problems of scientific nature, facilitating the creation and execution of experiments using a large amount of available data and services [4].

This paper, therefore, discusses the use of Semantic Web as a way to facilitate the discovery and composition of Web Services for the development of scientific workflows. Our contribution is the development of a framework that uses semantic resources such as ontologies and semantic Web services, together with Petri Nets and Workflow languages to

assist scientific workflows composition, thus making the process simpler and more effective. This paper is organized as follows: in section 2, we present the theoretical basis, in section 3, some proposals and related work are analyzed, in section 4, our proposal to develop a framework that aggregates the semantic in the development of scientific workflows is formulated and its architecture is presented and, finally, in section 5 conclusions and future work are discussed.

## II. THEORETICAL BASIS

The term e-Science is generally used to describe the development of an infrastructure able to provide access to remote facilities, remote computing resources, storage of information in dedicated databases, and dissemination and sharing of data, results and knowledge. Some characteristics of e-Science are: access to a vast collection of data, use of computer resources on a large scale, use of heterogeneous and dynamic resources of multiple organizations and dynamic workflows [5]. To monitor and facilitate the use of various resources available in the context of e-Science and to enable efficient analysis of large volumes of data, workflows with specific features for e-Science have been defined, which are called scientific workflows.

Traditionally, the Web has been seen as a distributed source of information. The joint application of Semantic Web and Web services in order to generate “more intelligent” Web services, has led to Semantic Web services [10]. A Semantic Web service consists of a Web service with a semantic description that allows the automatic discovery, composition and invocation of the service, for example, using intelligent agents able to process available semantic information [7].

## III. RELATED WORKS

The CelOWS [11] is a broker based on ontologies, to register, discover and execute biological web services. In this work, a semantic description of a service is used, by storing in the database, the concepts inferred in the ontology. An inference machine is used to retrieve services that match with the concepts provided by the client (scientist). Our proposal is an evolution of this work, including the discovery of Web services through the use of ontologies, making use of OWL-S and inference mechanisms that help in the finding of services.

The discovery of services becomes part of a larger process that involves the composition of scientific workflows using the services discovered, and its subsequent implementation.

The work of [7] has as main objective to improve the efficiency in the consumption of Web services and increase the automation of some tasks related to them. Our proposal differs from theirs as we want to go beyond these functions. Our goal is to perform these same tasks offered by the system proposed by [7] and also allow the composition of scientific workflows with these services.

The work present in [12] classifies Web services according to their interfaces and builds a composite service through logical and physical composition. The work proposed in [13] uses semantic matching, which considers domain dependant information as well as domain independent information. Our proposal differs from them, once it uses ontology in order to specify the user's intention.

The works presented in [15] and [16] classify available services according to their functionalities, where each set of services with the same functionality is called a community service or meta-service, respectively. Their work is similar to our proposal. However, we use ontologies and transform our workflow of metaservices in a concrete one, with the help of Petri nets and WS-BPEL [17]. The work described in [18] proposes a Web service composition method which guarantees functional correctness of Web service composition, including information services, and also improves time complexity of the composition process. However, it does not generate a concrete workflow, as we do in our work.

#### IV. COMPOSER-SCIENCE

With the objective of assisting the discovery of Web services for composition of scientific workflows, we propose the development of a framework named Composer-Science, that performs semantic search of services and compose these services defining a scientific workflow. Specifically, the framework can do:

- a. Storage of domain ontologies (OWL) and semantic annotations of web services (OWL-S), in distributed repositories (databases) of the framework.
- b. Implementation of semantic search in distributed repositories, based on requirements provided by the user, in order to discover semantic Web services that meet these user's requirements.
- c. Parsing, based on structural requirements (input and output), and semantic analysis of the discovered services in order to obtain possible compositions of them.
- d. Generation of workflow models in WS-BPEL based on the compositions discovered.

Figure 1 represents the layers of Composer-Science, and allows a better understanding of the steps taken to achieve its main and secondary objectives. We can divide the framework in two components: the client component that invokes the service, and the component that contains the layers that compose the framework.

The client component implements an interface through which the user interacts with the framework.

The **Search Layer** is the layer that includes the Semantic Search Manager. This module is responsible for semantic search and discovery of services, considering the descriptions

provided by user and the results of the inferences obtained from them. This layer is responsible also for performing inferences (reasoning) services in the domain ontologies, sending as parameters, the semantic specifications for each task, defined based on available ontologies. So, we can obtain, as results of inference, terms related to the ones provided by the user, thus expanding the web services discovering.

The information provided by the user and that obtained by means of inference are used to perform semantic search (discovery) in distributed repositories to find Web services semantically compatible with each task in the workflow. To find the services, the semantic descriptions (OWL-S) of available services in the repositories are analyzed and compared with the semantic data related to each task.

Semantic information is extracted from the semantic description (OWL-S) for each service found in the repositories through the use of OWL-S API [20]. Information obtained from the OWL-S of a given service is compared with information provided by the user considering a particular task, to determine whether the service described by OWL-S can be used to accomplish the task. As search results, we can find several services that meet the specifications of a given task. The Application Layer is responsible for selecting the service that best fits the given task.

The **Application Layer** is composed of the Composition Manager and the Client Manager modules. The Composition Manager is responsible for generating compositions based on the discovered services, which represent the workflow specified by the user as well as the generation of the WS-BPEL model, which is the result of the composition. The Client Manager is responsible for all interaction with the users of Composer-Science. Its main functionality is to prevent that users have access to the internal structure of the framework.

To understand the operation of the Application Layer, we must consider that, after obtaining services that accomplish each task, initially set by the user, as part of the workflow, the Application Layer starts the process of composition of services in order to obtain possible compositions capable of represent the chosen workflow at the end of the process. The composition of services is performed using the paradigm of Petri Nets, which assists in the analysis of structural and semantic compatibility of each service and of the composition.

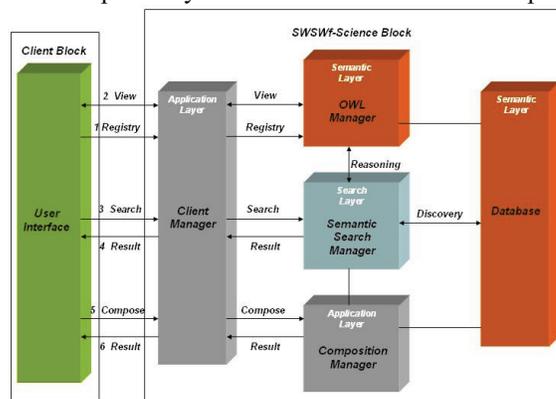


Figure 1. Composer-Science Architecture.

Considering space limitations, this paper will not detail the search process. Part of the search process can be seen in [11]. In this paper we will detail the composition process.

### A. Service Composition

Most studies on the automatic composition of web services provide compositions considering only the input and output of the available services, without considering its semantics [23]. Thus, results that do not meet the user’s intentions can be generated. The framework proposed in this paper aims to solve these problems and enable that scientific workflows are created through composition of semantic web services. Among the languages for composition of Web services, WS-BPEL [17] has distinguished itself by being an XML based language specifically designed to support service composition. In terms of formalization, the basic objective is the use of formal description techniques such as Petri nets, in order to verify composition properties. Petri nets are a mathematical formalism that provides graphical representation. The state and transitions of Petri nets are used to model a logical point of view of applications.

In this section we will describe how Petri nets can be used in order to compose Web services. The methodology consists of three steps: first, we will construct the specification of a Petri Net consistent with the semantics of the user-defined workflow; the second step is to establish the possible semantically compositions with the available services; and in the third step is performed the structural analysis of services composition and the presentation of the possible semantic and structural compositions.

#### Step 1: Definition of the Petri Net topology

Based on the definition of workflow provided by the user, a network is defined to represent the abstract workflow. This definition of the workflow is stored in a relational database and, based on these data, the framework is able to set the Petri Net that represents the abstract workflow. In Figure 2 we have an example of a Petri Net that represents an abstract workflow defined by a given user. For each task, the user defines semantic terms of a domain ontology that can best describe the task and its input and output. This term selection will be described in step 2.

#### Step 2: Possible semantically compositions with the services found

Considering the abstract workflow shown in Figure 2, it is possible to do a semantic search for services in distributed repositories. This search is based on the semantic descriptions of the tasks of the abstract workflow defined by the user and that is represented using Petri Nets

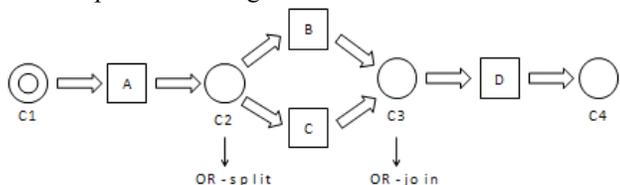


Figure 2. An abstract workflow representation using Petri Net.

In Figure 2, based on the abstract model of the workflow, the framework selects in the database, descriptions related to each task that make up the workflow. Suppose, for example, that scientists are designing an abstract workflow to the domain of cardiac electrophysiology. Thus, it selects the CeLO ontology [11] to perform the semantic search. Task A has CellElement as one of terms selected from CeLO for description of it, task B has the term CellProcess as one of its description, task C has the term CellStructure, and task D has CellMLModel as one of its description. Then, from the terms that describe each task, inferences are made on the ontology in order to retrieve more related terms and thus extend the power of semantic search for services, in the repositories of the framework. The relevant information contained in the semantic descriptions (OWL-S) of services that are stored in distributed repositories, are extracted using the OWL-S API, and then the framework compares the terms (and the inferred ones) that describe the tasks with the terms obtained from the information extracted of each service and tries to match these descriptions. Once the services are discovered, the framework generates executable workflow models that are semantically and structurally possible, which are presented to the user to choose one of the models to generate the corresponding WS-BPEL file.

The services found are assigned to each task of the abstract workflow (represented by the Petri net), as can be seen in Figure 3. After the allocation of the discovered services to the tasks of the abstract workflow, the framework defines Petri nets that represent executable workflows. These Petri nets are considered, in the context of this work, compositions semantically possible. These executable models (possible semantically compositions) are stored in the database so they can be accessed later for the structural analysis of the services.

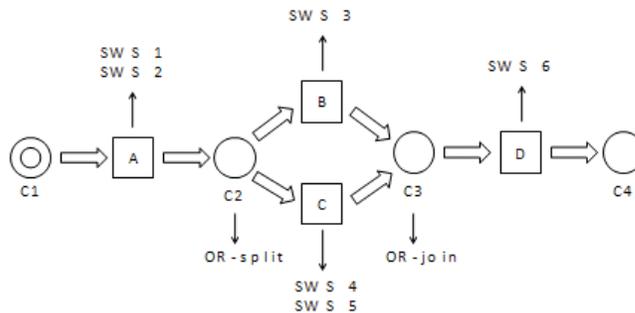


Figure 3. Abstract workflow with assigned web services

#### Step 3: Definition of compositions semantically and structurally possible

Each service of each possible semantically composition (Figure 3) is analyzed structurally (input and output match) so that it can be set if the composition of these services is structurally compatible and so the composition of workflow formed by them is structurally possible too.

Considering each executable model of the workflow, the input and output data of each service are obtained from its WSDL. The code in Figure 4 extracts the names of messages "request" and "response" of the WSDL and their parameters

(PartName) input (request parameters) and output (response criteria), with their respective data types (PartTypeName).

```
public void wsdlData(String wsdlFile) throws WSDLException{
    WSDLFactory factory = WSDLFactory.newInstance();
    WSDLReader reader = factory.newWSDLReader();
    Definition wsdlInstance = reader.readWSDL(null, wsdlFile);
    Map messages = wsdlInstance.getMessages();
    Iterator msgIterator = messages.values().iterator();
    while (msgIterator.hasNext())
    {
        Message msg = (Message)msgIterator.next();
        if (!msg.isUndefined())
        {
            System.out.println("Message: "+msg.getQName());
            Map parts = msg.getParts();
            for(Iterator pIt = parts.values().iterator();
                pIt.hasNext();){
                Part part = (Part) pIt.next();
                System.out.println("PartName: "+part.getName());
                System.out.println("PartTypeName: "+
                    part.getTypeName().toString());
            }
        }
    }
}
```

Figure 4. Code for WSDL parameters extraction

To test the ideas, we developed a prototype that executes a small scientific workflow related to cell models specified in CellML language [21]. Considering space limitations, this validation could not be presented here.

## V. CONCLUSIONS

In the context of e-Science, the computational resources are becoming increasingly important for the conduction of scientific research, accompanied by a proliferation of data and tools. In this article we presented the proposal of a framework that helps in the discovery and composition of semantic Web services. Our contribution is the development of a framework that uses semantic resources such as ontologies and semantic Web services, together with Petri Nets and Workflow languages to assist scientific workflows composition, thus making the process simpler and more effective. Specifically, the Composer-Science allows the storage and search for domain ontologies and web services semantic annotations and also allows compositions based on these semantic annotations and in web services input and output. Future work includes dealing with data provenance related to these compositions and also improves the performance of the framework.

## ACKNOWLEDGMENT

This work was partially supported by FAPEMIG

## REFERENCES

[1] Atkins, D. E.; Droegemeier, K. K.; Feldman, S. I.; Garcia-Molina, H.; Klein, M. L.; Messerschmitt, D. G.; Messina, P.; Ostriker, J. P.; Wright, M. H., 2003, Revolutionizing Science and Engineering Through Cyberinfrastructure: Report of the National Science Foundation Blue-Ribbon Advisory Panel on Cyberinfrastructure.

[2] Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E. A., Tao, J., Zhao, Y., 2004, Scientific Workflow Management and the KEPLER System. Available at

<<http://www.sdsc.edu/~ludaesch/Paper/kepler-swf.pdf>>. Accessed in June 1, 2009.

[3] Hine, C. M., 2006, New infrastructures for knowledge production: understanding E-science. 1.ed. Information Science Publishing.

[4] Pignotti, E., Edwards, P., Preece, A., Gotts, N., Polhill, G., 2008, Enhancing workflow with a semantic description of scientific intent. 5th European Semantic Web Conference, Tenerife, Spain.

[5] Huang, Li-Can; Wu, Zao-Hui; Pan, Yun-He, 2005, A Grid Architecture for Scalable e-Science and Its Prototype. Journal of Software, Vol. 16, No. 4

[6] Gil, Yolanda; Deelman, Ewa; Ellisman, Mark; Fahringer, Thomas; Fox, Geoffrey; Gannon, Dennis; Goble, Carole; Livny Miron; Moreau, Luc; Myers, Jim., 2007, Examining the Challenges of Scientific Workflows. IEEE Computer, vol. 40, no. 12, pp. 24-32.

[7] Sánchez, F. G., García, R. V., Béjar, R. M., Breis, J. T. F., 2009, An ontology, intelligent agent-based framework for the provision of semantic web services. Available at <<http://www.sciencedirect.com>>. Accessed in June 1, 2009.

[8] Baraka, R., Schreiner, W., 2006, Semantic Querying of Mathematical Web Service Descriptions. Web Services and Formal Methods, Springer Berlin/Heidelberg, pp. 73-87.

[9] Studer, R., Benjamins, R., Fensel, D., 1998, Knowledge engineering: Principles and methods. Data and Knowledge Engineering. Available at <<http://www.das.ufsc.br/~gb/pg-ia/KnowledgeEngineering-PrinciplesAndMethods.pdf>>. Accessed in June 1, 2009.

[10] McIlraith, S., Son, T. C., Zeng, H., 2001, Semantic web services. IEEE Intelligent Systems, Vol. 16, No. 2.

[11] Matos, Ely Edison ; Campos, Fernanda; Braga, Regina ; Palazzi, Daniele . CelOWS: an ontology based framework for the provision of semantic web services related to biological models. Journal of Biomedical Informatics **JCR**, v. 43, p. 125-136, 2009.

[12] Agarwal,V.; Dasgupta, K.; Kamik N., Kumar A., A service creation environment based on end to end composition of web services, in: Proceedings of WWW'05, ACM, New York, 2005, pp. 128-137.

[13] Akkiraju R., Ivan A., Goodwin R., Srivastava B., Syeda Mahmood T., Semantic matching to achieve web service discovery and composition, in: Proceedings of CEC/EEE'06, IEEE Computer Society, Washington, DC, 2006, p. 70.

[14] Ye L., Zhang B., Discovering web services based on functional semantics, in: Proceedings of APSCC'06, IEEE Computer Society, Washington, DC, 2006, pp. 348-355.

[15] Gamha Y., Bennacer N., Ben Romdhane L., Vidal-Naquet G., Ayeb B., A Statechart- based model for the semantic composition of web services, in: Proceedings of SERVICES'07, IEEE Computer Society, Washington, DC, 2007, pp. 49-56

[16] Li H., Wang H., Cui L., Automatic composition of web services based on rules and meta-services, in: Proceedings of CSCWD'07, IEEE, 2007, pp. 496-501

[17] BPEL. Business Process Execution Language for Web Services Version 1.1. 2007. Available at <<http://www.ibm.com/developerworks/library/specification/ws-bpel/>>. Accessed in June 1, 2009.

[18] Shin, D ; Lee, K.; Suba, T., Automated generation of composite web services based on functional semantics, in Journal of Web Semantics, vol. 7, pp 332-343. 2009.

[19] LU, J., MA, L., ZHANG, L., BRUNNER, J., WANG, C., PAN, Y. e YU, Y. 2007, Sor: a practical system for ontology storage, reasoning and search, In: VLDB '07: Proceedings of the 33rd international conference on Very large data bases, VLDB Endowment, ISBN 978-1-59593-649-3

[20] Sirin, E. – OWLSAPI – in <http://www.mindswap.org/2004/owl-s/api/> accessed in 03-15-2010

[21] Lloyd, C. M., Halsted, M. D. B., Nielsen, P., “CellML: its future, present and past”. Progress in Biophysics & Molecular Biology 85. pp 433-450. 2004.

# A Stochastic Model for Optimizing the Patching Time of Software Bugs

Yong Wang\*  
Department of Computer Science  
Texas A&M University  
College Station, Texas 77843  
johnwangy@gmail.com

Dianxiang Xu  
National Center for the Protection of  
the Financial Infrastructure  
Dakota State University  
Madison, SD 57042

William M Lively  
Department of Computer Science  
Texas A&M University  
College Station, Texas 77843

Dick B. Simmons  
Department of Computer Science  
Texas A&M University  
College Station, Texas 77843

## Abstract

New bugs and vulnerabilities are discovered and reported from time to time even after software products are released. One of the common ways to handle these bugs is to patch the software. In this paper, we propose a stochastic model for optimizing the patching time for software bugs and vulnerabilities. The optimal patching time can be computed in the patching script development and operational costs in fix. We present one case study using the Nimda worm vulnerability in Microsoft Internet Information Services web server. The study indicates that the patch application is later than the optimal fix time..

## 1. Introduction

Quality assurance of software has been a challenge because modern software is inherently complex. New bugs and vulnerabilities are discovered and reported from time to time even after software products are released. One of the common ways to handle these bugs is to patch the software [6],[16]. There are different types of patches. For example, a patch for proprietary software is often distributed in binary executable. Such a patch modifies existing executable either by including the fixes into the binary file or by completely replacing it. Patches can also be distributed through source code modifications. They consist of textual differences between source codes. In this case, users are expected to compile the new code. Such patches are often observed in open source software systems. In most cases, “patch” means a small and fast fix. Large fixes may use a different name. For example, Microsoft calls bulky patches or patches that change a program to a large degree as “service packs” or “software updates”. Patches in this paper include all of these types.

Because patches are required to develop and test, there is a time lag between the defect reports and the deployment of patches. When the defects make the software vulnerable, early disclosure of defects or late deployment of security patches can lead hackers to take advantage of vulnerability knowledge. The major cost of patching comes from two major sources: script development and testing and interrupted service when applied to an existing system in operation. Therefore optimizing patching time for software systems can achieve the best economic benefit when the software systems are in service.

In this paper, we propose a stochastic model for optimizing the patching time for software bugs and vulnerabilities. The optimal patching time can be computed in the patching script development and operational costs in fix. We present two case studies using the Nimda worm vulnerability in Microsoft Internet Information Services web server. The study indicates that the patch application is later than their optimal fix time.

The rest of this paper is organized as follows. Section 2 presents the model. Section 3 describes one case study. Section 4 reviewer related work. Section 5 concludes this paper.

## 2. The model

After a bug or vulnerability is reported, it is often desirable to get it fixed. In the following, we discuss the optimal time for fixing it. Let us assume the time of software system failure due to software defects with a distribution function  $F(t)$ , where  $F(t) = P\{T \leq t\}$ .  $P\{T \leq t\}$  is the probability of the software system failure within time  $t$  due to software defects. The cost to fix the vulnerability or bug by patching is  $C_r$  [13]. Additional cost

will be  $C_f$  if the software is patched during service and the systems can not provide the intended service. Let  $W_i = \{T, t_0\}$  where  $t_0$  is the time to fix vulnerabilities and bugs,  $\{w_i\}$  is software failure time to provide service and  $\{w_i\}$  has identical independent distribution. Let  $N_t =$  number of bugs or vulnerabilities discovered in  $(0, t]$ . If  $W_i$  has an exponential distribution,  $N_t$  is the renewal random variable and has a Poisson distribution, thus,  $\{N_t; t \geq 0\}$  is a renewal process.

Let system maintenance cost be associated with the  $i$ th week after the software is released with value of  $C_i$ ,  $C_i$  includes the cost from the software patch development (and testing cost) and the cost from the interrupted services for patching purpose. Thus,  $\{C_1, C_2, \dots\}$  have an identical independent distribution.

$$\lim_{n \rightarrow \infty} \frac{1}{t} E\left[\sum_{n=1}^{N_t} C_n\right] = \frac{E[C_i]}{E[W_i]}$$

Or

$$\lim_{t \rightarrow \infty} \frac{1}{t} E[C_1 + C_2 + \dots + C_{N_t}]$$

is an average expected cost incurred in  $(0, t]$

When we fix time:  $t_0$

Let  $Z(t_0)$  denote average cost for  $t_0$

$$Z(t_0) = \frac{\text{expected\_cost } t}{\text{expected\_time\_length}} = \frac{C_r + C_f P\{T \leq t_0\}}{E[\min(T, t_0)]}$$

where expected cost is a cost for patching the vulnerabilities or software bugs in software systems, expected\_time\_length is the expected time for patching the software.

Let  $W = \min\{T, t_0\}$

$$\hat{F}(t) = P\{W \leq t\}$$

Thus,

$$\hat{F}(t) = F(t) \text{ if } t < t_0 \text{ or } \hat{F}(t) = 1 \text{ if } t \geq t_0$$

When  $x \geq 0$ ,

$$E[X] = \int_0^{\infty} x f(x) d_x = \int_0^{\infty} [1 - F(x)] d_x$$

$$E[\min(T, t_0)] = \int_0^{\infty} [1 - \hat{F}(t)] d_t$$

$$= \int_0^{t_0} [1 - \hat{F}(t)] dt + \int_{t_0}^{\infty} [1 - \hat{F}(t)] d_t$$

$$= \int_0^{t_0} [1 - F(t)] d_t$$

$$= \int_0^{t_0} \bar{F}(t) d_t$$

Thus,

$$Z(t_0) = \frac{\text{expected\_cost } t}{\text{expected\_time\_length}} = \frac{C_r + C_f P\{T \leq t_0\}}{E[\min(T, t_0)]} = \frac{C_r + C_f F(t_0)}{\int_0^{t_0} [1 - F(t)] d_t}$$

To minimize the cost, we need to take  $dZ(t_0)/dt_0 = 0$ , so,

$$\frac{d}{dt_0} Z(t_0) = \frac{\left[ \int_0^{t_0} [1 - F(t)] C_f f(t_0) - [C_r + C_f F(t_0)] [1 - F(t_0)] \right]}{\left[ \int_0^{t_0} [1 - F(t)] d_t \right]^2}$$

$$\frac{d}{dt_0} Z(t_0) = 0 ;$$

$$\left[ \int_0^{t_0} [1 - F(t)] d_t C_f f(t_0) - [C_r + C_f F(t_0)] [1 - F(t_0)] \right]$$

$$\left[ \int_0^{t_0} [1 - F(t)] d_t \right] C_f \frac{f(t_0)}{1 - F(t_0)} = C_r + C_f F(t_0)$$

For any F, its hazard rate is defined as

$$h(t) = \frac{f(t)}{1 - F(t)}$$

$$h(t) = \lim_{\Delta t \rightarrow 0} \frac{P\{t < T \leq t + \Delta t\}}{P\{T > t\}}$$

$$= \lim_{\Delta t \rightarrow 0} P\{t < T < t + \Delta t | T > t\}$$

For exponential and Weibull distribution, we can calculate the hazard rate:

For exponential distribution,

$$h(t) = \frac{1 - e^{-\lambda \Delta t}}{1 - (1 - e^{-\lambda t})} = \frac{1 - e^{-\lambda \Delta t}}{e^{-\lambda t}} = e^{\lambda t} - e^{\lambda(t - \Delta t)}$$

$$h(t) = \frac{f(t)}{1 - F(t)} = \frac{\lambda * e^{-\lambda t}}{e^{-\lambda t}} = \lambda$$

It is memoryless. Hazard rate is a constant

For Weibull distribution, if  $\alpha > 1$ , we have an increasing failure rate (IFR):

$$h(t) = \frac{f(t)}{1 - F(t)} = \frac{-e^{-(\lambda t)^\alpha} * (-\alpha)(\lambda t)^{\alpha-1}}{e^{-(\lambda t)^\alpha}}$$

$$= \alpha \lambda (\lambda t)^{\alpha-1}$$

If  $\alpha = 1$ , it is exponential distribution.

From  $\frac{d}{dt_0} Z(t_0) = 0$ ,

We can have:

$$\left[ \int_0^{t_0} [1 - F(t)] d_t \right] * h(t) = \frac{C_r}{C_f} + 1$$

$$\left[ \int_0^{t_0} [1 - F(t)] d_t \right] * h(t) - F(t) + 1 = \frac{C_r}{C_f} + 1$$

The optimal  $t_0$  for IFR distribution is such that:

$$\left[ \int_0^{t_0} [1 - F(t)] d_t \right] * h(t) + [1 - F(t)] = 1 + \frac{C_r}{C_f}$$

In the above equation,  $h(t)$  is hazard rate from the software failure either from the software bugs or vulnerabilities.  $C_r$  is a cost for patching development for the vulnerabilities or bugs.  $C_f$  is a cost associated with software system failure while the software systems can not provide the intended services.  $F(t)$  is the probability for the system failure in the software systems from the vulnerabilities or bugs.

### 3. Case study

In general, patch is applied to software bugs or vulnerabilities. In the following, we will apply the proposed model to security vulnerabilities as an application example. First, we will present a case study on software vulnerabilities. From the software systems in operation until the vulnerability report, we call blind safety period. During this period of time, hackers do not exploit the vulnerability. Our subject system is a web-based application using IIS web server. Nimda is a self-executing virus attack IIS web server and Microsoft outlook [10]. The worm attacks Microsoft outlook users using blank message with an attachment in readme.exe file. According to the websites from the bugtraq in <http://www.securityfocus.com> and from <http://www.secunia.com>, the Nimda worm can attack IIS web server 4.0/5.0 version by reading documents outside of the web root. The vulnerability possibly allows executing the arbitrary commands by using malformed URLs that contains Unicode encoded characters. This vulnerability is also called web server folder traversal vulnerability. The vulnerability was reported on October 17, 2000 in CVE-2000-0884. The patch for the basic Nimda worm was released on August 15, 2001. The time span from the vulnerability report to the patching fix takes approximately 42 weeks. Patch development and testing

are the major initial cost. The past experience for a similar vulnerability patching indicates that the actual probability mass function for security failed time is as follows in the Table 1. For simplicity, we assume that security failure always occurs before the end of the week. The patching can be made at the end of week without any interruption to the operation schedule. In this case, patching cost is basic patching development cost of 3,000 dollars. If the security failed before patching, an additional cost of \$1,000 is incurred because the software systems can't provide intended services. One patch normally can fix one or more reported vulnerabilities. What is the optimal patching time?

According to the (4) model we propose, we can calculate the optimal  $z(k)$  value to identify which week is the best patching time in Table 2. From the analysis, we know the week 34 is the best time to patch the software system because average cost  $z(k)$  value in week 34 is the lowest based on our proposed model. Therefore, ideally, we need to patch software systems to prevent further economic losses from security breaches earlier than the patch release.

**Table 1. Mass function of security failure distribution.**

Week	Probability	Week	Probability
1	0.00	22	0.01
2	0.01	23	0.02
3	0.01	24	0.01
.....	.....	.....	.....
13	0.01	34	0.06
14	0.01	35	0.06
15	0.01	36	0.10
16	0.01	37	0.07
17	0.01	38	0.10
18	0.01	39	0.08
19	0.01	40	0.04
20	0.01	41	0.025
21	0.01	42	0.05

### 4. Related work

Arbaugh et al. [1] defined a vulnerability life-beneath cycle model in seven stages sequentially. The seven

**Table 2. Optimal patching time for the vulnerability in a software system.**

week	$f_k$	$F_k$	$C_r+C_f$ $F_k$	$F_k^-$	$\sum_{i=0}^{k-1} F_k^-$	$Z(k)$
1	0.0 0	0.0 0	3000	1.00	1.00	3000.0 0
2	0.0 1	0.0 1	3010	0.99	1.99	1512.5 6
3	0.0 1	0.0 2	3020	0.98	2.97	1016.8 4
4	0.0 1	0.0 3	3030	0.97	3.94	769.04
5	0.0 1	0.0 4	3040	0.96	4.90	620.41
.....	...	...	.....	...	.....	.....
31	0.0 2	0.3 4	3340	0.66	26.21	127.43
32	0.0 4	0.3 8	3380	0.62	26.83	126.42
33	0.0 8	0.4 6	3460	0.54	27.37	125.98
34	0.0 6	0.5 2	3250	0.48	28.85	122.01
35	0.0 6	0.5 8	3580	0.42	29.27	122.31
36	0.1 0	0.6 8	3680	0.32	29.59	124.37
37	0.0 7	0.7 5	3750	0.25	29.84	125.67
38	0.1 0	0.8 5	3850	0.15	29.99	128.38
39	0.0 8	0.9 3	3930	0.13	30.12	130.47
40	0.0 4	0.9 7	3970	0.03	30.15	131.67
41	0.0 25	0.9 95	3995	0.00 5	30.155	132.45
42	0.0 05	1.0 0	4000	0.00	17.29	132.65

stages are birth, discovery, disclosure, publicity, scripting, correction, and death. Koetzle et al. [8] specified the time period of increased security risk between disclosure and correction stages as time period in “days of risk”. So, we can calculate days of risk by subtracting the date a vulnerability was public reported from the day the vendors released the fix patch. Once a vulnerability is fixed in correction stage, the vulnerability moves to death stage.

Generally, for software systems, patching release is a quick fix for the correction stage.

Different defect sources in software bugs and vulnerabilities may take different time spans to fix the problems. Kim and Whitehead [6] studied how long it takes to fix bugs in software systems in two software related projects. They found that bug fix time ranges from 100 days to 300 days. The median bug fix days take 200 days in software systems and PostgreSQL database systems [6].

For software vulnerabilities, different vendors may take different time span to fix. Microsoft spent several months to develop security fix in 2003. For many years, Microsoft has been criticized for taking longer time to issue patches [9]. According to the report in digital bond, US-CERT vulnerabilities take 3-6 months from disclosure to patch fix [3][5]. Their research showed that disclosure can pressure vendors to issue patches quickly and improve their software quality. The disclosure also can educate computer programmers and network administrators on new vulnerabilities. Therefore, the overall security quality for software systems may be improved. However, all these research only show passive response from software vendors to vulnerability disclosures. Currently once the software vulnerabilities are reported, a fix patch normally is a quick solution to fix the problems. In this paper, we will present an optimal patching time for software security vulnerabilities and bugs. This is a positive response for software system development companies to fix the software vulnerabilities and bugs after the software systems are released and the defects are reported.

Stochastic renewal process has been widely used in many applications domains, ranging from manufacturing systems to different computer systems. For example, Rupe and Kuo [12] applied renewal process to a manufacturing system which consists of multiple machines. Each machine has multiple parts which may have different kinds of failures. They derived a performance model using renewal process. Stochastic renewal process also has been used in computer networks and systems [11]. In a renewal process, inter-arrival times for all arrivals have independently and identically distributed. A renewal process is a general case of a Poisson process. Zhu et al. [17] assumed that soft real-time tasks arrive in renewal process. These tasks have random service requirements and deadlines in a real-time system. They used queuing theory to computer the long run percentage of tasks which miss the deadline. Belzunce et al. [2] compared expected failure times for several replacement policies. They compared the expected failure times of a block replacement policy with a renewal process with no planned replacements. In this paper, we plan to apply the

renewal process to software systems to estimate optimal software patching policy for software vulnerabilities and bugs after software systems are released.

## 5. Conclusions

In this paper, we developed stochastic model to optimally estimate the fix time of software vulnerabilities and bugs. The probability of software security failure is a statistical estimate. From the case study, we found the date of actual patch release from the software owner is not optimal time for vulnerability fix. There are several reasons for this. One of important reasons may be that software vulnerabilities and bug are in low priority. The other reason may be that software vendors do not realize the potential risk that software system defects may cause. Late fix of vulnerabilities may cause software systems to fail in providing reliable services. It could cause a significant cost when the software systems are not available because of security failures. Active planning to fix the software vulnerabilities can make the software systems more reliable in providing intended services.

## Acknowledgement

The authors wish to thank Dr. Richard Feldman at Texas A&M University for advising stochastic renewal model.

## References

- [1] W., Arbaugh, W. Fithen, and J. Mchugh, "Windows of vulnerability: A case study analysis". *Computer*, Vol. 33, no. 12, 2000, pp.52-59.
- [2] F. Belzunce, E. M. Oretega, J. M. Ruiz, "Comparison of expected failure times for several replacement policies" *IEEE Transactions on Reliability*, Vol. 55, No. 3, 2006, pp. 490-495.
- [3] CERT Coordination Center Statistics, <http://www.cert.org/stats/cert-stats.html>, 20006.
- [4] Exforsys Inc., "Bug life cycle & guidelines". [http://www.exforsys.com/tutorials/testing/bug\\_life\\_cycle\\_guidelines.html](http://www.exforsys.com/tutorials/testing/bug_life_cycle_guidelines.html), 2006.
- [5] Franz, M, "US-CERT Vulnerability disclosure". <http://www.digitalbond.com/index/php/2006/05/16/us-cert-livedata-iccp-vulnerability-note/>, 2006.
- [6] Gardler, R., "OSS Watch – What is a software patch?" <Http://www.oss-watch.ac.uk/resources/softwarepatch.xml>, 2009.
- [7] .S. H. Kim and E. J. Whitehead, " How Long Did It Take to Fix Bugs?" *Proceeding of International Workshop on Mining Software Repositories*, Shanghai, China, 2006.
- [8] L. Koetzle, C. Rustein, N. Lambert, and S. Wenninger, 2004. "Is Linux More Secure than Windows?" *Forrester Research*, [www.forrester.com/Research/Document/Excerpt/0,7211,3941,00.html](http://www.forrester.com/Research/Document/Excerpt/0,7211,3941,00.html), 2004.
- [9] B. Krebs, " Security fix", [http://voices.washingtonpost/security/fix/2006/01/a\\_time\\_to\\_patch](http://voices.washingtonpost/security/fix/2006/01/a_time_to_patch), 2006.
- [10] K. Pousen., "Nimda Worm Hits Net". <http://www.securityfocus.com/news/253>, 2001
- [11] Robertazzi, T. G., *Computer networks and systems: Queuing theory and performance evaluation*, 3<sup>rd</sup> edition, Springer-Verlag New York, Inc, New York, NY 10010, USA, 2000.
- [12] J. Rupe and W. Kuo "Performability of systems based on renewal process models". *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, VOL. 28, No. 5, pp. 691-698.
- [13] S. M. Ross *Stochastic Processes*, 2<sup>nd</sup> Edition. John Wiley & Sons, Inc, 1996.
- [14] Scalet et al., "ISO/IEC 9126 and 14598 integration aspects: A Brazilian Viewpoint". *The Second World Congress on Software Quality*, Yokohama, Japan. 2000.
- [15] A. Stone, "Software Flaws: To Tell or Not to Tell?" *IEEE Software* (January/February), 2003. pp. 70-73.
- [16] Wikipedia, Patch (computing), [http://en.wikipedia.org/wiki/Patch\\_\(computing\)](http://en.wikipedia.org/wiki/Patch_(computing)).
- [17] H. F. Zhu, J. P. Hansen, J. P. Lehoczy and R. Rajkumar, "Optimal partitioning for quantized EDF scheduling," *Proceedings of the 23<sup>rd</sup> IEEE Real-Time Systems Symposium*, 2002.

# Do More People Make the Code More Defect Prone?: Social Network Analysis in OSS Projects

Salifu Alhassan<sup>1</sup>, Bora Çağlayan<sup>2</sup>, Ayşe Bener<sup>3</sup>  
Boğaziçi University  
{salifu.alhassan<sup>1</sup>, bora.caglayan<sup>2</sup>, bener<sup>3</sup>} @ boun.edu.tr

*Abstract- A software quality model consists of three pillars: 3Ps (Product, Process, and People). Many software quality models in the literature have so far taken process and product related attributes into consideration, but there are very few studies that model people. In Open Source Software (OSS) development projects developers, and their interactions with each other, and the software product are even more important than commercial projects. In open source software loosely defined processes usually increase the significance of people. In this research we are motivated to explore the social network of people in OSS projects. For this purpose we extracted the social network metrics of developers in four large-scale open source software in order to examine their relation to software quality. First, we analyzed the effect of social network metrics in predicting post release defects. Second, we explored the effect of social networks on defect proneness of open source software. Our experimental results showed that social network metrics do not have a distinct impact over and above static code and churn metrics in predicting defects. On the other hand, our results revealed that there is a positive correlation between social network complexity and defect proneness.*

## INTRODUCTION

Open Source Software (OSS) is rapidly gaining popularity in the software industry. Many major industrial players continue to invest heavily in OSS Projects [1]. Key players like IBM and HP are developing open source consultation services. New companies like Google, Yahoo and Facebook have built massive scalable infrastructures by primarily using OSS technologies.

So far, many defect prediction models proposed in the literature use metrics derived from the source code (static code attributes), and/ or history (churn) metrics. However, software development involves product, process and people. Modeling product and processes are easier than modeling people (mainly developers and testers). Most of the product and process attributes can be automatically extracted either from the source code or from the existing documentation. One of the emerging research areas to model people in software development is social network analysis [3, 4, 5].

People contributing to the source code and their interactions make up the social network of a software system [6, 7]. Due to particular characteristics of OSS projects, people interactions may have an important effect on the defect proneness of software.

The following characteristics of OSS can make social network analysis of developers more important than commercial projects. First of all management in OSS is not as centralized as in closed source projects. In some OSS projects,

a group of core developers, normally founding members of the project, has control over the project [9]. In other OSS projects, communication or authority among developers is coordinated horizontally [9]. Governance in OSS tends to be less formal than in closed source software. The absence of central governance leads to developers depending on the source code and documentation for coordination. It is claimed that this results in OSS having code with higher quality [10]. Informal governance also improves the overall product quality by removing the pressure of deadlines from contributors [10].

We believe that from a project development life cycle standpoint OSS has different characteristics. Therefore, the research findings for closed loop commercial products may not hold true for OSS products. In this research we attempt to understand the effect of social interactions of developers on predicting defects as well as on the overall defect proneness of software. Hence we ask the following research questions:

- **How do social metrics affect the prediction performance of defect predictors in OSS?**
- **How do complex social interactions affect the defect proneness of the code in OSS?**

The paper is organized as follows: Section 2 summarizes the related work on defect prediction as well as the use of social networks in software development projects. Section 3 discusses our methodology, the details of the empirical work, the datasets used, data extraction steps, and experimental setup. Section 4 talks about the results of our experiments related to the two research questions. Section 5 discusses threats to validity of our experiments and how we overcame these threats. Section 6 summarizes our research and presents possible future directions.

## RELATED WORK

The concept of forming developer network from repositories has been studied in previous research [11, 12, 13]. However, the developer (social) networks constructed in those research are mostly used for descriptive analysis rather than for defect prediction. Wolf et al. [6] used social network metrics, built from developer communications, to predict build failures in the Jazz Dataset. Meneely et al [7] used logistic regression methods to examine the collaborative social structures that exist between developers together with other metrics derived from code churn information. Meneely conducted a case study and found that a significant correlation exists between file-based developer network models and defects.

Churn metrics are another metric set based on the changes done on software modules. Code churn metrics have been used to form successful predictive models in previous work [2, 14].

Process metrics related to organizational structure have also been used to investigate the relationship between human factor, organizational structures and software defects [15]. Although process metrics have been shown to provide high prediction performance, they are difficult to collect in OSS projects or in small companies. In OSS projects, a well defined process tends to be nonexistent or not formally documented [15]. In small companies, the number of developers tends to be limited hence the depth of the organizational tree may not provide a significant measure for the company [15].

In previous work Weyuker used social metrics to predict defects on a large scale AT&T project but they found no significant improvement on the performance of predictors [8]. In other research [10, 16], social network metrics have been used to determine OSS governance types. They estimated project structure based on collaborations on software. Governance varies depending on the project structure having a more uniform collaboration network or not.

Weyuker et al. [8] conducted a study to find out the effect the number of developers maintaining a file has on the file's defect proneness in a large-scale closed source software project. In their work only the numbers of developers were taken into consideration and social network of developers was not used. Based on the empirical results they concluded that the number of developers has no significance influence on the defect proneness.

## METHODOLOGY

### A. Datasets

In our research, we extracted metrics from the repositories of four OSS projects.

**Linux Kernel** is an Open Source operating system that is used in a variety of platforms ranging from mobile phones to super computers. The Linux kernel repository has been openly available since the first release in September 1991 [17].

**Android** is a new Open Source operating system for mobile devices which provides a base operating system, a middle layer for applications and a java-based Software Development Kit (SDK) [18].

**Perl** is an Open Source interpreted programming language that is optimized for extracting information from text files and generating reports based on the extracted content. Although Perl is used in a variety of tasks, it is especially used for tasks that run on the Web [20].

**VLC** is an Open Source multimedia player, encoder and streamer that supports various audio and video formats and codec [19]. VLC was started as a student project in 1996. However in 1998, VLC was completely rewritten and it became an Open Source Project.

### B. Data Collection

The source code for the above projects are all managed by the Git revision control system. For the purpose of this research, "2.6.28-stable kernel tree" repository of Linux Kernel, "android-2.6.27" repository of Android, "Perl5 Master Repository" repository of Perl and the "Main Development Tree" repository of VLC were cloned and analyzed. Some important features of the repositories are given in Table 1.

TABLE 1. Some Features of the Software Repositories used

Project	Releases	Commits	Files	Editors	Age (Days)
Linux Kernel	148	120,559	15,252	4,813	1,327
Android	130	110,378	14,716	4,487	1,210
Perl	54	33,418	189	974	5288
VLC	10	35,177	1248	252	382

For each project, we extracted static, history and social metrics at the file level. Static code and history metrics are extracted in order to compare their effect with that of social metrics. Defect data was also extracted for each project.

History metrics used in our research are given in Table 2.

TABLE 2. History Metrics used

<b>commit span</b>	Number of Days between the first and last commits
<b>commit frequency</b>	Total number of commits divided by "commit Span"
<b>no_of_commits</b>	Total Number of commits
<b>unique_editors</b>	Number of distinct editors
<b>unique_committers</b>	Number of distinct committers
<b>lines_added</b>	Sum, Average, Max and Min lines added during commit
<b>lines_deleted</b>	Sum, Average, Max and Min lines deleted during commit

The social network used in the research is based on the definitions found in the work of Meneely et. al. [7]. We created a collaborative graph by using all the authors who edited any given file as nodes. An edge is drawn between two nodes if the two authors representing the respective nodes edited the same file during the same release. The social metrics used in our research is given in Table 3.

TABLE 3. Social metrics used

<b>distinct_authors</b>	Total number of Nodes (authors)
<b>betweenness</b>	Sum, Average, Max and Min of Betweenness
<b>closeness</b>	Sum, Average, Max and Min of Closeness
<b>degree</b>	Sum, Average, Max and Min of Degree
<b>total_editors</b>	Number of edits
<b>edits</b>	Sum, Average, Max and Min of edits on file
<b>weighted_edges</b>	Number of weighted edges. Weights are number times the two editors edited the same file.

Degree is a connectivity metric that measures total number of nodes that are incident on a particular node [30]. Degree gives an indication of how connected a node is. A node without any edges has zero degree and it is said to be a disconnected node.

Closeness is a centrality measure which gives an indication of how a node is indirectly connected to other nodes [30]. The closeness of a node  $v$  is defined as the average distance from it to any other node in the graph that is connected to it. Closeness can be calculated from formula (1) where  $d_G(v, t)$  is the number of edges between  $v$  and  $t$  and  $|V(G, v)|$  is the number of nodes directly connected to  $v$ .

$$D_c(v) = \left( \frac{1}{|V(G, v)|} \right) \sum_{t \in G} d_G(v, t) \quad (1)$$

Betweenness is a centrality measure which determines a node's centrality by looking at the number of paths that the node is a part of [30]. Betweenness of a node  $v$  is defined as the number of geodesic paths that include the node divided by the total number of geodesic paths and it can be calculated from formula (2).

$$B_c(v) = \sum_{s, t, v \in G, s \neq t, t \neq v} \frac{\sigma_{st}(v)}{\sigma_{st}} \quad (2)$$

Where  $\sigma_{st}(v)$  is the number of geodesic paths from  $s$  to  $t$  passing through  $v$  and  $\sigma$  is the total number of geodesic paths.

Each of the above mentioned metrics captures a different aspect of a ~~single~~ developer's position in the network of developers. A developer's degree indicates how many peers he directly worked with. A disconnected developer is one who worked alone on files. A central developer is one with a high betweenness. This is because a central developer modifies many files together with other developers hence he/she will be on more geodesic paths. Developers with low closeness have a low social distance implying that they are well-known.

It is important to note that, at the initial stages of development developers tend to work solo on files and hence the collaborative graph has very few edges and the metrics extracted from such a graph are not useful in predicting defects.

### C. Automatic Defect-Labeling

Manually labeling files in a repository as defective or not is a time and resource consuming one. In fact, many companies do not implement defect prediction models because of the difficulty of the defect labeling task. In our research, we used Python scripts to automatically label the files in each software repository as defective or not. In order to perform the automatic labeling of defects, a snapshot version had to be chosen for each project. This snapshot version is used as a reference point and static code metrics are extracted from it. All commits relating to all releases up to and including the snapshot release are used for extracting the history and social metrics. The commits, related to versions that were released

after the snapshot version, are used for defect labeling. The commit messages were checked in order to label them as defective or not by using a bag of words (like bug, fixed...). If a file had a commit associated with a defect fix after our snapshot release, the file was labeled as defective and safe otherwise. The snapshot versions selected for each project together with number of files and defective file counts are given in Table 4.

TABLE 4. Snapshot Version and defects

Project	Version	files	Defective Files	Percentage
Kernel	v2.6.27	14801	816	6%
Android	v2.6.26	12981	831	6%
Perl	perl-6.0	125	77	62%
VLC	0.9.0	936	367	39%

### D. Defect Prediction Model

We have been using a learning based model for many years for defect prediction [2, 24, 25]. Our model uses machine learning algorithms to learn, classify defects from past projects and predict defects in future projects. We have used Naïve Bayes as the learning algorithm since it takes signals from multiple sources. It is also a simple statistical estimator which has proven to be effective in the field of software defect prediction [21, 23, 24, 25]. In a benchmark study by Lessman et al. Naïve Bayes was compared to numerous complicated machine learners and it was found to perform equally as the more complicated learners [26]. The defect prediction model that we have used can be seen in Figure 1.

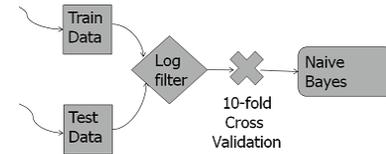


FIGURE 1, Defect Prediction Model

### E. Performance Measures

We use two different performance measures to evaluate the classification accuracy of our prediction model. These are probability of false alarms (pf), and probability of detection (pd) [27]. These measures can be analyzed as a typical ROC curve of signal detection theory that is used for comparing different predictors. A ROC curve is represented in Figure 2. In the ideal scenario, hit rate, pd, should be 1, i.e. the predictor should detect all defects. Moreover, false alarms, pf, should be 0, meaning that the predictor should never detect a defect-free module as defective. In general, an increase in pd causes an increase in pf rates, since the model triggers more often to achieve the ideal case [26]. To compute pd, and pf, the confusion matrix in Table 5 is needed. From the confusion matrix, the performance measures are calculated using Equations 3 and 4.

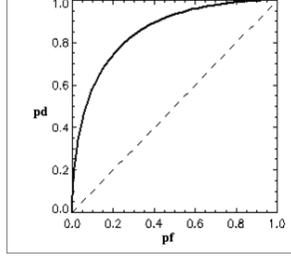


FIGURE 2: A Simple ROC (Relative Operating Characteristic) Curve

TABLE 5 - Confusion Matrix

Predicted As	Actual	
	defected	Defect Free
defected	A	B
defect free	C	D

$$pd = \frac{A}{A+C} \quad (3) \quad pf = \frac{B}{B+D} \quad (4)$$

#### F. Model's Benefit Estimation

We have conducted a cost-benefit analysis of our results based on the work of Arisholm et al. [1]. Equations 5, 6 and 7 determine the savings derived from our predictive model in terms of effort spent. The formula contains the files we marked correctly as defective as well as our false alarms. Therefore it includes the pf rate in it. The lower the pf rate the higher the benefit would be. In other words, the real value comes from the increase in "D" and decrease in "B" which is the definition of pf.

$$pfa = \frac{A+B}{A+B+C+D} \quad (5) \quad benefit(\%) = (pd - \frac{A+B}{A+B+C+D})(6)$$

#### G. Experiments

We conducted three sets of experiments to answer our research questions. In the first two sets of experiments we tackled our first question, namely the value of using social network metrics in defect prediction. In the first set of experiments, each metric set was used separately as input to the model. In the second set of experiments, we used all the possible combination of metric sets as input. We therefore have the following combinations: static-history, static-social, history-social and static-history-social (all). In the third experiment we tackled our second research question namely the effect of complex social network of modules on defect proneness. For this part we divided the data into two clusters based on euclidian distances of social network metrics. For this reason, we used K-means clustering with k=2 since we wanted to find the centers of natural clusters in the data. We chose k to be 2 to check if we can find 2 clusters where one of them is significantly more complex than the other one. We compared the defect proneness of the two clusters afterwards. The Open

Source Machine Learning Tool WEKA [28] was used to perform all of our experiments.

## EXPERIMENTAL RESULTS

### A. Usage of Social Network Metrics in Defect Prediction

Regarding our first research question we examined the pd and pf rates when using only social network metrics, combining social network metrics with static code metrics and history (churn) metrics as well as combinations of different metrics sets.

### Results of Using Individual Metric Sets:

In the first set of experiments we compared the predictive power of each of the metric sets on each project and the results are displayed in Table 6. It can be observed that the social metrics-based models give the lowest pf value in all the four projects that we used in our experiments. However, pd values are also lower. VLC and Perl projects have the best pd values when history metrics are used as input to the model whilst Android and Linux kernel projects have the best pd values when static metrics are used.

TABLE 6 - Results of experiment 1

Metric	ANDROID		PERL		LINUX KERNEL		VLC	
	pd	pf	pd	pf	pd	pf	pd	pf
Static	76.17	50.37	61.04	16.67	75.74	46.86	80.93	51.49
history	64.98	27.54	89.61	18.75	63.85	24.78	83.65	48.68
social	52.83	19.50	81.82	12.50	56.00	18.78	69.21	29.70

### Results of Using Metric Combinations

The second set of experiments compared the performance of models built upon the various combinations of the three metric types. The results of this experiment set are as displayed in Table 7. In this experiment too, it can be observed that the minimum pf value is achieved when the metric set combination includes social metrics. With the exception of Perl, the best pf value is obtained, in all projects, when a combination of history and social metrics are used as input to the model. In the case of Perl, the best pf value is obtained when all metric sets are used.

TABLE 7 - Results of Experiment 2

Metrics Combination	ANDROID		PERL		LINUX KERNEL		VLC	
	pd	pf	pd	pf	pd	pf	pd	pf
static-history	76.53	46.91	64.94	12.50	75.25	43.74	82.02	44.29
static-social	73.29	42.04	79.22	12.50	72.92	38.42	79.29	40.95
history-social	59.69	23.48	87.01	12.50	59.80	22.14	73.30	35.85
all	73.77	39.95	83.12	10.42	72.92	36.58	78.47	38.66

### B. Cost Benefit Analysis

We conducted cost benefit analysis to show how our proposed model will be of benefit to developers and project managers. The cost-benefit analysis was performed as described in section 3.6 above. The details of the cost-benefit analysis for each metric type or combination can be seen in Table 8. We can see that for 2 of the 4 projects using social and history metrics together for defect prediction gave the best results while for the remaining ones using only history metrics gave the best results.

TABLE 8 - Derived Benefit from Model

	KERNEL	ANDROID	PERL	VLC
static	27.29%	24.15%	17.04%	17.89%
history	36.92%	35.05%	27.21%	21.26%
social	35.18%	31.20%	26.62%	24.02%
static-history	29.77%	27.73%	20.14%	22.94%
static-social	32.60%	29.24%	25.62%	23.31%
history-social	35.59%	33.89%	28.61%	22.76%
all	34.33%	31.65%	27.92%	24.20%

### C. Social Network Relation with Defect Proneness

In the third experiment, we conducted clustering as explained in section 3.7 on the social network metrics of each project. It was observed that one of the clusters had a centroid with higher values for each social network metric. The relatively more complex cluster was found to be significantly more defect prone than the simpler cluster for each project that was observed. The defect proneness of each cluster is given in Table 9. Cluster A is the cluster with lower network complexity centroids whilst Cluster B is the cluster with higher network complexity centroid. In Table 8, we can see that the cluster with higher social network complexity was 2 to 3 times more defect prone than the cluster with simple social network complexity.

TABLE 9 - Defect Proneness of Cluster A and Cluster B

Android	Non-Defective	Defective	Perl	Non-Defective	Defective
Cluster A	6393	211	Cluster A	46	18
Cluster B	5757	620	Cluster B	2	18
Linux	Non-Defective	Defective	VLC	Non-Defective	Defective
Cluster A	8165	221	Cluster A	339	85
Cluster B	5820	595	Cluster B	230	282

### THREATS TO VALIDITY

We would like to address the threats to the validity of our empirical work under four categories: 1) Internal validity, 2) external validity, 3) construct validity and 4) statistical validity.

Internal validity fundamentally questions to what extent the cause-effect relationship between dependent and independent variables hold. For addressing the threats to internal validity of our results, 10-fold cross-validation is used in the experiments to eliminate ordering effects in our datasets.

External validity, i.e. generalizability of results, addresses the extent to which the findings in a particular study are applicable outside the specifications of that study [24]. To ensure the generalizability of our results, we used multiple projects with different characteristics and compared the results of each project.

Construct validity (i.e. face validity) assures that we are measuring what we actually intended to measure. As performance evaluation of defect prediction we used pd and pf which are commonly used for evaluation of defect prediction models. Moreover, we conducted a cost/ benefit analysis to see how effective the social metrics are. Analysis of commit message patterns to find bugs was validated manually when possible.

To validate our results statistically we employed statistical significance tests. In order to form significance tests on our data, we used the Mann-Whitney U significance test on the pd and pf values of all the possible project pairs (Kernel-Android, Kernel-Perl, Kernel-VLC, Android-Perl, Android-VLC and Perl-VLC). We found that using social metrics alone or in combination with other metrics did not improve pd and pf values significantly within a confidence interval of 0.95.

### CONCLUSIONS

Most of the research on software in the last decade has focused on technical and quantitative factors rather than on human factors [29]. However, software is developed by people for the use of other people. An analysis that does not consider people would be an incomplete one. OSS projects rely more on people than processes. Therefore, social networks and people interaction may have more significant impact on the overall quality of the software product compared to a closed loop commercial software development project. In this research our motivation was to better understand the effect of social metrics on the performance of defect prediction in OSS as well as to explore the effect of people interactions in OSS projects on its defect proneness. We took four different datasets from well known OSS projects to conduct our experiments.

In regards to our research question, **“How do social metrics affect the prediction performance of defect predictors in OSS?”**, our experimental results showed that social network metrics alone do not have significant impact on the prediction performance of the model. Although social metrics give us the lowest false alarm (pf) rates, the detection rates are also very low. Therefore, we can recommend that social network metrics can be used as a supplement to other metrics when managers would like to lower the inspection time and cost. This may be important especially for OSS projects since the users (customers) of OSS projects and/ or components would like to have a reliable product to integrate to their environment.

In regards to our second research question, **“How do complex social interactions affect the defect proneness of**

**the code in OSS?”**, our experimental results revealed that after clustering one of the clusters (from a two equal size clusters) emerged with significantly higher social complexity. The cluster with higher social complexity was consistently more defect prone than the simpler cluster in all projects. This shows that people interacting with the same software modules on a project lead to more defect prone code modules. This refutes the findings of the previous work on closed source software done by Weyuker et al. [8]. Reason for this can be attributed to informal processes and project-wide code ownership in Open Source projects. Maintaining a software module with a simpler social network needs a separation of responsibilities of developers into smaller groups when possible. This can be realized with a top-down governance of the projects.

Another contribution of our research is that we collected three different types of metrics data from four open source projects and shared them with the research community [31]. As a future work, we would like to extend this work by adding new people related metrics to enhance the People pillar of 3P (Product, Process, People).

#### ACKNOWLEDGEMENTS

This research is supported in part by TUBITAK project number EEEAG 108E014, and Turkish State Planning Organization (DPT) under the project number 2007K120610.

#### REFERENCES

- [1] Gregory DeKoenigsberg: How Successful Open Source Projects Work, and How and Why to Introduce Students to the Open Source World, 978-0-7695-3144-1 (2008) IEEE
- [2] Caglayan, B., Bener, A., and Koch, S. 2009. Merits of using repository metrics in defect prediction for open source projects. In Proceedings of the 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development (May 18 - 18, 2009). International Conference on Software Engineering. IEEE Computer Society, Washington, DC, 31-36.
- [3] Weissgerber, P., Pohl, M., Burch, M., Visual Data Mining in Software Archives to Detect How Developers Work Together, Proceedings of the Fourth International Workshop on Mining Software Repositories, p.9, May 20-26, 2007
- [4] Pohl, M. and Diehl, S. 2008. What dynamic network metrics can tell us about developer roles. In Proceedings of the 2008 international Workshop on Cooperative and Human Aspects of Software Engineering (Leipzig, Germany, May 13 - 13, 2008). CHASE '08. ACM, New York, NY, 81-84.
- [5] Weissgerber, P., Pohl, M., and Burch, M. 2007. Visual Data Mining in Software Archives to Detect How Developers Work Together. In Proceedings of the Fourth international Workshop on Mining Software Repositories (May 20 - 26, 2007). International Conference on Software Engineering. IEEE Computer Society, Washington, DC, 9.
- [6] Wolf, T., Schroter, A., Damian, D., and Nguyen, T. 2009. Predicting build failures using social network analysis on developer communication. In Proceedings of the 2009 IEEE 31st international Conference on Software Engineering (May 16 - 24, 2009). International Conference on Software Engineering. IEEE Computer Society, Washington, DC, 1-11.
- [7] Meneely, A., Williams, L., Snipes, W., and Osborne, J. 2008. Predicting failures with developer networks and social network analysis. In Proceedings of the 16th ACM SIGSOFT international Symposium on Foundations of Software Engineering (Atlanta, Georgia, November 09 - 14, 2008). SIGSOFT '08/FSE-16. ACM, New York, NY, 13-23.
- [8] Weyuker, E. J., Ostrand, T. J., and Bell, R. M. 2008. Do too many cooks spoil the broth? Using the number of developers to enhance defect prediction models. Empirical Softw. Engg. 13, 5 (Oct. 2008), 539-559
- [9] Ezeala, A., Kim, H., and Moore, L. A. 2008. Open source software development: expectations and experience from a small development project. In Proceedings of the 46th Annual Southeast Regional Conference on XX (Auburn, Alabama, March 28 - 29, 2008). ACM-SE 46. ACM, New York, NY, 243-246.
- [10] Capra, E., Francalanci, C., Merlo, F., An Empirical Study on the Relationship Between Software Design Quality, Development Effort and Governance in Open Source Projects, IEEE Transactions on Software Engineering, v.34 n.6, p.765-782, November 2008
- [11] Gonzales-Barahona, J. M., Lopez-Fernandez, L., and Robles, G., "Applying Social Network Analysis to the Information in CVS Repositories," in 2005 International Workshop on Mining Software Repositories, 2004.
- [12] Huang, S.-K. and Liu, K.-m., "Mining Version Histories to Verify the Learning Process of Legitimate Peripheral Participants," in 2005 International Workshop on Mining Software Repositories, 2005, pp. 1-5.
- [13] Ohira, M., Ohsugi, N., Ohoka, T., and Matsumoto, K.-i., "Accelerating Cross-project Knowledge Collaboration using Collaborative Filtering and Social Networks," in 2005 International Workshop on Mining Software Repositories, 2005, pp. 1-5.
- [14] Nagappan, N. and Ball, T. 2005. Use of relative code churn measures to predict system defect density. In Proceedings of the 27th international Conference on Software Engineering (St. Louis, MO, USA, May 15 - 21, 2005). ICSE '05. ACM, New York, NY, 284-292.
- [15] Layman, L., Kudrjavets, G., and Nagappan, N. 2008. Iterative identification of fault-prone binaries using in-process metrics. In Proceedings of the Second ACM-IEEE international Symposium on Empirical Software Engineering and Measurement (Kaiserslautern, Germany, October 09 - 10, 2008). ESEM '08. ACM, New York, NY, 206-212.
- [16] Nagappan, N., Murphy, B. and Basili, V.R. 2008. The Influence of Organizational Structure on Software Quality: An Empirical Case Study. In Proceedings of the International Conference on Software Engineering (ICSE), Leipzig, Germany, 521-530.
- [17] Linux kernel Website, <http://www.kernel.org>.
- [18] Android Website, <http://www.android.com>
- [19] VideoLAN Website, <http://www.videolan.org>.
- [20] Perl Website, <http://www.perl.org>
- [21] Turhan, B. and Bener, A. 2009. Analysis of Naive Bayes' Assumptions on Software Fault Data: An Empirical Study. Data and Knowledge Engineering Journal, Vol. 68, No. 2, 78-290.
- [22] Turhan, B., Menzies, T., Bener, A., Distefano, J. 2008. On the Relative Value of Cross-company and Within-Company Data for Defect Prediction. accepted for publication in Empirical Software Engineering Journal.
- [23] Menzies, T., Greenwald, J., Frank, A., Data mining static code attributes to learn defect predictors. Software Engineering, IEEE Transactions on, 33(1):2-13-, 2007.
- [24] Tosun, A., B. Turhan and A. Bener, "Ensemble of Software Defect Predictors: A Case Study", Proceedings of the 2nd International Symposium on Empirical Software Engineering and Measurement, Germany, October 2008.
- [25] Turhan, B., Bener, A., A multivariate analysis of static code attributes for defect prediction. In Quality Software, 2007. QSIC '07. Seventh International Conference on, pages 231-237-, 2007.
- [26] Lessmann, S., B. Baesens, C. Mues, and S. Pietsch, "Benchmarking classification models for software defect prediction: A proposed framework and novel findings", accepted for publication IEEE Transactions on Software Engineering, 2009.
- [27] Heeger, D. 1998. Signal Detection Theory. <http://www.cns.nyu.edu/~david/handouts/sdt/sdt.html>.
- [28] Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques. 2 edn. Morgan Kaufmann, San Francisco (2005)
- [29] John, M., Maurer, F., and Tessem, B. 2005. Human and social factors of software engineering: workshop summary. SIGSOFT Softw. Eng. Notes 30, 4 (Jul. 2005), 1-6.
- [30] Brandles, U., Network Analysis: Methodological Foundations. Berlin:Springer, 2005.
- [31] Data Weblink, <http://xtdocs.tarla.org/lpav-soc-churn-stat.zip>
- [32] Arisholm, E., Briand, L. C., Predicting fault prone components in a java legacy system, ISESE 2006, pages 8-17, New York, NY, USA, 2006

# Performance Analysis of a Web Server with Dynamic Thread Pool Architecture

Jijun Lu and Swapna S. Gokhale  
Department of Computer Science & Engineering  
University of Connecticut  
Storrs, CT 06269  
{jijun.lu, ssg}@engr.uconn.edu

## Abstract

*Modern Web servers commonly employ the thread pool architecture so that their I/O-intensive workload can benefit from the performance improvements offered by multi-threading. For acceptable service performance, however, the various configuration options and design choices embodied in the thread pool architecture need to be selected appropriately considering the service and workload characteristics. To enable cost-effective and informed selection of configuration options, this paper presents a quantitative methodology to analyze the performance of a Web server which embodies the thread pool architecture. The key aspect of the methodology is a queuing network model which represents the operational mechanisms of the Web server and the dynamic thread pool management policy. The model is implemented using the AnyLogic simulation environment. We demonstrate the utility of the methodology for performance estimation and sensitivity analysis using several examples.*

## 1 Introduction and motivation

The services offered over the World Wide Web (WWW) are becoming increasingly prevalent in business and critical domains. As a result, these services must be offered with superior performance to retain current users and attract new ones [2]. An important component of any WWW service is a Web server. Web servers are expected to serve millions of transaction requests per day with acceptable performance [25]. Therefore, these stringent performance requirements need to be considered explicitly in selecting the configuration options of the hardware and software infrastructure on which the Web service will be deployed.

Web servers and network middleware are particularly apt to fall under the category of I/O-intensive applications [12], which spend a vast majority of their time waiting for I/O operations to complete. In such I/O-intensive applica-

tions, multi-threading can increase server responsiveness, scalability and throughput, and can also enhance process-to-process communication. Therefore, multi-threading is commonly employed in the implementation of a high-performance Web server. The thread pool architecture is one of the most popular concurrency models used to implement multi-threading [20].

Prior to deployment, the different configuration and design options of the thread pool architecture need to be selected appropriately, considering the service and workload characteristics, in order for the service performance to remain acceptable. One way to guide this selection is to experimentally measure the service performance under various configuration scenarios. While such measurement-based selection may be feasible for a small set of alternative choices, it quickly becomes cumbersome, costly, and infeasible to explore the vast design space afforded by the combinations of different parameters. Model-based analysis, which consists of representing the relevant aspects of the thread pool architecture into an appropriate model, validating the model and then using the validated model to predict server performance under different settings can enable such exploration efficiently. Model-based approach is thus an attractive alternative to the measurement-based approach.

In this paper, we present a model-based approach for quantitative performance analysis of a Web server which embodies the thread pool architecture. Central to our approach is a queuing network model which represents the operational mechanics of a Web server, and the dynamic pool management policy of the thread pool architecture. We implement the queuing model using the general-purpose AnyLogic simulation environment [27]. Using several examples, we illustrate how the model could guide the service provider in estimating performance for a given set of configuration options and to assess the sensitivity of the performance metrics to different configuration parameters.

The balance of this paper is organized as follows: Section 2 provides an overview of the thread pool architecture. Section 3 presents the performance analysis methodology.

Section 4 illustrates the utility of the model. Section 5 compares our work to related research. Section 6 offers concluding remarks and directions for future research.

## 2 Thread pool architecture

The thread pool architecture is one of the most widely used models to implement multi-threading in a Web server [4]. In this architecture, a pool of threads is created when a server application is started up. An incoming request is immediately assigned to a thread, if one is available in the thread pool. If all the threads in the pool are busy, the request is added to a queue. Once a thread frees up, it services a request from the head of the queue.

A critical parameter in the thread pool architecture is the thread pool size because it exerts a significant influence on server performance. While a thread pool of higher size may improve throughput, too many threads may also degrade performance because of the additional context switching and management overhead [7]. The size of the thread pool may be controlled either statically or dynamically. In the static approach, the server creates a limited number of threads at startup. If all these threads are busy, then an incoming request is queued. The static approach thus does not consider the actual presented workload while deciding the thread pool size, although it is well recognized that the benefits derived from the thread pool architecture will depend on the workload and service characteristics [28]. For example, when the requests are I/O-intensive, a larger thread pool will improve throughput. By contrast, as the requests turn more CPU-intensive, they consume the processing capacity of the CPU, leaving little room for performance improvement. Thus, for CPU-intensive services, thread pool architecture may provide lower performance benefits than for I/O-intensive requests. A smaller thread pool may thus suffice for CPU-intensive workload, while an I/O-intensive workload may benefit from a larger pool of threads.

In the case of dynamic thread pool management, the pool size is managed using a heuristic approach [19], where the number of threads are varied with the load. Many modern Web servers such as Apache [14] and Microsoft IIS [15] employ the dynamic approach. In this approach, the size of the pool increases with load allowing more requests to be serviced simultaneously, while the pool size is reduced if the load is low. The objective of the heuristic algorithm employed for pool management is to ensure that the pool contains as many threads as necessary to satisfy the workload demand. An example heuristic approach is presented in [4]. In this approach, when the pool is executing two requests simultaneously and the CPU utilization is below 50%, it suggests that the requests are waiting for events or doing some kind of I/O operation. When this situation is detected, the number of threads can be increased so that

more requests can be processed at the same time. A systematic and quantitative evaluation of such heuristic approaches prior to deployment is necessary for effective management of server performance.

## 3 Analysis methodology

In this section, we first describe the basic queuing network model which captures the mechanics of a Web server. We then discuss the heuristic dynamic thread pool management policy. We then define and motivate the relevant performance metrics. Subsequently, we describe the implementation of the model using AnyLogic [27].

### 3.1 Basic queuing model

The basic queuing model for a Web server is shown in Figure 1, which is similar to other popular Web server models [11]. In Figure 1,  $Q_a$  represents a FIFO queue used to hold requests if all the threads in the pool are busy. A request is held in the queue until a thread in the pool becomes available. If at least one thread is available, the incoming request is accepted immediately for service and is routed to  $Q_c$ , where it waits to receive service at the CPU.

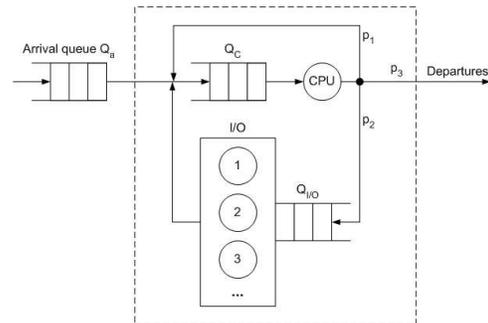


Figure 1. Queuing model of a Web server

In this model, each thread gets a small unit of CPU time (called time quantum) in a round-robin fashion [19]. After this time has elapsed, the thread is preempted and added to the end of the ready queue. If a thread finishes execution before the time quantum expires, it is also preempted and returns to the thread pool to service the request. The processing of a request at the CPU may also be interrupted if the request requires an I/O operation, such as a database access or a disk I/O. Upon each such occurrence, the request is routed and queued at one of the I/O devices. After completing I/O service, the request is sent back to  $Q_c$ . This may be repeated several times depending on routing probabilities  $p_1$  and  $p_2$ .

We note that most of the existing queuing models of a Web server assume that there can be no successive CPU

bursts [11]. Our model is more general compared to these because it allows consecutive CPU runs when  $p_1 > 0$ . Thus, our model can also consider CPU-intensive jobs, where the CPU is mostly bound to heavy computation and the I/O requirements are not as high.

### 3.2 Pool management policy

The heuristic approach [19] we consider to manage the size of the thread pool seeks to emulate a real world Web server. The Web server spawns a certain initial number of threads,  $N_{min}$  at startup. Subsequently, the size of the thread pool is adjusted in response to the workload as follows. If the number of current free threads,  $f$  falls below the minimum allowable number of free threads ( $f < f_{min}$ ), the size of the thread pool is increased by  $s$ , so long as the increased thread pool size remains under the maximum allowable thread pool size,  $N_{max}$ . On the other hand, if the number of free threads exceeds the maximum number of allowable threads ( $f > f_{max}$ ), the thread pool size should be decreased by  $s$ , so long as the reduced thread pool size remains above the minimum allowable size  $N_{min}$ . We note that the  $f_{max}$  and  $f_{min}$  should be set so that  $f_{max} - f_{min} \geq s$ . This is necessary to prevent oscillations between consecutive increases and decreases to the size of the thread pool.

### 3.3 Performance metrics

In this section, we define and motivate the relevant performance metrics for a Web server.

- **Response time:** The response time of a request is defined as the time elapsed between the point the request is accepted by the server to the point it departs from the server. It is one of the most common metrics used to characterize server performance, and also one of the most important ones from the clients' perspective. A long response time may suggest configuration issues or a slow processing server and may discourage the clients. This can result in financial losses, especially for e-commerce providers.
- **Average number of busy threads:** We measure the thread pool utilization with the commonly used metric of average number of busy threads [5]. A service provider will be interested in maximizing the return on investment, and hence, will seek to maintain the average number of busy threads as close to the maximum allowable size of the thread pool as possible. From the users' point of view, however, the average number of busy threads will have a direct impact on the response time. Thus, the average number of busy threads should strike a balance between utilizing the

thread pool to an extent that it remains financially beneficial to the provider, while simultaneously ensuring that the response time remains acceptable to the users.

- **Loss probability:** A request is rejected and typically a "503 Service Unavailable" [5] message is sent back to the client if the queue is full. Loss probability is the likelihood that a request is rejected due to the lack of buffer space. Since the service should operate with minimal loss, this metric could be used to guide the provisioning of the thread pool and buffer capacity.

### 3.4 Model implementation

We implemented the dynamic thread pool management policy described above in AnyLogic [27], which is a Java-based hybrid simulation development environment. We assume that requests arrive at the server according to a Poisson distribution with rate  $\lambda$ . The CPU run length, context switching and I/O access times are exponentially distributed with means  $1/\rho$ ,  $1/k$  and  $1/\tau$  respectively. The capacity of the queue is denoted  $Q$ . The model and pool management parameters are summarized in Table 1.

**Table 1. Model and management parameters**

Parameter	Interpretation
$N$	Current pool size
$N_{min}$	Minimum pool size allowed
$N_{max}$	Maximum pool size allowed
$p_1$	Prob. of successive CPU runs
$p_3$	Prob. of I/O after a CPU run
$Q$	Queue size
$\lambda$	Request arrival rate
$1/\rho$	Average CPU run length
$1/k$	Average context switch time
$1/\tau$	Average I/O time
$f$	Current free threads
$f_{min}$	Minimum free threads
$f_{max}$	maximum free threads
$s$	Step of thread pool change

## 4 Illustrations

In this section, we illustrate the utility of the performance analysis methodology with examples.

### 4.1 Performance estimation

Typically, a service provider is interested in estimating the performance of a Web server for a given set of configuration options. Furthermore, the service provider may also

be interested in determining the level of load at which a chosen set of configuration options ceases to provide acceptable performance. Our analysis approach can offer quantitative performance estimates to the provider to answer these questions. For the sake of illustration, we set the parameters of the model to their nominal values in Table 2. We note that these values may not reflect the parameters found in practical Web server configurations. We choose these values merely for the sake of illustration.

**Table 2. Nominal parameter values**

Parameter	Value	Parameter	Value
$\rho$	500/sec	$N$	10
$\tau$	10/sec	$N_{min}$	10
$k$	10000/sec	$N_{max}$	50
$p_1$	0.01	$f_{min}$	5
$p_2$	0.49	$f_{max}$	10
$p_3$	0.50	$s$	5
$Q$	100		

The values of  $p_1$ ,  $p_2$  and  $p_3$  are chosen as follows. Since typical Web requests are I/O- rather than CPU-intensive,  $p_2$  is expected to be much higher than  $p_1$ , so the likelihood of a request going through two consecutive CPU runs is minimal. We then use the observed response time,  $T$ , to estimate  $p_2$ .  $T$  can be expressed as:

$$T = 1/\rho + (p_1 \cdot T_1 + p_2 \cdot T_2 + p_3 \cdot T_3) \quad (1)$$

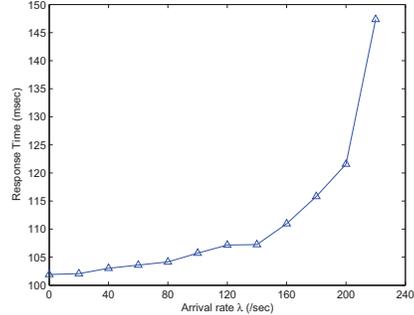
where  $T_i$  denotes the remaining time a request stays in the system when choosing routing path  $i$  with probability  $p_i$  after a CPU run. Specifically, we note that  $T_3 = 0$  since  $p_3$  denotes the probability that a request exits the server. Since  $p_1$  is expected to be very small, we have:

$$\begin{aligned} T &\approx 1/\rho + p_2 \cdot T_2 \\ &= 1/\rho + p_2 \cdot (1/\tau + 1/\rho + p_2 \cdot (1/\tau + 1/\rho + p_2 \cdot (\dots))) \\ &= 1/\rho \cdot \frac{1}{1 - p_2} + 1/\tau \cdot \frac{p_2}{1 - p_2} \end{aligned} \quad (2)$$

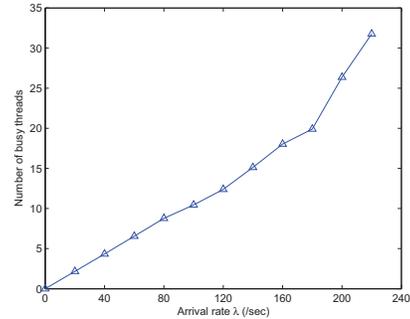
Using the average response time,  $T$  of 100 msec. from the earlier work [13],  $p_2$  computed from Equation (3) is 0.5, leading to an approximate value of 0.5 for  $p_3$ .

To determine the level of load for which the current configurations provide acceptable performance, we vary the arrival rate of the requests and estimate the performance metrics by simulating the AnyLogic model. The results show that until the arrival rate approximately reaches 200/sec., the average response time shows a linear increase with the arrival rate. Beyond the threshold rate of 200/sec., however, the average response time records a sharp increase. The increase in the average number of busy threads as a function

of the arrival rate is consistent with the trend in the response time; beyond a threshold arrival rate of 200/sec., there is a sharp increase in the average number of busy threads.



(a) Average response time (msec)



(b) Average number of busy threads

**Figure 2. Performance metrics vs. arrival rate**

## 4.2 Sensitivity analysis

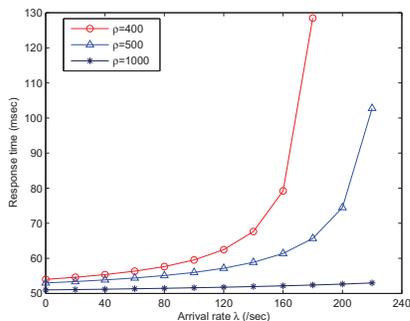
Next, we illustrate how the queuing model could be used to analyze the sensitivity of the service performance to the different configuration and design parameters of the Web server infrastructure. For these experiments, we report the average response time and the average number of busy threads. The loss probability is not reported because it is negligible as would be expected in any typical service offering. In each experiment, we vary the values of the parameter whose impact is being assessed, and hold other parameters at their initial values in Table 2.

### 4.2.1 Impact of CPU run length

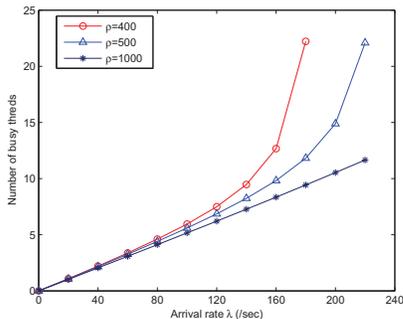
Computational capacity is one of the most important index of a Web server's performance [18]. Although a faster CPU is preferable, a cutting-edge powerful CPU may not always be necessary and cost effective. Furthermore, upgrading the

CPU without considering other parameters, such as I/O access time, may provide little performance benefit if the CPU is not the performance bottleneck. Through this experiment, we demonstrate how sensitivity analysis could suggest an appropriate CPU speed for a given workload.

We consider three values of  $\rho$ , namely, 400/sec., 500/sec. and 1000/sec. Figure 3 shows the performance metrics as a function of the arrival rate for the three experiments. The figure indicates that for a given value of  $\lambda$ , as  $\rho$  increases, the response time decreases. Furthermore, for lower values of  $\lambda$ , the response time offered by all the three CPUs is very close. As the arrival rate increases, for each run length, a threshold value of  $\lambda$  is reached beyond which the response time increases sharply. For these configurations, arrival rates exceeding 160/sec. require a faster CPU, while slower CPUs may suffice for lower arrival rates.



(a) Average response time (msec)



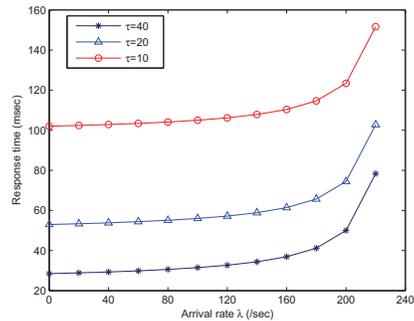
(b) Average number of busy threads

**Figure 3. Impact of  $\rho$  on server performance**

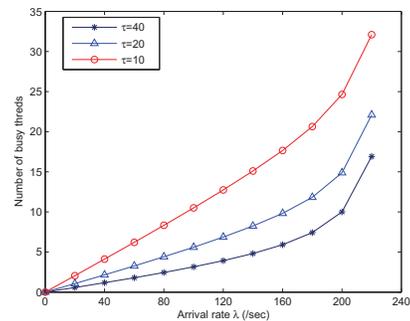
#### 4.2.2 Impact of I/O access time

Sensitivity analysis with respect to I/O access time is especially important because I/O access time typically dominates the response time of requests served by a Web server. As in the case of CPU, a faster I/O device though preferable may not be the most cost effective. Furthermore, if the I/O device is not configured properly, it may degrade

the response time rapidly, thereby significantly limiting the performance improvement potential.



(a) Average response time (msec)



(b) Average number of busy threads

**Figure 4. Impact of  $\tau$  on server performance**

We consider three values of  $\tau$ , namely, 40/sec, 20/sec and 10/sec. The performance metrics as a function of the arrival rate are shown in Figure 4. The figure suggests that for each  $\tau$ , the average number of busy threads increases linearly with the arrival rate and the rate of increase is the highest for the lowest I/O access time. When the arrival rate is approximately 160/sec., the average number of busy threads records a sharp increase for all values of  $\tau$ . Furthermore, for each  $\tau$ , the response time increases very slowly with arrival rate, until the arrival rate reaches 160/sec. At the arrival rate of 160/sec., the response time increases sharply due to queuing. Additionally, for each arrival rate, the response time increases with decreasing  $\tau$ .

It can be seen from Figure 4, however, that although larger  $\tau$  (lower I/O access time) results in a lower response time, the degradation of the response time as a function of  $\tau$  is faster than as a function of  $\rho$ . This is because when  $\tau$  increases to 40, the I/O device is upgraded, but the CPU capacity and the thread pool size remain at their original levels. Thus, the capacity of the thread pool is exhausted at a rate faster than the rate at which the I/O device becomes the bottleneck. Therefore, to reap the benefits of a faster I/O device we also need to upgrade the pool size and/or the

CPU capacity to slow the degradation in the response time.

The above analysis indicates that the process of improving Web server performance must consider two objectives, namely, lowering the actual response time and slowing down the performance degradation. To achieve the second goal, we must consider the contributions of the CPU and I/O to the response time. Since Web requests are I/O-intensive, an upgrade of the CPU should at least be matched with an upgrade of the I/O to achieve the second goal.

## 5 Related research

Web server performance has been an active area of research in the past decade. Most of the research efforts treat the server as a black box. Slothouber [22] proposes to model a Web server as an open queuing network. Van der Mei *et al.* [25] present an end-to-end queuing model for the performance of Web servers, encompassing the impact of client workload characteristics, server hardware/software configurations, communication protocols, and interconnect topologies. Cao *et al.* [3] use an  $M/G/1/K * PS$  queuing model to analyze the performance of a Web server. Nossenson and Attiya [17] introduce a new  $N - Burst/G/1$  queuing model with heavy-tailed service time distribution for Web server performance modeling. Squillante *et al.* [23] employ a  $G/G/1$  queue to model high-volume Web sites. Kant and Sundaram [10] describe a queuing network model for a multiprocessor system running a static Web workload. Dalal *et al.* [6] incorporate user impatience into a  $M/M/1$  queuing model of a Web server. Iyengar *et al.* [8] model a Web server using PH/PH/P model. Kant *et al.* [10] describe a queuing network model of a Web server with architectural details. Tanaka *et al.* [24] model the disk subsystem of a Web proxy server using a BCMP queue. Sha *et al.* [21] propose a combination of feedback control and a  $M/M/1$  queuing model to control the timing performance of a Web server. Abdelzaher *et al.* [1] use a control-theoretic approach to provide performance guarantees and differentiated levels of service in a Web server. Kamra *et al.* [9] present a control-theoretic approach that both prevents overload and enforces absolute response times. Liu *et al.* [13] provide a model of a three-tiered Web services architecture, where each tier is modeled by a multi-station queuing center. Wells *et al.* [26] use a Timed Hierarchical Colored Petri Net to model a Web server and describe how the model could be used to reveal the impact of arrival rates and examine alternative configurations of the Web server.

A few efforts analyze the attributes of a Web server based on its software architecture. Menascé [16] provides a classification of Web server software architectures and present an approach based on queuing networks to study the pool size behavior. Ling *et al.* [12] provide an analysis of optimal thread pool size, which handles Web requests with small

service time. Compared to these two efforts, our model is more general and can handle various aspects of a thread pool architecture, including dynamic thread pool management and context switching. Through a proper choice of routing probabilities, it also allows us to vary the workload characteristics. Furthermore, it does not place any limitations on the length of the service time. The methodology thus provides an integrated framework to analyze the impact of different factors, such as thread pool size, CPU run length, I/O access and context switching times and workload characteristics on server performance. The model can be thus used to facilitate the selection of hardware and software configuration parameters in a cost-effective manner.

## 6 Conclusions and future research

This paper proposes a quantitative methodology to analyze the performance of a Web server which uses the thread pool architecture to implement multi-threading. The core of the methodology is a queuing model which captures the mechanics of the operation of a Web server and the dynamic pool management policy. We implement the queuing model using the AnyLogic simulation framework. We illustrate the potential of the methodology for performance estimation and sensitivity analysis. Our future research directions include extending the methodology to incorporate: (i) different queuing models, and (ii) the impact of caching.

## Acknowledgments

This research is supported in part by a CAREER award from the National Science Foundation (#CNS-643971).

## References

- [1] T. Abdelzaher, K. Shin, and N. Bhatti. Performance guarantees for Web server end systems: a control-theoretical approach. *IEEE Trans. on Parallel and Distributed Systems*, 13(1):80–96, January 2002.
- [2] Y. Bakos. The emerging role of electronic marketplaces on the Internet. *Communications of the ACM*, 41(8):35–42, 1998.
- [3] J. Cao, M. Andersson, C. Nyberg, and M. Kihl. Web server performance modeling using an  $M/G/1/K*PS$  queue. In *Proc. 10th Int'l Conference on Telecommunications*, pages 1501–1506, 2003.
- [4] D. Carmona. Programming the thread pool in the .NET framework. <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndotnet/html/progthreepool.asp>.

- [5] B. Curry, G. V. Reilly, and H. Kaldestad. The art and science of web server tuning with Internet Information Services 5.0. <http://www.microsoft.com/technet/prodtechnol/windows2000serv/technologies/iis/maintain/optimize/iis5tune.mspx>.
- [6] A. C. Dalal and S. Jordan. Improving user-perceived performance at a World Wide Web server. In *Proc. IEEE Global Telecommunications Conference*, pages 2465–2469, 2001.
- [7] B. Goetz. Java theory and practice: Thread pools and work queues. <http://www-128.ibm.com/developerworks/java/library/j-jtp0730.html>.
- [8] A. Iyengar, E. MacNair, M. Squillante, and L. Zhang. A general methodology for characterizing access patterns and analyzing Web server performance. In *Proc. 6th IEEE/ACM Int'l Symp. on Modeling, Analysis and Simulation of Computer Telecommunications Systems*, pages 167–174, 1998.
- [9] A. Kamra, V. Misra, and E. M. Nahum. Yaksha: a self-tuning controller for managing the performance of 3-tiered Web sites. In *Proc. 12th IEEE Int'l Workshop on Quality of Service*, pages 47–56, 2004.
- [10] K. Kant and C. R. M. Sundaram. A server performance model for static Web workloads. In *Proc. IEEE Int'l Symp. on Performance Analysis of Systems and Software*, pages 201–206, 2000.
- [11] S. Kounev and A. Buchmann. Performance modeling of distributed e-business applications using Queuing Petri Nets. In *Proc. Int'l Symp. on Performance Analysis of Systems and Software*, pages 143–155, March 2003.
- [12] Y. Ling, T. Mullen, and X. Lin. Analysis of optimal thread pool size. *ACM SIGOPS Operating System Review*, 34(2):42–55, 2000.
- [13] X. Liu, J. Heo, and L. Sha. Modeling 3-tiered Web applications. In *Proc. 13th IEEE Int'l Symp. on Modeling, Analysis and Simulation of Computer Telecommunications Systems*, pages 307–310, 2005.
- [14] Apache Software Foundation. Apache HTTP server project. <http://httpd.apache.org/>.
- [15] Microsoft Corporation. Internet information services (IIS). <http://www.microsoft.com/WindowsServer2003/iis/default.mspx>.
- [16] D. A. Menascé. Web server software architecture. *IEEE Internet Computing*, 7(6):78–81, 2003.
- [17] R. Nossenson and H. Attiya. The N-burst/G/1 model with heavy-tailed service-times distribution. In *Proc. 12th Annual Int'l Symp. on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems*, pages 131–138, 2004.
- [18] F. Prefect, L. Doan, S. Gold, Th. Wicki, and W. Wilcke. Performance limiting factors in http (web) server operations. In *Proc. Technologies for the Information Superhighway*, pages 267–272, 1996.
- [19] J. Richter. *Programming applications for Microsoft Windows (4th Edition)*. Microsoft Press, 1999.
- [20] D. C. Schmidt and S. Vinoski. Comparing alternative programming techniques for multi-threaded servers - the thread-pool concurrency model. *SIGS*, 8(4), April 1996.
- [21] L. Sha, X. Liu, Y. Lu, and T. Abdelzaher. Queuing model based network server performance control. In *Proc. 23rd IEEE Real-Time Systems Symp.*, pages 81–90, 2002.
- [22] L. Slothouber. A model of Web server performance. In *Proc. 5th Int'l World Wide Web Conference*, 1996.
- [23] M. S. Squillante, D. D. Yao, and L. Zhang. Web traffic modeling and Web server performance analysis. In *Proc. 38th Conference on Decision and Control*, pages 4432–4439, 1999.
- [24] A. Tanaka, M. Takahashi, and K. Tatsukawa. Performance modeling of a disk subsystem and its application to performance design of a Web proxy server. In *Proc. 2003 IEEE Int'l Performance, Computing, and Communications Conference*, pages 407–415, 2003.
- [25] R. D. van der Mei, R. Hariharan, and P. Reeser. Web server performance modeling. *Telecommunication Systems*, 16(3-4):361–378, 2001.
- [26] L. Wells, S. Christensen, L. M. Kristensen, and K. H. Mortensen. Simulation based performance analysis of Web servers. In *Proc. 9th Int'l Workshop on Petri Nets and Performance Models*, pages 59–68, 2001.
- [27] XJ Technologies. AnyLogic. <http://www.xjtek.com/>.
- [28] D. P. Xu and B. Bode. Performance study and dynamic optimization design for thread pool systems. In *Proc. Int'l Conference on Computing, Communications and Control Technologies*, 2004.

# Some Improvements for More Precise Model Checking

Zhi Zhang, Qingkai Zeng, Ming Huang

State Key Laboratory for Novel Software Technology, Nanjing University  
Department of Computer Science and Technology, Nanjing University  
Nanjing 210093, P.R. China  
zhangzhi110119@gmail.com, zqk@nju.edu.cn

**Abstract**—Model checking is now used more and more widely in vulnerability detection for its capacity to model the program and reason about its behavior. However, the state explosion problem impedes its application in large-scale systems, and most of current model checkers are mainly used for moderate-sized systems, like communication protocols and device drivers. To our knowledge, MOPS is the only model checker that can be used for checking an entire linux distribution-Red Hat Linux 9. MOPS is a tool designed to check for violations of rules that can be expressed as temporal safety property. However, it trades precision for scalability. So, in this paper, we introduce a tool called EMOPS (Extended Model checking Programs for Safety properties), which extends MOPS through not only the combination of dataflow and control flow information but also the path verification technique. The goal of EMOPS is to detect vulnerabilities more precisely and keep MOPS' scalability. Experiments show EMOPS's effectiveness compared with MOPS.

**Keywords**-model checking; dataflow; control flow; path verification

## I. INTRODUCTION

Model checking is an automatic technique for verifying finite-state systems [1]. It exhaustively checks a finite-state model of a system for violation of safety property formally specified as a formula in some temporal logic, an automaton, or a collection of assertions [2]. In recent years, the usage of model checking has been growing to check correctness of a wide-variety of systems because of its capacity to reason about the behavior of finite-state systems, and it attracts more and more attention from not only the academic researchers but also many international corporations and government research labs. The model checkers for C programs can be broadly classified into two categories: execution-based and abstraction-based model checkers. Execution-based model checkers, like Zing [4], CMC [5], VeriSoft [6] and SPIN [7], exhaustively search the concrete state space of the system for error states. While the abstraction-based model checkers, like BLAST [8], SLAM [10], MAGIC [11] and CBMC [12], use predicate abstraction to construct tractable model for the system during the model checking. However, one of the most apparent problems with these execution-based and abstraction-based model checkers is the state explosion, since the number of states can grow exponentially with the size of the system. Consequently, many state space reduction techniques have been proposed to address this problem, including partial order reduction, slicing and symmetry reduction [14] and so on. Nevertheless, the state explosion problem remains a serious difficulty in most

applications of model checking. That also explains why most of current model checkers are mainly used in the verification of protocols, device drivers or other moderate-sized systems.

To the best of our knowledge, MOPS [3] is the only model checker that can be used for checking an entire linux distribution-Red Hat Linux 9 [13]. MOPS is control flow and path sensitive but dataflow insensitive in the checking process. The experimental results [13] show that it has successfully detected many security bugs in the linux, but it also suffers from high false positive and false negative because of its dataflow insensitive method.

In a word, existing model checkers either cannot be applied to large-scale systems because of state explosion or trade precision for scalability like MOPS. To overcome these problems, we have developed an extended tool based on MOPS, called EMOPS, to greatly increase MOPS' precision and maintain its scalability. The new features added into MOPS are:

(1) Combination of control flow and dataflow information: EMOPS tracks both control flow and dataflow information to precisely detect the vulnerabilities in the program. To reduce its cost, a method for simplified and demand-driven dataflow analysis is implemented in EMOPS.

(2) Counterexample path verification: EMOPS eliminates the spurious counterexample paths to have more precise results through path verification. First, the counterexample path is instrumented with assert statements. Then, the instrumented path is submitted to the model checker BLAST [8] to verify its feasibility.

The rest of this paper is organized as follows: Section II presents an overview of EMOPS. Section III and V describe the main techniques used in dataflow analysis and counterexample path verification separately. Section IV discusses an extended model checking algorithm used in EMOPS. Section VI gives experimental results to show the effectiveness of EMOPS compared with MOPS. Section VII discusses some related work and section VIII makes a conclusion.

## II. EMOPS OVERVIEW

MOPS [3] is a model checker for C programs and it has achieved high scalability with the price of lower precision in vulnerability detection. Three main reasons for its lower precision are revealed as follows.

First, MOPS describes the safety property by finite state automaton (FSA). So whether FSA is accurate enough to characterize the safety property has a great influence on

MOPS’ precision. For example, when MOPS is used to check the standard file descriptor attack [13], it returns many spurious warnings. One of main reasons is that program may do safe opens in some other way that is out of the description of FSA.

Second, MOPS models the program as a pushdown automaton (PDA) according to its control flow. When we try to check safety property containing dataflow information, especially the alias information, MOPS will fail to detect the relevant vulnerabilities.

Third, MOPS takes account of all possible paths regardless of their feasibility in the program. This will cause false alarms because of spurious paths.

All these factors contribute to MOPS’s lower precision and also point out the way for its improvements. For the first problem, we can address it by studying thoroughly the safety property and trying to include all necessary information during the construction of FSA. For the other two problems, to make MOPS more precise, a new method is proposed to perform efficient dataflow analysis during the model checking and an improved algorithm is also proposed for handling the extended pushdown automaton. In addition, the model checker BLAST [8] is integrated into MOPS to verify the counterexample paths’ feasibility to eliminate the spurious ones.

#### A. The tool overview

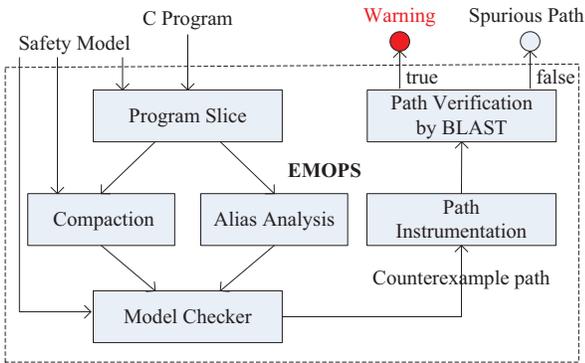


Figure 1. EMOPS overview

Figure 1 shows the overview of EMOPS. The inputs for EMOPS are the C program to be checked and the safety model described as FSA to represent the temporal safety property. The counterexample paths violating the safety model are reported as the output while the spurious paths are ignored.

In EMOPS, Program Slice is first used to extract safety-relevant functions and remove those irrelevant ones in C program according to the safety model. Then Alias Analysis is performed on the sliced C program to reduce the cost of dataflow analysis. Compaction is mainly used to remove the irrelevant statements in the sliced C program according to operations taken in the safety model, which will greatly reduce the state space of the constructed model for the sliced C program. After that, Model Checker constructs pushdown automaton for the compacted C program, and checks it against safety model with the help of dataflow information derived from Alias Analysis. Finally, Model Checker returns the counterexample paths that may violate the safety model. To eliminate spurious paths, BLAST is employed to verify their feasibility. If the counterexample path is feasible, then EMOPS reports this vulnerability to user, otherwise the spurious one is

discarded. Before we submit the counterexample path to BLAST, Path Instrumentation inserts an assert statement before each condition statement along the counterexample path to indicate the true or false choice of the control flow on the path.

### III. DATAFLOW ANALYSIS

In MOPS, the temporal safety property is represented as a finite state automaton FSA and the C program as a pushdown automaton PDA. To check the property, MOPS has to compute the intersection of PDA and FSA and then determines whether the error state can be reached. During the construction of PDA for C program, MOPS uses dataflow insensitive method to simplify the checking process. However, there exist many safety properties needed to be analyzed with both program’s control flow and dataflow. Such cases can be found during the checking for memory-related vulnerabilities, like the null pointer dereferences, memory leak caused by dynamical memory allocation  $y=malloc()$  without calling  $free(y)$ , and double free vulnerability which executes  $free(y)$  twice.

Here  $malloc$  and  $free$  are referred as safety-relevant operation and  $y$  as safety-relevant operation object. The combination of safety-relevant operation and its safety-relevant operation object can trigger a state change in safety model FSA. Safety-relevant statement is a statement containing safety-relevant operation, and safety-relevant function is defined as a function that calls safety-relevant statements directly or indirectly.

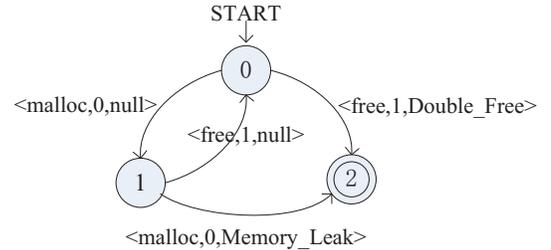


Figure 2. Safety model FSA for  $malloc$  and  $free$

In a safety model FSA, node represents state and edge represents the transition between the states, which is labeled by:  $\{<op, loc, vul\_t> \mid op \in \text{safety-relevant operations}, loc \in \{0,1,2,\dots\}, vul\_t \in \text{vulnerability types}\}$ .  $loc$  denotes the location of the safety-relevant operation object in  $op$ , where 0 represents the return value and  $i(i>0)$  represents the  $i$ th parameter of  $op$ .  $vul\_t$  is set to the vulnerability type if the edge points to the error state, otherwise  $vul\_t$  is set to  $null$ . A simple safety model FSA for  $malloc$  and  $free$  is showed in Figure 2, where 2 is the error state. The path  $0 \rightarrow 1 \rightarrow 2$  denotes state transitions of a memory leak vulnerability in program, with the assumption that there are some dataflow dependency between the safety-relevant operation objects of two  $malloc$ , in other words, they are the same or point to the same memory.

To verify the satisfaction of those temporal safety properties, which describes a sequence of safety-relevant operations related through some dataflow dependency, a combination of dataflow and control flow information should be integrated into the model checking process. The key idea behind the solution is to introduce the alias analysis mechanism to have more precise vulnerability detection.

### A. Simplified and Demand-driven Alias Analysis in EMOPS

To reduce the cost of alias analysis, we have used a demand-driven, flow and context insensitive alias analysis method on the sliced program, which can carry out efficient alias analysis for those parts of the program of our interest.

During the model checking of temporal safety property, we are mainly concerned about the behavior of safety-relevant operations and their safety-relevant operation objects in the program. So, in this paper, we focus on alias analysis in only those safety-relevant functions, ignoring the impact of other safety-irrelevant ones. The FSA in Figure 3 accepts any functions of the form  $e[op_1|op_2|op_3|\dots|op_k]^*x$ , where  $e$  is the entry point of the function,  $x$  is the exit point and  $op_i$  represents safety-relevant operations appeared in the corresponding safety model FSA. Then safety-relevant function can also be defined as a function that is accepted by FSA in Figure 3.

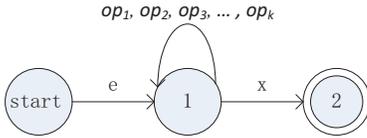


Figure 3. FSA that accepts the language  $e[op_1|op_2|op_3|\dots|op_k]^*x$

Figure 4 shows the algorithm for extracting safety-relevant functions from the program.

1. Use GCC to compile the source program to get its intermediate representation in Abstract Syntax Tree (AST).
2. For each function  $f_i$  in the program, define a vector  $cv_i$  to store  $f_i$ 's callers and a boolean  $rel_i$  to denote whether  $f_i$  is a safety-relevant function.
  - /\* Find out safety-relevant functions calling safety-relevant statements directly. \*/
3. For each function  $f_j$ :
  - For each statement  $s_i$  of  $f_j$  in the form of AST:
    - If  $s_i$  is a safety-relevant statement
      - Set  $rel_i := \text{true}$ ;
    - Else if  $s_i$  is a call to a function  $f_j$  defined in the program
      - Add  $f_i$  into  $cv_j$ ;
  - /\* Find out safety-relevant functions calling safety-relevant statements indirectly. \*/
4. For each function  $f_i$  whose  $rel_i$  is true:
  - Add  $f_i$  into vector  $rel\_func\_set$ ;
5. While  $rel\_func\_set$  is not empty
  - Get an element  $f_i$  from  $rel\_func\_set$
  - For each  $f_j$  in  $cv_i$ , if  $rel_j$  is false
    - Set  $rel_j := \text{true}$ ;
    - Add  $f_j$  into  $rel\_func\_set$ ;

Figure 4. Compute safety-relevant functions

Then the demand-driven alias analysis is done on the safety-relevant functions in bottom-up order. In traditionally exhaustive alias analysis, we compute points-to sets for each program variable. And then intersect their points-to sets to determine whether they point to the same address. However, in demand-driven alias analysis, given an alias query about “whether  $p$  and  $q$  point to the same address”, we search from  $p$  and  $q$  respectively to find out if they can reach the same node in points-to graph representation of the pointer-related

assignment in a program. In the worst case, the demand-driven analysis might end up performing as much work as constructing full points-to sets. However, in most cases it can quickly answer alias queries without deriving full points-to sets.

During the alias analysis in safety-relevant functions (in Figure 5), the demand-driven alias analysis (in Figure 6) is called for leaf nodes and the fix point computation is called for nodes in loop until no new alias relations are generated. Finally, the result about alias relationship between safety-relevant operation objects is used in the model checking process to indicate whether the sequence of safety-relevant operations are related through some dataflow dependency.

1. Construct call graph for the safety-relevant functions. Each node  $nd$  represents one function  $f$ , where  $nd \rightarrow callers$  and  $nd \rightarrow callees$  are used to store  $f$ 's callers and callees respectively, and  $nd \rightarrow aliases\_c$  to store aliases generated at call statements to other functions.
  - /\*  $leaf\_set$  is used to store the leaf nodes and  $node\_set$  to store all the nodes \*/
2. For each node  $nd$ 
  - Add  $nd$  to  $node\_set$ ;
  - If  $nd \rightarrow callees$  is empty
    - Add  $nd$  to  $leaf\_set$ ;
3. While  $leaf\_set$  is not empty
  - Get a node  $nd$  from  $leaf\_set$ ;
  - Remove node  $nd$  from  $node\_set$ ;
  - $alias\_result := \text{Demand-Driven Alias Analysis}(nd)$ .
  - /\* Map the alias analysis result  $alias\_result$  to its callers \*/
  - For each caller  $c$  (corresponding to node  $nd$ ) of  $f(nd)$ 
    - Map the aliases in  $alias\_result$  to aliases between variables in the call function  $c$ ;
    - Add the Mapping result to  $nd_c \rightarrow aliases\_c$ ;
    - Remove  $nd$  from  $nd_c \rightarrow callees$ ;
    - If  $nd_c \rightarrow callees$  is empty
      - Add  $nd_c$  to  $leaf\_set$ ;
  - /\* When a loop is found and no leaf is in  $leaf\_set$  \*/
  - If  $leaf\_set$  is empty and  $node\_set$  is not empty
    - /\* Compute fix point of alias for the functions in the loop \*/
    - Fix Point Computation** (nodes in loop)
      - For each node  $nd_c$  outside the loop pointing to a node  $nd$  in loop
        - Remove  $nd$  from  $nd_c \rightarrow callees$ ;
        - If  $nd_c \rightarrow callees$  is empty
          - Add  $nd_c$  to  $leaf\_set$ ;

Figure 5. Alias analysis algorithm

According to the call graph constructed in step 1 in figure 5, we can carry out the alias analysis from the leaf nodes in a bottom-up way. Step 2 finds out the leaf nodes, whose children nodes ( $nd \rightarrow callees$ ) is null, in the call graph.  $node\_set$  will be used in step 3 to determine whether there is a loop. In step 3, demand-driven alias analysis is called for each leaf node function, and then map the alias analysis results to its callers.  $leaf\_set$  and  $node\_set$  are used to indicate whether a loop is found, if so, then fix point computation is called to deal with the recursive functions in the loop.

Figure 6 presents the algorithm for demand-driven alias analysis. In step 1, the points-to graph  $G_p$  is constructed for the function  $f$  according to the pointer-related assignments and the alias relations derived from the functions called by  $f$ . Then, in

step 2, the demand-driven alias analysis is performed for the operation objects of safety-relevant operations in  $f$ . Besides, the alias relations in one function can be changed by calling other functions through formal parameters and global variables. So we also take this into account during the alias analysis.

#### Demand-Driven Alias Analysis (node $nd$ )

1. Construct points-to graph  $G_p$  for the function  $f$  corresponding to node  $nd$ .

For each pointer-related assignments  $lhs=rhs$

Find or Create new nodes in  $G_p$ , then Add an edge from node ( $rhs$ ) to node ( $lhs$ ).

For each alias relation  $alias_r$ , where  $p_i$  pointing to  $p_j$ , in  $nd \rightarrow aliases_c$

Find or Create new nodes in  $G_p$ , then Add an edge from node  $p_j$  to node  $p_i$ .

2. Demand-Driven alias analysis

Define a vector  $target\_p$  to store formal parameters of pointer type and global variables appearing in  $f$ .

Define a vector  $op\_obj$  to store the operation objects of safety-relevant operations.

For each pair of objects ( $target\_p_i, target\_p_j$ ) in  $target\_p$

If  $target\_p_i$  and  $target\_p_j$  can reach the same node in the points-to graph  $G_p$

Merge  $target\_p_i$  and  $target\_p_j$  into one set;

Put the merged set into  $target\_p$ ;

( $op\_obj$  is dealt with similar way as  $target\_p$ )

If there is no new alias relationship between objects in  $target\_p$  any more

Return  $target\_p$ ;

Figure 6. Demand-driven alias analysis algorithm

## IV. MODEL CHECKER

The model checker implemented by MOPS can only deal with pushdown automaton containing the control flow of the modeled program while ignoring the dataflow information. So, we have to extend it to handle both control flow and dataflow information. In our model checker, we first extend the basic data structure of the pushdown automaton (PDA) to store the dataflow information by adding an object  $obj$  of our interest to track. A variable is defined as an object of our interest if there is some dataflow dependency between the variable and safety-relevant operation object based on the analysis of dataflow information.

Following is the extended rules for constructing the PDA from the program and the safety model FSA.

For an edge in the program's CFG that is from a program point  $p_1$  to  $p_2$  with a statement  $i$ :

(1) If  $i$  is not a function call, for each transition  $s_1 \xrightarrow{i} s_2$  in the safety model FSA, add a transition  $\langle s_1, p_1, obj \rangle \rightarrow \langle s_2, p_2, obj \rangle$  to PDA, where  $obj$  is an operation object in statement  $i$  that we are interested to track,  $s_2$  is the state of program after the execution of statement  $i$  from state  $s_1$ . The object  $obj$  can be null if the statement  $i$  is safety-irrelevant or has no operation object.

(2) If  $i$  is a call to a function  $f$  and there exists some actual parameter in  $f$  that is an object of our interest, for each state  $s_1$  in FSA, add a transition  $\langle s_1, p_1, actual\_obj \rangle \rightarrow \langle s_1, p_2p_3, formal\_obj \rangle$  to PDA, where  $p_2$  is the entry point of the

function  $f$ ,  $p_3$  is the return point when the call for  $f$  returns,  $actual\_obj$  and  $formal\_obj$  correspond to the actual parameter and formal parameter separately of function  $f$ . Else, add a transition  $\langle s_i, p_1, null \rangle \rightarrow \langle s_i, p_2p_3, null \rangle$  to PDA. It means that when the actual parameter in  $f$  is an object of our interest, we will track it during the function call by passing the actual parameter  $actual\_obj$  from the calling point to its corresponding formal parameter  $formal\_obj$  and passing the state  $s_1$  into the function  $f$ . Otherwise,  $null$  is used and the initial state  $s_i$  of FSA is passed into the function  $f$ .

For an edge in the program's CFG that is a return statement from a function  $f$ , if there exists some object  $obj$  of our interest to track, for each state  $s_1$  in FSA, add a transition  $\langle s_1, f_e, obj \rangle \rightarrow \langle s_1, \epsilon, \epsilon \rangle$  to PDA. Else, add  $\langle s_1, f_e, null \rangle \rightarrow \langle s_1, \epsilon, \epsilon \rangle$  to PDA, where  $f_e$  is the exit point of the function  $f$ , the first  $\epsilon$  denotes that no symbol is pushed onto the stack and the second  $\epsilon$  maps the  $obj$  in  $f$  back to the corresponding object in the calling point. Here we mainly focus on the objects of our interest that are global or passed by parameters, because their states can be changed by  $f$  when  $f$  is called. So we have to map these changes back to the calling point when  $f$  returns.

Then, like MOPS, we use a P-automaton (PA) which is a FSA to describe all reachable states of the constructed PDA. First, take all the states from the PDA and make them as the initial states of the PA, then add a final state to the PA. Second, make all the pairs of the form  $(p, obj)$  in the PDA as the input symbols of the PA. Third, add transitions to the PA to represent its initial configuration. For example, if the PDA's initial configuration is  $\langle s_0, p_0, null \rangle$ , where  $s_0$  is the initial state of the safety model FSA,  $p_0$  is the entry point of the program and  $null$  denotes that it starts with no object, and  $s_e$  is the final state of the PA, add a transition  $s_0 \xrightarrow{(p_0, null)} s_e$  to the PA as its initial configuration. Each time we encounter a new object of interest during the construction process, add a transition  $s_0 \xrightarrow{(p_i, new\_obj)} s_e$  to the PA, where  $p_i$  is the position of  $new\_obj$ 's first appearance and  $s_0$  represents its initial state, and then continue to construct the PA using the extended construction rules.

Following is the extended rules for constructing the PA from a PDA.

Add to PA a new state  $s_t$  for each transition  $t = \langle s_1, p_1, actual\_obj \rangle \rightarrow \langle s_2, p_2p_3, formal\_obj \rangle$  in the PDA, where  $actual\_obj$  and  $formal\_obj$  can be  $null$ , and a new transition  $s_2 \xrightarrow{(p_2, formal\_obj)} s_t$ .

Add new transitions to PA according to the following saturation rules:

(1) If  $t = \langle s_1, f_e, obj \rangle \rightarrow \langle s_2, \epsilon, \epsilon \rangle$  is in the PDA and  $s_1 \xrightarrow{(f_e, obj)} s_q$  is in PA, where  $obj$  can be null, add a new transition  $s_2 \xrightarrow{(\epsilon, \epsilon)} s_q$ .

(2) If  $t = \langle s_1, p_1, obj \rangle \rightarrow \langle s_2, p_2, obj \rangle$  is in the PDA and  $s_1 \xrightarrow{(p_1, obj)} s_q$  is in PA, add a new transition  $s_2 \xrightarrow{(p_2, obj)} s_q$ , where  $obj$  can be  $null$ . For all other transition  $s' \xrightarrow{(p_1, obj')} s''$

in PA, add a new transition  $s' \xrightarrow{(p_2, obj')} s''$ , which means that for all  $obj'$  other than  $obj$  whose state is  $s'$ , after an operation on  $obj$ , the state of  $obj'$  is unchanged and passed to the next position  $p_2$ .

(3) If  $t = \langle s_1, p_1, actual\_obj \rangle \rightarrow \langle s_2, p_2 p_3, formal\_obj \rangle$  is in the PDA and  $s_1 \xrightarrow{(p_1, actual\_obj)} s_q$  is in PA, add a new transition  $s_t \xrightarrow{(p_3, actual\_obj)} s_q$ , where  $actual\_obj$  and  $formal\_obj$  can be *null*. For all other transition  $s' \xrightarrow{(p_1, obj')} s''$  in PA, add a new transition  $s' \xrightarrow{(p_3, obj')} s''$ , which means that for all  $obj'$  other than  $obj$  whose state is  $s'$ , after a call for a function whose entry point is  $p_2$ , only  $obj$  and its state are passed into the called function,  $obj'$  and its state is unchanged and passed directly to the return position  $p_3$ .

In the end, the model checker tests whether the error state in FSA is reachable in PA. If not, the program satisfies the safety property. Otherwise, pass the potential counterexample path to the next step for further verification to remove the spurious path. Details about the counterexample path verification can be found in the next section.

## V. COUNTEREXAMPLE PATH VERIFICATION

The counterexample path returned by MOPS has not been verified, so the path may be feasible in CFG while unfeasible in the real program, which is a source of false positive for MOPS. The advantage of MOPS is that it can ensure the soundness of certain property checking, since it uses the control flow and path sensitive method to traverse all possible paths in the program. So the counterexample paths returned by MOPS include not only paths that indeed violate the property described as FSA but also those infeasible paths. The program in Figure 7 is a fragment of a PCI device driver from [15].

```
do {
    KeAcquireSpinLock();
    nPacketsOld = nPackets;
    if(*) {
        KeReleaseSpinLock();
        nPackets++;
    }
} while(nPackets != nPacketsOld);
KeReleaseSpinLock();
```

Figure 7. A fragment of a device driver code

To check the proper use of spin locks, we have to track the call sequence of lock function *KeAcquireSpinLock()* and unlock function *KeReleaseSpinLock()*. It requires that unlock function *KeReleaseSpinLock()* should be called after the lock function *KeAcquireSpinLock()* and should not be called immediately after itself, which will cause double unlock vulnerability. Obviously, the program in Figure 7 obeys this rule. However, MOPS will give a spurious warning about this double unlock vulnerability because of its control flow sensitive and dataflow insensitive way.

In this case, the problem of false positive cannot be addressed through alias analysis discussed in section III. To get the right result, we have to verify the potential vulnerability path's feasibility and then remove the spurious one.

In this paper, we employ the model checker BLAST [8] to verify the path's feasibility. There are several reasons for this choice. First, BLAST is an open source freeware compared with the commercial software like SLAM developed by Microsoft and it is also an automatic verification tool for checking temporal safety properties of C program. Second,

BLAST uses lazy abstraction [9] to construct, verify and refine the abstract model of the program, which offers significant advantages in performance by maintaining the minimal necessary information to check the property. Third, in BLAST, the model checking of safety property is finally converted to reachability problem of statement with special label or the assert statement. In order to verify the feasibility of a counterexample path by BLAST, we first insert an assert statement before each condition statement along the counterexample path to indicate the choice of the control flow on the path. Then submit the instrumented path to BLAST to check whether all the assert statements along the path are satisfiable. If so, then the counterexample path is feasible. Otherwise, it is spurious and we ignore it. However, when the counterexample path is long and complex, it may be out of the capacity for BLAST to check its feasibility. To overcome this problem and make BLAST scalable to large-scale system, there is another option for us to verify the counterexample path. That's we can adopt the divide and conquer strategy to break the counterexample path into two types of units: the path unit through recursive function and the one through non-recursive function. In this way, the verification of long inter-procedural path can be reduced to feasibility checking of many shorter intra-procedural paths. If anyone of the intra-procedural paths is spurious, we can stop the verification process instantly and eliminate this counterexample path from the warning reports. With the help of BLAST for counterexample path verification, we can get more precise results.

## VI. EXPERIMENTAL RESULTS

We have evaluated EMOPS with several open source applications (each of them has one known bug)[18][19][20], which are carried out on a PC with Dual-Core Intel Pentium E2160 1.80GHz and 1GB memory. All these buggy applications have some memory-related vulnerabilities. To catch these vulnerabilities, we have to track both control flow and dataflow information, and then use path verification to check the feasibility of counterexample paths. According to vulnerability types, they can be classified into three categories: double free (cvs-1.11.4, krb5-1.4.1); memory leak (squid-2.4, wget-1.10.2, which-2.16); and buffer overflow (gzip-1.2.4, ncompress-4.2.4, sendmail-8.7.5 and wu-ftp-2.4.2).

TABLE I. EXPERIMENTAL RESULTS OF EMOPS AND MOPS

Vulnerabilities	Applications	MOPS	EMOPS	Real/Total CE-paths	Path Filter
Double free	cvs-1.11.4	No	Yes	1/2	1
	krb5-1.4.1	Yes	Yes	1/1	0
Memory leak	squid-2.4	No	Yes	1/4	2
	wget-1.10.2	No	Yes	1/9	6
	which-2.16	No	Yes	1/5	2
Buffer overflow	gzip-1.2.4	No	Yes	1/1	0
	ncompress-4.2.4	No	Yes	1/1	0
	sendmail-8.7.5	No	Yes	1/2	1
	wu-ftp-2.4.2	No	Yes	1/3	2

**EMOPS evaluation.** Table I shows the overall results of EMOPS and MOPS with nine buggy applications. "No" and "Yes" in MOPS and EMOPS columns denote whether the vulnerability is caught. "Real/Total CE-paths" column presents the number of feasible counterexample paths in the total reported paths, and the number of spurious counterexample

paths filtered by path verification is shown in “Path Filter” column. The result shows that EMOPS successfully catches all known vulnerabilities in these applications while MOPS fails to do so in most of them. The main reason is that during the model checking, EMOPS tracks and analyzes both intra-procedural and inter-procedural dataflow information. So EMOPS can identify whether any two safety-relevant operations are related through some dataflow dependency. What’s more, the safety model FSA is constructed specially for each known vulnerability in the applications, and path verification in EMOPS also helps reduce false positive. However, MOPS takes only control flow into account, which results in its failure in vulnerability detection in most of these buggy applications.

In all these buggy applications, MOPS only succeeds in catching the double free vulnerability in `krb5-1.4.1`, because this vulnerability is simple and can be detected by intra-procedural analysis. Figure 8 represents the code extracted from `krb5-1.4.1/src/lib/krb5/krb/recvauth.c`.

```
if ((retval = krb5_read_message(context, fd, &inbuf))
    return(retval);
if (strcmp(inbuf.data, sendauth_version)){
    krb5_xfree(inbuf.data);
    problem = KRB5_SENDAUTH_BADAUTHVERS;
}
krb5_xfree(inbuf.data);
```

Figure 8. A fragment of `recvauth.c` in `krb5-1.4.1`

In Figure 8, function `krb5_read_message()` is followed by two `krb5_xfree()` functions, which are used to free the buffer allocated by `krb5_read_message()`. When the `sendauth` version string do not match the expected value `inbuf.data`, the program will call `krb5_xfree()` twice, which results in double free vulnerability.

TABLE II. THE RESULT OF PROGRAM SLICE

Applications	Before Program Slice	After Program Slice	Compaction Rate
<code>cvs-1.11.4</code>	733	315	42.97%
<code>krb5-1.4.1</code>	2439	225	9.23%
<code>squid-2.4</code>	1838	132	7.18%
<code>wget-1.10.2</code>	593	102	17.20%
<code>which-2.16</code>	18	5	27.78%
<code>gzip-1.2.4</code>	96	10	10.42%
<code>ncompress-4.2.4</code>	15	2	13.33%
<code>sendmail-8.7.5</code>	415	197	47.47%
<code>wu-ftp-2.4.2</code>	221	83	37.56%

TABLE III. COMPARISON BETWEEN ALIAS ANALYSIS BASED ON POINTS-TO SETS AND DEMAND-DRIVEN METHOD AND THEIR COST (MS)

Applications	Points-to Set	PTS Cost Rate	Demand -driven	DD Cost Rate	Improvement Rate
<code>cvs-1.11.4</code>	1632.65	31.47%	1225.39	23.62%	7.85%
<code>krb5-1.4.1</code>	1558.30	23.92%	1120.24	17.19%	6.73%
<code>squid-2.4</code>	360.89	22.10%	67.28	4.12%	17.78%
<code>wget-1.10.2</code>	270.43	25.01%	175.93	16.27%	8.74%
<code>which-2.16</code>	171.39	48.91%	124.56	35.54%	13.37%
<code>gzip-1.2.4</code>	122.41	16.06%	94.33	12.39%	3.67%
<code>ncompress-4.2.4</code>	63.05	27.41%	49.25	21.48%	5.93%
<code>sendmail-8.7.5</code>	1327.47	89.09%	1111.36	74.62%	14.47%
<code>wu-ftp-2.4.2</code>	177.33	21.81%	119.83	14.51%	7.30%

Adding dataflow analysis would affect the efficiency of the whole model checking process. To reduce its cost, it is performed on the sliced program. Meanwhile, demand-driven method is also adopted to accelerate the alias analysis. Table II lists the number of functions in the checked applications before and after the program slice according to the safety model. And “Compaction Rate” column shows that program slice greatly reduces the number of functions needed to be analyzed. On average, the size of the program can be reduced to 23.68%. However, the compaction rate varies widely with different applications. For `squid-2.4`, program slice can perform best to compact the application to 7.18% of its original size. While the compaction result of `sendmail-8.7.5` does not seem so striking. So the results of program slice depends on both the structure of the program and the safety model to be checked.

In Table III, it compares the time cost of alias analysis based on points-to sets (“Points-to Set” column) and demand-driven method (“Demand-Driven” column). And “PTS Cost Rate” and “DD Cost Rate” columns show the cost rate of alias analysis based on points-to sets and demand-driven method during the whole model checking process. Compared with the traditional one based on the intersection of points-to sets, in most cases, demand-driven alias analysis is more faster to get result without deriving full points-to sets. From the experimental results in Table II and III, we can conclude that the cost of dataflow analysis would be greatly reduced by combination of program slice and demand-driven method.

**Limitations of EMOPS.** First, EMOPS uses a flow insensitive alias analysis to deal with pointer updates. Sometimes, it may cause false alarms for the conservative approximations in the analysis. However, this limitation can be overcome by using flow sensitive and context sensitive alias analysis with the price of sacrificing its scalability and efficiency. Second, exceptional control flow, which is a change in control flow due to run-time conditions, and non-local jumps by calling `longjump` may cause unexpected control flow, and they are not in Control Flow Graph of the program. So EMOPS cannot analyze them. Third, the decision procedures underlying BLAST implement linear arithmetic and uninterpreted functions. Operations such as multiplication and bit-level manipulations are conservatively treated as uninterpreted functions, which may cause false alarms.

## VII. RELATED WORK

The most closely related work to ours is the MOPS [3]. MOPS is a model checking tool for temporal safety property of C programs. It represents the temporal safety property as a finite state automaton FSA and the C program as a pushdown automaton PDA. To check the property, MOPS has to compute the intersection of PDA and FSA and then determines whether the error state can be reached. It can be used to check large-scale system like an entire linux [13]. However, it trades precision for scalability by considering only control flow information. Finally, it will return the counterexample path without verification of its feasibility, which makes the result inaccurate. Our work is just to overcome these disadvantages of MOPS.

SLAM [10] is an abstraction-based model checker used to verify device drivers. It uses predicates to convert a C program into a Boolean program that contains only Boolean types. Then

it checks the Boolean program to see if an error state is reachable. If the counterexample path to the error state is spurious, SLAM adopts the counterexample-guided abstraction refinement (CEGAR) technique to refine the Boolean program and continues to check until a real error has been found or the program satisfies the property. In this checking process, the CEGAR technique guarantees the feasibility of counterexample path. However, it will also take more effort because of its iterative refinement process when doing dataflow analysis to discover new predicates to refine the abstraction model. Other similar tools include MAGIC [11]. However, the dataflow analysis in EMOPS is more simple and efficient, which is done only once in only those safety-relevant functions through demand-driven method.

CMC [16] is a model checker for checking C and C++ implementations directly, which has been used to find errors in large network protocol implementations like the Linux TCP/IP implementations. Unlike the abstraction-based model checkers, CMC checks the program itself directly instead of the program's abstraction model, which reduces the effort to use model checking. From this point, EMOPS is a bit like CMC. However, when CMC finds a bug in the program, it will print the trace to the error state without feasibility checking. So it cannot guarantee the reliability of its results.

Feaver [17] is developed by Bell labs, which is only able to check a small subset of ANSI C. Before checking the program, it has to translate the C code into an intermediate language Promela, and then submits the Promela code to SPIN for model checking. During the translation to Promela, user has to provide a table for translating different C constructs, which requires a significant amount of manual effort and also can be more error prone. In our work, EMOPS can check program automatically without manual intervention except for the construction of safety model FSA which is easier to construct.

### VIII. CONCLUSION AND FUTURE WORK

In this paper, we describe a tool which improves MOPS's performance from two aspects: combination of control flow and dataflow information, and path verification. Experimental results demonstrate its effectiveness in vulnerability detection compared with MOPS. However, in EMOPS, as most of program analysis tools, the safety model for the temporal safety property has to be constructed manually. In our future work, we will try to make this process automatic through mining techniques to automatically get specification about the temporal safety property from source code.

### ACKNOWLEDGMENT

This work is supported by grants from National Natural Science Foundation of China (60773170, 60721002, and 90818022), National High Technology Research and Development Program of China (2006AA01Z432), Specialized Research Fund for the Doctoral Program of Higher Education of China (200802840002).

### REFERENCES

[1] E. Clarke, O. Grumberg, and D. Peled. Model checking. MIT press, 2000.

[2] M. Robby, M. B. Dwyer, and J. Hatcliff. Bogor: a flexible framework for creating software model checkers. In the Proceedings of Testing:

Academic & Industrial Conference - Practice And Research Techniques (TAICPART'06), pages 3-22, 2006.

[3] H. Chen and D. Wagner. MOPS: an infrastructure for examining security properties of software. In Proceedings of the 9th ACM Conference on Computer and Communications Security, pages 235-244, Washington, DC, USA, 2002.

[4] T. Andrews, S. Qadeer, S. K. Rajamani, J. Rehof, and Y. Xie. Zing: a model checker for concurrent software. In 16th International Conference on Computer Aided Verification, pages 484-487, Boston, MA, USA, 2004.

[5] M. Musuvathi, D. Y. Park, and A. Chou. CMC: a pragmatic approach to model checking real code. In proceeding of the 5th Symposium on Operating Systems Design and Implementation, pages 75-88, Boston, MA, USA, 2002. USENIX Association.

[6] P. Godefroid. Model checking for programming languages using VeriSoft. In Proceedings of the 24th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, pages 174-186, New York, NY, USA, 1997. ACM.

[7] G. J. Holzmann. The model checker Spin. In IEEE Transactions on Software Engineering, Vol. 23, Issue 5, pages 279-295, 1997.

[8] D. Beyer, T. Henzinger, R. Jhala, and R. Majumdar. The software model checker BLAST. In International Journal on Software Tools for Technology Transfer (STTT), Vol. 9, No. 5-6, pages 505-525, 2007.

[9] T. A. Henzinger, R. Jhala, and R. Majumdar. Lazy abstraction. In The 29th SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pages 58-70, Portland, OR, USA, 2002. ACM.

[10] T. Ball, and S. K. Rajamani. The SLAM toolkit. In 13th International Conference on Computer Aided Verification, pages 260-264, Paris, France, 2001.

[11] S. Chaki, E. Clarke, and A. Groce. Modular verification of software components in C. In IEEE Transactions on Software Engineering, Vol. 30, No. 6, pages 388-402, 2004.

[12] E. Clarke, D. Kroening, and F. Lerda. A tool for checking ANSI-C programs. In Tools and algorithms for the construction and analysis of systems (TACAS'04), pages 168-176, 2004.

[13] B. Schwarz, H. Chen, and D. Wagner. Model checking an entire linux distribution for security violations. In Proceedings of the 21st Annual Computer Security Applications Conference, pages 13-22, 2005.

[14] D. Tang, S. Malik, A. Gupta, and C. Norris. Symmetry reduction in SAT-based model checking. In 17th International Conference on Computer Aided Verification, pages 125-138, Edinburgh, Scotland, UK, 2005.

[15] T. Ball, E. Bounimova, B. Cook, V. Levin, J. Lichtenberg, C. McGarvey, B. Ondrusek, S. K. Rajamani and A. Ustunur. Thorough static analysis of device drivers. In Proceedings of the 2006 EuroSys conference, pages 73-85, 2006.

[16] M. Musuvathi, D. Park, A. Chou, D. R. Engler, and D. L. Dill. CMC: a pragmatic approach to model checking real code. In Proceedings of the Fifth Symposium on Operating Systems Design and Implementation, pages 75-88, 2002.

[17] G. J. Holzmann, and M. H. Smith. Software model checking: extracting verification models from source code. In Formal Methods for Protocol Engineering and Distributed Systems, pages 481-497, 1999.

[18] Z. Xu, and J. Zhang. Path and context sensitive inter-procedural memory leak detection. In Proceedings of the 2008 The Eighth International Conference on Quality Software, pages 412-420, 2008.

[19] S. Lu, Z. Li, F. Qin, L. Tan, P. Zhou, and Y. Zhou. BugBench: a benchmark for evaluating bug detection tools. In Proc. of Workshop on the Evaluation of Software Defect Detection Tools, pages 1-5, 2005.

[20] M. Zitser, R. Lippmann, T. Leek. Testing static analysis tools using exploitable buffer overflows from open source code. In Proceedings of the 12th ACM SIGSOFT twelfth international symposium on Foundations of software engineering, pages 97-106, 2004.

# Software Development Effort and Quality Prediction Using Bayesian Nets and Small Local Qualitative Data

Lukasz Radliński

Department of Information Systems Engineering, Institute of Information Technology in Management  
University of Szczecin  
ul. Mickiewicza 64, 71-101 Szczecin, Poland  
e-mail: lukrad@uoo.univ.szczecin.pl

**Abstract**— Large and homogeneous datasets are typically required to predict software development effort and quality accurately. Also, many statistical methods can only be applied when meeting various constraints. This study focuses on developing Bayesian nets (BNs) automatically from a small local dataset. Predictive accuracy of generated BNs keeps the level of other published results but the procedure of building the models is simpler. The accuracy can be improved by incorporating domain expert knowledge.

**Keywords**— effort prediction; quality prediction, Bayesian nets, local data, qualitative factors

## I. INTRODUCTION

Software engineering and statistics literature suggest using large and homogeneous datasets to predict effort and quality. This requirement can hardly be met in industry as extensive metrics programs are expensive.

Additionally, many statistical methods require meeting specific assumptions in respect with the data, like normal distribution, linear relationships etc. Software engineers are hardly aware of these constraints and often either incorrectly use methods which they should not use in their environment or skip the modeling process at all.

Thus, there is a need for such modeling method which is easy to use for non-specialists in statistics, artificial intelligence (AI) or related fields. The procedure of building a model should be flexible about incorporating expert knowledge, generating the model purely from empirical data or mixing expert knowledge with empirical data. Such method should also allow to present both the model and the results clearly.

One of very few methods which meet such criteria is Bayesian nets (BNs). This probabilistic method have been successfully used in various studies, including in software engineering field.

The aim of this study is to analyze if BNs can be effectively used to predict software development effort and software quality. There are two assumptions in this experiment – realistic and important, especially from industrial perspective:

- Available dataset is small but contains local data about past projects from a single company.
- A model is automatically generated from available data without expert input.

It needs to be stressed that it is not the aim of this study to develop predictive models for a general use – for example in different environments and for different types of projects. Rather, it analyzes if other software companies can make use of such small and relatively cheap datasets to predict effort and quality.

## II. BAYESIAN NETS

Bayesian net [8], [16], [23] is a probabilistic model which contains two perspectives: graphical and numeric. Graphically BN is a directed acyclic graph consisting of a set of nodes (variables) and directed links between pairs of nodes. Through this graphical representation BNs allow to clearly present relationships between variables. Numerically each node is defined in terms of conditional probability distributions given its parents. BNs allow both forward and backward reasoning using a Bayes' Theorem [3], one of the bases of probability theory. Although Bayes' Theorem was formulated back in the XVIII century, BN concepts (and the term itself) were introduced in the 1980's in pioneering work by Pearl [22], [23].

Table I summarizes some of recent BNs for software engineering. Most of these models have been built to predict either effort or quality (various measures), some enable an integrated effort and quality prediction. Very few of them focus on other aspects. Four main types of structures have been used in these studies:

- Naive Bayesian classifier (NBC) – a structure of diverging star with all the links directed from a single dependent variable to each predictor.
- Converging star (CS) – all the links pointing from each predictor to a single dependent variable.
- Causal BN (CBN) – contains causal relationships between variables.
- Dynamic BN (DBN) – a sequence of CBNs linked together, where each instance represents system state at specific point of time.

When a model is built purely from the data a CBN structure is developed by a learning algorithm. However, it does not ensure creating causal links but rather links which represent strong statistical dependencies. All of those structures can be used to learn parameters from the dataset. CBNs, DBNs and, partially, NBCs can be used to incorporate expert knowledge in parameter definition.

TABLE I. SOME OF RECENT BNS FOR SOFTWARE ENGINEERING

Author	Main problem analyzed	BN topology
[26]	types of defects	NBC
[27]	effort, productivity	NBC
[21]	fault content, fault proneness	CS
[30]	change coupling	CS
[6]	effectiveness of inspections	CBN
[7]	defect rate	CBN
[9]	need for requirements review	CBN
[11]	trade-off between: size, effort, quality	CBN
[12], [13]	defects, partly: effort	CBN/DBN
[17], [19]	web development effort	CBN
[20]	maturity of requirements	CBN
[24]	trade-off between: size, effort, quality	CBN
[28]	various aspects of software quality	CBN
[29]	testing process	CBN
[2]	failures	DBN
[4]	effort	DBN
[10], [25]	defects	DBN
[15]	project velocity (functionality)	DBN

### III. MATERIAL AND METHOD

In this study the extended version of publicly available dataset [5], [12] of 31 software projects from a single company have been used. This extension includes one additional predictor – *project type*. The dataset contains 29 predictors: *project size*, *project type* and 27 factors describing process and people quality expressed on a 5-point ranked scale (from ‘very low’ to ‘very high’). Two cases have been excluded from the original dataset because they did not contain data on *project size*, which is one of the main factors in effort and defect prediction. Thus, 29 cases have been used in further analysis. Predictive accuracy of four dependent variables have been investigated:

- *effort*,
- *productivity rate (effort / project size)*,
- *number of defects*,
- *defect rate (number of defects / project size)*.

The research procedure followed in this study consists of the following steps:

1. Data preparation – adjusting categories for *project type*, discretizing numeric variables by domain expert into 4 or 5 categories.
2. Random data split with proportions: 22 cases (75%) for model generation, 7 cases (25%) for validation.
3. Learning model structure using Greedy Thick Thinning algorithm implemented in Genie [14].
4. Obtaining predictions from models using testing data subset in Agenarisk tool [1].
5. Repeating steps 1-4 (total of 10 times).
6. Calculating measures of predictive accuracy: MMRE (mean magnitude of relative error), MmMRE

(median magnitude of relative error and  $\text{Pred}(l)$  (prediction at level  $l$ ), as previously in [12].

7. Discovering factors most closely related with the dependent variables which most frequently appear in BN’s Markov blankets for each dependent.

A Markov blanket for a node  $A$  is a set of nodes which, if they are instantiated (i.e. observations have been assigned to them), are the only nodes influencing node  $A$ . Thus, it shields a node from the impact of nodes not belonging to its Markov blanket. It can be identified from the BN structure – for a node  $A$  it contains: parent nodes for  $A$ , child nodes for  $A$ , and other parent nodes for child nodes for  $A$ .

### IV. RESULTS

#### A. Predictive Accuracy

Table II illustrates the summary of predictions by demonstrating different evaluation measures. Results of this study have been compared with two earlier studies where BNs have been used for:

- Defect prediction [12] – where both BN structure and parameters have been built prior to obtaining the dataset (the same dataset as in this study), and the dataset have been used only in validation;
- Effort prediction for web development [17] – where various BNs have been built, and then validated against two datasets (65 projects in each of them).

High predictive accuracy is very rare in the literature, also where other methods have been used. This comparison reveals that predictions in this study are respectable – worse than in study [12] for defect prediction but significantly better than study [17] for effort prediction. They keep the average accuracy in the literature. Yet, the process of achieving them, through BN generation, is easy to follow for non-statisticians and free from various constraints.

Additionally, it should be noted that reduced precision of numeric (discretized) could have a negative impact on limited accuracy. The motivation for discretizing *project size*, apart from such requirement by a structure learning algorithm, was that in real projects it can never be precisely estimated at the early project stage.

Preliminary results from experiments involving other modeling techniques (like stepwise linear regression or neural networks) show that some of them may outperform BNs in terms of predictive accuracy. This may be due to the fact that these other techniques provide predictions for continuous variables without the need to discretize them. However, further experiments are performed to draw conclusions from such comparisons and to publish detailed results.

Higher prediction accuracy obtained from stepwise regression models over BNs in effort prediction have been found in earlier studies by other authors [17], [19]. They have argued that there might have been two reasons for this. The first is not reducing the BN structures to take into account the correlations between factors and effort. The second reason was small dataset used, yet still much larger than in the current study, which did not enable to incorporate various combinations of factors influencing effort [18].

TABLE II. SUMMARY OF PREDICTIONS

Study and dependent variable	Evaluation measure		
	MMRE	MdMRE	Pred(25)
<b>this study:</b>			
effort	0.97	0.60	0.17
productivity	0.77	0.40	0.27
defects	2.14	0.79	0.14
defect rate	0.83	0.53	0.21
<b>in [12]:</b>			
defects	0.96	0.27	0.58 <sup>a</sup>
<b>in [17]: dataset 1</b>			
effort (1)	13.97	2.57	0.05
effort (2)	7.65	1.67	0.08
effort (3)	36.00	4.90	0.08
effort (4)	1.90	0.86	0.15
<b>in [17]: dataset 2</b>			
effort (1)	14.93	6.46	0
effort (2)	4.09	0.96	0.02
effort (3)	37.31	8.05	0.02
effort (4)	27.95	5.31	0.03

a – Pred(30)

**B. Identifying relationships most important predictors**

Figure 1 illustrates a part of the structure of the best performing BN. The best performing BN has been selected as the one with the lowest average of MMREs for all dependent variables. To improve the figure clarity only those variables have been included which are in within a Markov blanket for at least one dependent variable.

Two dependent variables (*productivity rate* and *defect rate*) are defined deterministically as the ratio of *effort* and *defects*, respectively, with *project size*. Unexpectedly, these relationships have not been identified in the generated BNs, apart from some partial exceptions, like relationship between *productivity rate* and *project size* in the case of this BN. It may suggest that such trivial relationships should be identified by experts and encoded in the model prior for learning the structure from the dataset.

Table III lists the predictors most frequently appearing (with counts) in the Markov blanket for each dependent variable. Dependent variables are also listed in rows as predictors because in some learned BNs they have been in Markov blanket for other dependent variable. The other predictors (real ones) have been listed in the order of the decreasing total frequency in Markov blankets in all BNs.

It can be observed that three predictors (*development staff motivation*, *project size* and *requirements stability*) have been the most frequent predictors for dependent variables. Surprisingly, project size have been a dominating predictor only for *productivity rate*, not for *effort* nor *number of defects*. This can be explained by the fact that the variability of *project size* end *effort* was generally low in the dataset.

*Number of defects* seem to have strong predictors identified – *requirements stability* appeared in each BN’s Markov blanket and *development staff motivation* appeared in 8 of 10 Markov blankets. No other dependent variable had such dominating predictors. However, it is the *number of defects* which had the lowest prediction accuracy among all four dependent variables. The explanation for this might be that, although these predictors have been the related to *number of defects* so often, they might have still not been highly statistically correlated with *defects*. This confirms a general difficulty of defect prediction.

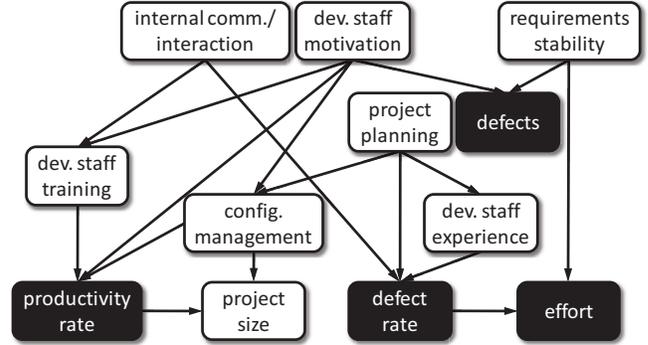


Figure 1. Schematic of best performing BN.

TABLE III. FACTORS MOST FREQUENTLY INFLUENCING DEPENDENT VARIABLES

Factor	Dependent variable			
	effort	productivity rate	defects	defect rate
effort	–	0	1	6
productivity rate	0	–	3	0
defects	1	3	–	0
defect rate	6	0	0	–
development staff motivation	7	1	8	4
project size	5	8	2	2
requirements stability	5	0	10	1
complexity of new functionality	3	0	6	0
project planning	1	4	0	1
development staff experience	0	2	2	2
vendor management	4	0	0	2

V. CONCLUSIONS

The research method with automatic generation of BNs from empirical data followed in this study is easy to use – it does not involve significant data preparation and analysis. The low volume of data used in model generation does not allow to exploit these BNs in other environments. But the relationships identified during model generation can still be a useful base for further studies.

The results of this experiment confirm that it is difficult to obtain highly accurate predictions from BNs built only from small empirical data. There are two general approaches which can be followed to solve this problem – both of them require stronger input from domain expert:

1. An expert prepares a BN topology while parameters are learnt automatically from data. It enables defining relationships between the variables that might have not been discovered by a structure learning algorithm.
2. BN structure and parameters are learnt automatically from available data, then an expert may adjust parameter definition. It reduces the problem arising from the fact that small local dataset may not be representative.

In future I plan to investigate the predictive accuracy of other methods using the same dataset and similar procedure, where possible. Also, part of the current study may be repeated in different setting, with effort and productivity assumed to be known, to simulate defect prediction at the end of development stage.

#### ACKNOWLEDGMENT

I would like to thank Prof. Marek Druzdzal (Univ. of Pittsburgh) and Prof. Norman Fenton (Queen Mary, University of London) for providing tools for developing and managing BNs: Genie and Agenarisk, respectively.

#### REFERENCES

- [1] Agenarisk, Agena, Ltd., <http://www.agenarisk.com>, 2009.
- [2] Bai C.G., Hu Q.P., Xie M., Ng S.H., Software failure prediction based on a Markov Bayesian network model, *J Systems and Software*, vol. 74(3), pp. 275–282, 2005.
- [3] Bayes T., An essay towards solving a Problem in the Doctrine of Chances. By the late Rev. Mr. Bayes, F.R.S. communicated by Mr. Price, in a letter to John Canton, A.M.F.R.S., *Phil. Trans. Royal Soc. London*, vol. 53, pp. 370–418, 1763.
- [4] Bibi S., Stamelos I., Software Process Modeling with Bayesian Belief Networks, *Proc. Int. Software Metrics Symposium*, Chicago, 2004.
- [5] Boetticher G., Menzies T., Ostrand T., PROMISE Repository of Empirical Software Engineering Data, West Virginia University, Department of Computer Science, <http://promisedata.org/repository>, 2010.
- [6] Cockram T., Gaining Confidence in Software Inspection Using a Bayesian Belief Model, *Software Qual. J.*, vol. 9(1), pp. 31–42, 2001.
- [7] Dabney J.B., Barber G., Ohi D., Predicting Software Defect Function Point Ratios Using a Bayesian Belief Network, *Proc. 2<sup>nd</sup> Int. Workshop on Predictor Models in Software Engineering*, Philadelphia, PA, 2006.
- [8] Darwiche A., *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009.
- [9] del Salgado Martinez J., del Aguina Cano I.M., A Bayesian Network for Predicting the Need for a Requirements Review, in: Meziane F., Vadera S. (eds.), *Artificial Intelligence Applications for Improved Software Engineering Development: New Prospects*, Information Science Reference, New York, 2008, pp. 106–128.
- [10] Fenton N., Hearty P., Neil M., Radliński Ł., Software Project and Quality Modelling Using Bayesian Networks, in: Meziane F., Vadera S. (eds.), *Artificial Intelligence Applications for Improved Software Engineering Development: New Prospects*, Information Science Reference, New York, 2008, pp. 1–25.
- [11] Fenton N., Marsh W., Neil M., Cates P., Forey S., Taylor M., Making Resource Decisions for Software Projects, *Proc. 26<sup>th</sup> Int. Conference on Software Engineering*, Washington, DC, IEEE Computer Society, pp. 397–406, 2004.
- [12] Fenton N., Neil M., Marsh W., Hearty P., Radliński Ł., Krause P., On the effectiveness of early life cycle defect prediction with Bayesian Nets, *Empir. Software Eng.*, vol. 13, pp. 499–537, 2008.
- [13] Fenton N.E., Neil M., Marsh W., Krause P., Mishra R., Predicting Software Defects in Varying Development Lifecycles using Bayesian Nets, *Inf. and Software Techn.*, vol. 43(1), pp. 32–43, 2007.
- [14] Genie, Decision Systems Laboratory, University of Pittsburgh, <http://genie.sis.pitt.edu/>, 2009.
- [15] Hearty P., Fenton N., Marquez D., Neil M., Predicting Project Velocity in XP using a Learning Dynamic Bayesian Network Model, *IEEE Trans. Software Eng.*, vol. 37(1), 124–137, 2009.
- [16] Jensen F.V., *An Introduction to Bayesian Networks*, UCL Press, London, 1996.
- [17] Mendes and N. Mosley, Bayesian Network Models for Web Effort Prediction: A Comparative Study, *IEEE Trans. Software Eng.*, vol. 34, 2008, pp. 723–737.
- [18] Mendes E. Using Bayesian Networks for Web Effort Estimation, in: Meziane F., Vadera S. (eds.), *Artificial Intelligence Applications for Improved Software Engineering Development: New Prospects*, Information Science Reference, New York, 2008, pp. 26–44.
- [19] Mendes E., The Use of Bayesian Networks for Web Effort Estimation: Further Investigation, *Proc. 8<sup>th</sup> Int. Conf on Web Engineering*, Yorktown Heights, NJ, pp. 203–216, 2008.
- [20] Pai G., Bechta-Dugan J., Lateef K., Bayesian Networks applied to Software IV&V, *Proc. 29<sup>th</sup> Annual IEEE/NASA Software Engineering Workshop*, IEEE Computer Society, Washington, DC, pp. 293–304, 2005.
- [21] Pai G.I., Dugan J.B., Empirical Analysis of Software Fault Content and Fault Proneness Using Bayesian Methods, *IEEE Trans. Software Eng.*, vol. 33(10), pp. 675–686, 2007.
- [22] Pearl J., Bayesian Networks: A Model of Self-Activated Memory for Evidential Reasoning. (UCLA Technical Report CSD-850017). *Proc. 7<sup>th</sup> Conf. of the Cognitive Science Society*, University of California, Irvine, pp. 329–334, 1985.
- [23] Pearl J., *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, 1988.
- [24] Radliński Ł., Fenton N., Neil M., Marquez D., Improved Decision-Making for Software Managers Using Bayesian Networks, *Proc. 11<sup>th</sup> IASTED Int. Conf. Software Engineering and Applications*, Cambridge, MA, pp. 13–19, 2007.
- [25] Radliński Ł., Fenton N., Neil M., A Learning Bayesian Net for Predicting Number of Software Defects Found in a Sequence of Testing, *Pol. J. Env. Stud.*, vol. 17(3B), pp. 359–364, 2008.
- [26] Radliński Ł., Predicting Defect Types in Software Projects, *Pol. J. Env. Stud.*, vol. 18(3B), pp. 311–315, 2009.
- [27] Stewart B., Predicting project delivery rates using the Naive–Bayes classifier, *J. Software Maint. Evol.: Res. Pract.*, vol. 14, pp. 161–179, 2002.
- [28] Wagner S., A Bayesian network approach to assess and predict software quality using activity-based quality models, *Proc. 5<sup>th</sup> Int. Conf. on Predictor Models in Software Engineering*, New York, ACM Press, 2009.
- [29] Wooff D.A., Goldstein M., Coolen F.P.A., Bayesian Graphical Models for Software Testing, *IEEE Trans. Software Eng.*, vol. 28(5), pp.510–525, 2002.
- [30] Zhou Y., Würsch M., Giger E., Gall H.C., Lü J., A Bayesian Network Based Approach for Change Coupling Prediction, *Proceedings of the 15th Working Conference on Reverse Engineering*, Washington, DC, IEEE Computer Society, pp. 27–36, 2008.

# Multi-Tracker Collaboration in Bittorrent Systems

Sonia Gulrajani, Anuja Oka, and Xiao Su

Department of Computer Engineering  
San Jose State University  
San Jose, CA 95192, USA

Email: {soniasainani, anujaoka}@gmail.com, xiao.su@sjsu.edu

**Abstract**—While BitTorrent has fast gained popularity and is being extensively used to share files among users, its performance is limited by the fact that independent torrent networks for the same file are unable to collaborate. This can result in starved torrent networks, due to the incognizance of the existing active torrent networks for the same files. One way to address this problem is to have the trackers of such torrent networks collaborate with each other by exchanging peer lists, thereby resulting in an extended peer pool. Along with helping the starved networks, multi-tracker collaboration can also enhance torrent performance, with respect to download speed by bringing together independent healthy torrent networks for the same files and creating a virtual common torrent network. In this paper we propose to introduce multi-tracker collaboration into BitTorrent protocol, and study its performance gain by implementing it in the popular BNBT tracker software.

*Keywords* – peer-to-peer, file sharing, incentives, collaborative distribution.

## I. INTRODUCTION

BitTorrent is a very popular and widely used P2P file sharing protocol. It facilitates fast transfer of files among participating peers across unreliable networks. It has a distinct advantage over the traditional client-server architecture as it enables transferring huge amount of content without imposing tremendous bandwidth requirements and costs on the server.

BitTorrent is based on the cooperative distribution principle and hence is highly scalable as new participants bring in not only demand, but also supply. But with this scalability, BitTorrent systems become very dynamic, with many peer joins and departures. High peer churn rates can lead to starvation periods during the torrent lifetime. In case that many peers leave the system, it is possible that some data pieces are missing in the torrent, therefore, data exchange rate gradually drops to zero. The torrent becomes starved and cannot progress further unless peers with missing pieces join the torrent. This time period can be

indefinite and hence reflects poorly on the robustness and performance of the torrent.

The objective of our work is to propose mechanisms to revive starved torrents and improve the performance of torrents with few seeder peers. In our approach, we propose to facilitate collaboration between trackers that track the same files. This multi-tracker collaboration results in an increased peer pool and thereby deterministically revives starved torrents and increases download rates for torrents with few seeders.

The rest of the paper is organized as follows. In Section II, we provide an introduction to BitTorrent protocol and system. In Section III, we briefly discuss the motivation behind multi-tracker collaboration. Next in Section IV, we present the design and implementation of a BitTorrent system featuring multi-tracker collaboration. In Section V, we present and analyze the experimental results. Section VI and I conclude the paper.

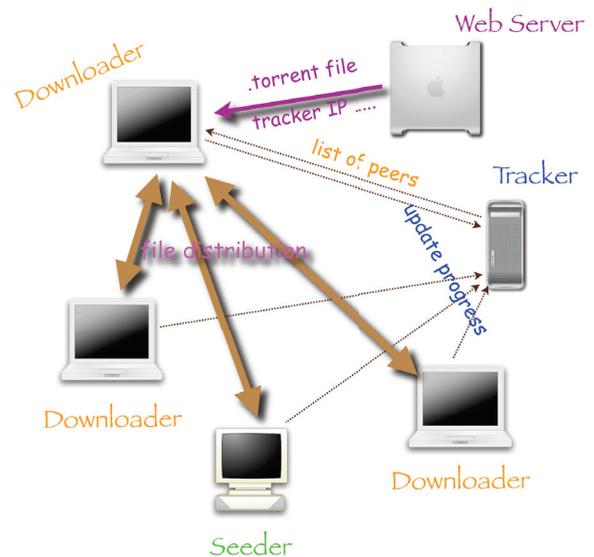


Figure 1. A BitTorrent system.

## II. BITTORRENT SYSTEM AND PROTOCOL

In earlier file sharing applications, peers request and obtain a complete file each time. In BitTorrent, a file is divided into equal-sized pieces. Peers in the system can request and transmit different file pieces at the same time, leading to higher system throughput and shorter file request time.

As shown in Figure 1. , a BitTorrent system consists of a web server hosting torrent files, a tracker, and participating peers. There is a separate torrent for each file to be distributed. When a peer is interested in a particular torrent, it first obtains the information about the torrent (stored in .torrent file) from a web server. The torrent file describes metainfo about the file being exchanged, including file name, total length, piece length, hash of file pieces, and most importantly, the URL of the tracker for this torrent.

Using the URL found in .torrent file, the peer contacts the tracker to obtain the IP addresses of active peers in the system. Finally, the peer interacts with these peers to obtain missing file pieces and to upload available file pieces. To keep the tracker up-to-date with their progresses, the peers in the torrent periodically send update packets to the tracker.

The BitTorrent protocol specification [2] consists of two parts: a tracker HTTP/HTTPS protocol used by a peer to communicate with the tracker and a peer wire protocol used by a peer to exchange file pieces with fellow peers in the torrent. A tracker listens at a HTTP service, responding to HTTP GET requests from the peer. In a peer's tracker request, it notifies the tracker with its total number of uploaded bytes, downloaded bytes, together with other statistics to help the tracker keep track of overall progress of the torrent. In the tracker's response packet, the tracker sends a list of peers, who are actively participating in the torrent.

In the BitTorrent system, a peer becomes a seeder, when it finishes downloading all the file pieces. A peer still in the process of downloading is called a leecher. Therefore, to keep a torrent system going, it is important to have a good percentage of seeders, who are willing to contribute their file pieces.

## III. SYSTEM BOTTLENECK

Measurement studies on BitTorrent [2][7][8] revealed two bottlenecks in the system: tracker and seeder availability.

Tracker, a centralized component in the system, is vulnerable to be a single point of failure. The work in[2] analyzed a one-month long BitTorrent trace and found that as high as 60% torrents failed because the tracker went offline. There have been multiple proposals addressing this problem. Distributed Hash

Table (DHT) [9][10][11] works around this problem by eliminating the need for the central tracker itself. It distributes the tracker functionality amongst the peers, making them lightweight trackers. Other schemes to address tracker failover management include developing backup tracker networks [1]. Slurpie[4], an enhanced collaborative bulk data transfer protocol, uses random backoff to alleviate load the centralized server.

At least one seeder peer has to be present for the BitTorrent system to make progress, as the seeder possesses every piece in the file and can satisfy the leechers' requests for missing pieces. However, there is no incentive for peers to stay in the system after they complete their downloading. As time evolves, a torrent is very likely to suffer from starvation, after seeders have gradually left the torrent. While there are solutions to make the tracker more robust and reliable, relatively little work has been done towards reviving starved torrents. In the literature, theoretical modeling and analysis work [5][6] exist, and they advocate improving the collaboration among torrents; however, no practical systems have been developed.

In this paper, we will address the second bottleneck, *i.e.*, insufficient number of seeders in the system, by designing a BitTorrent protocol to facilitate communication among trackers and to expand the potential seeder pool in multiple torrents.

TABLE I. EXAMPLE TORRENTS

	Without collaboration		With collaboration	
	T1	T2	T1	T2
Peer 1	leecher		leecher	seeder
Peer 2	leecher		leecher	
Peer 3	leecher	seeder	leecher	seeder
Peer 4	seeder	leecher	seeder	leecher
Peer 5		leecher	seeder	leecher
Peer 6		leecher		leecher

## IV. MULTI-TRACKER COLLABORATION

### A. An Illustrative Example

First, let us motivate our proposal by studying two torrents, T1 and T2, which have some overlapping peers. TABLE I. lists the leecher and seeder peers in T1 and T2. T1 consists of three leechers - Peer 1, 2, and 3, and one seeder - Peer 4, while T2's leechers include Peer 4, 5, 6 and its seeder is Peer 3. Without inter-torrent collaboration, T1 and T2 progress independently. In case seeder Peer 4 leaves T1 (or seeder Peer 3 leaves T2), then T1 (or T2)'s progress will be stalled until some new seeder peers join to revive the torrent.

Imagine the scenario that Peer 1 in T1 has previously completed downloading the file served in T2, and Peer 5 in T2 has the file served in T1. While Peer 1 is online downloading file pieces in T1, it can serve as a seeder peer in T2! The same reasoning applies to Peer 5, for it can work as a seeder in T1. With such collaboration between T1 and T2, both torrents are now served by two seeders, thus the loss of one peer will not make the torrent starved anymore.

Note that Peer 1 and Peer 5 are already online participating in a torrent, requiring them to join another live torrent as a seeder is not too demanding on the peers. However, tracker in T1 and T2 need to communicate with each other, so that peers in T1 learn that Peer 5 can be a seeder and peers in T2 learn that Peer 1 can be a seeder. Current BitTorrent protocol does not support such communication among trackers.

### B. Our Algorithm

In order to support multi-tracker collaboration in existing BitTorrent trackers, we propose an extension to the existing BitTorrent protocol. The extension involves introducing a new tracker-tracker communication protocol. As part of this protocol, each tracker provides tracker collaboration service to other participating trackers. The communication between trackers involves introduction of a new message type, tracker-tracker announce, which has been designed similar to the "Announce" messages exchanged between the peers and the tracker. The details of the new message - "Tracker Announce" - are given below [1].

**Request:** To contact another tracker, a tracker sends a standard HTTP GET request to the other tracker's URL for collaboration service. The GET request is parameterized as specified in the HTTP protocol. The following parameters have been added to the request message:

'info\_hash': This is a required 20-byte SHA1 value, which identifies the torrent for which the tracker wants to collaborate with the other tracker.

**Response:** Upon receiving the HTTP GET request, the tracker responds with a document having the "text/plain" MIME type. This document must contain a bencoded dictionary with the following keys:

'failure code': This key is optional. If present, the dictionary must not contain any other keys. The tracker interprets this message as a failure to collaborate with the other tracker. This may happen if the other tracker does not provide tracker collaboration service, or if it does not host the torrent requested.

'interval': The tracker sends regular HTTP GET requests to the other tracker to get the updated peer list.

This value indicates the amount of time the client tracker must wait before sending the next GET request.

'peers': This is a bencoded list of dictionaries containing a list of peers for the info hash in the request. It has the following structure:

'peer id': This is the self-designated ID of the peer.

'ip': This is the IP address of the peer.

'port': This is the port of the peer on which it should be contacted.

Tracker collaboration kicks in when the tracker gets an "announce" request from the first leecher for a hosted file. The tracker contacts the trackers providing tracker collaboration service by sending the "tracker-tracker announce" request. From the responses received, it builds a list of unique peers from collaborative torrents for the hosted torrent file. The tracker provides this list of peers to all the leechers joining the network. The tracker periodically sends the "tracker-tracker announce" request to the collaborating trackers based on the 'interval' received in the response to refresh the maintained peer list.

### C. System Architecture

Our design has leveraged a popular tracker implementation, BNBT tracker [1]. The block diagram in Figure 2. shows the various components of the BNBT server with support for multi-tracker collaboration.

The central part of collaboration is implemented in BNBT server, which is responsible for providing the tracker service to incoming clients (peers), and collaborating with other trackers.

The **Client Processing module** handles the incoming requests from the clients and sends responses for the received requests.

The **Incoming Tracker Processing module** is responsible for communicating with the trackers contacting the BNBT server. It processes the requests received from the trackers, and provides responses for the received requests.

The **Outgoing Tracker Processing module** monitors the health of existing torrent networks, and contacts configured trackers for peer lists, if it detects starved torrent networks.

BitTorrent uses a platform independent way of encoding data, namely Bencoding. The **Message Parser** is responsible for encoding and decoding BitTorrent messages.

The **Message Handler** is responsible for processing the decoded BitTorrent messages. Each message type is handled by a separate module in the Message handler, named after the message type. For e.g., Tracker announce messages are handled by **Tracker Announce module**.

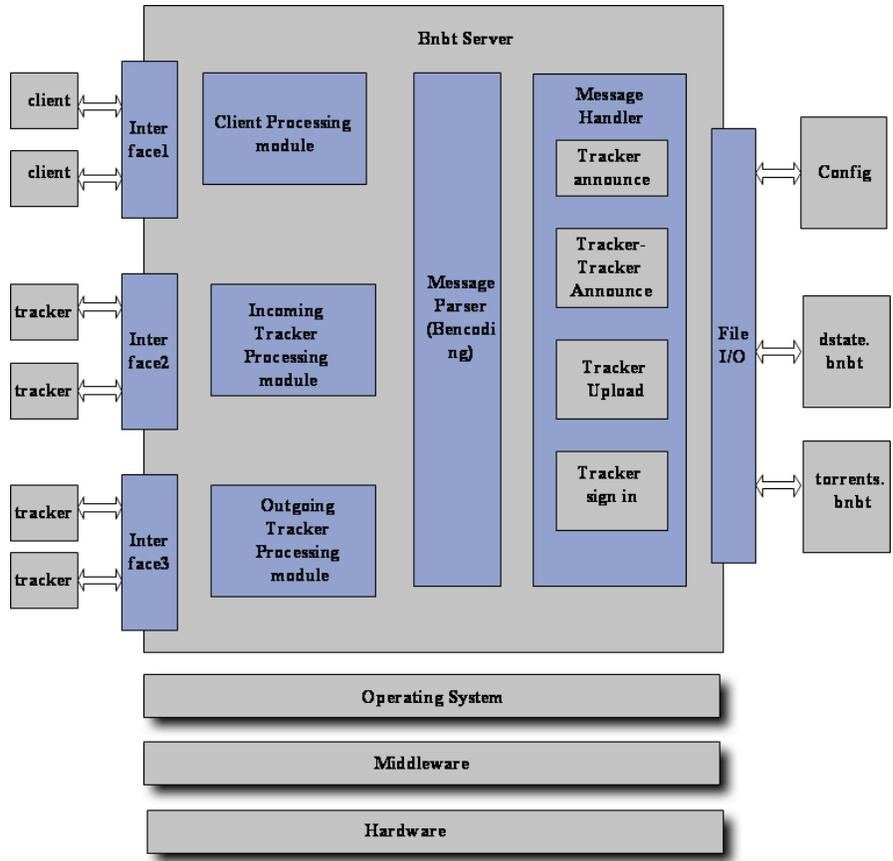


Figure 2. System architecture of a BNBT server with tracker collaboration

## V. EXPERIMENTAL RESULTS

### A. Experimental Setup

We performed all the experiments on PlanetLab nodes spread across the globe. The experiments aimed to compare the improvement in the download rates experienced by the peers with and without multi-tracker collaboration. We created two representative torrent networks: one managed by a tracker in USA and the other managed by a tracker in South Korea. The detailed setup for the torrent networks is given in TABLE II, Figure 3, TABLE III, and Figure 4.

TABLE II. TORRENT 1 TOPOLOGY

Role	IP address: port	location
Tracker	128.112.139.72:6969	USA
Seeder	150.65.32.68:30000	Japan
Leechers	141.22.213.35:10000	Germany
	128.112.139.75:20000	USA
	128.112.139.71:50000	USA
	192.33.90.66:50000	Switzerland

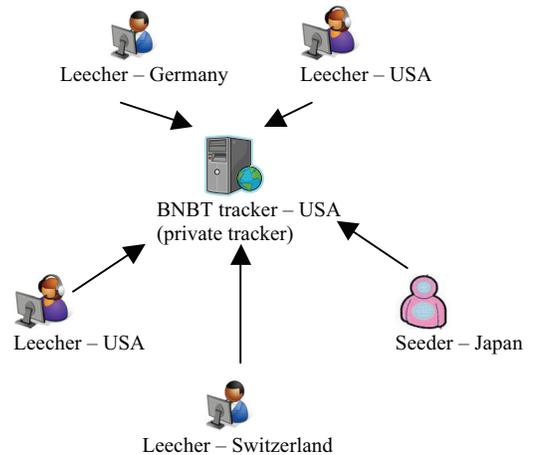


Figure 3. Torrent Network 1

Number of Seeders = 1  
 Number of Leechers = 4  
 Peer pool size = 5  
 Torrent size = 5

TABLE III. TORRENT 2 TOPOLOGY

Role	IP address: port	location
Tracker	210.125.84.15:6969	S. Korea
Seeders	128.112.139.108:20000	USA
	128.112.139.96:30000	USA
	193.167.182.130:10000	Finland
leechers	193.136.166.54:20000	Portugal
	150.65.32.68:40000	Japan

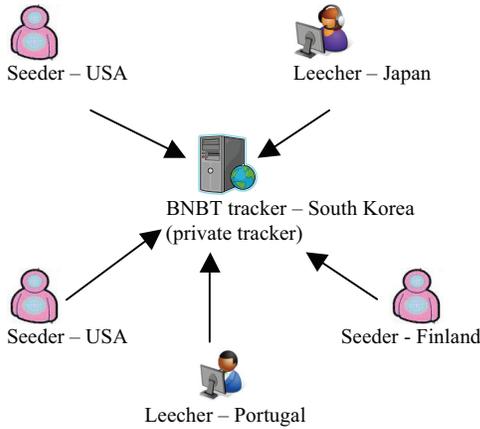


Figure 4. Torrent Network 2

Number of Seeders	=	3
Number of Leechers	=	2
Peer pool size	=	5
Torrent size	=	5

### B. Reviving a Starved Torrent

In our first experiment, we evaluated the revival of starved networks. In order to make torrent network 1 (in Figure 3.) starve, the sole seeder in the torrent network was brought down.

TABLE IV. compares the results of two cases: (1) torrent network 1 is isolated; and (2) torrent network 1 and 2 join together.

As seen from the table, without tracker collaboration, a leecher in torrent network 1 was not able to download the file. After enabling multi-tracker collaboration in case (2), the peer successfully completed its download because of the increased peer list provided by tracker 2, including three seeders.

### C. Measuring the Performance of a Leecher

In our second experiment, we measured the performance in healthy torrent networks, where there are one or more seeder peers available.

TABLE V. shows the experimental results for two cases: the left column shows the results when torrent network 1 stands alone and the right column shows the results when torrent network 1 and 2 work together.

TABLE IV. REVIVING A STARVED TORRENT.

Parameter	Multi-tracker collaboration absent	Multi-tracker collaboration present
Torrent size	4	4
No. of seeders	0	3
No. of Leechers	4	6
Ratio of seeders:Leechers	0 : 4	3 : 6
Peer pool size	4	9
Piece Length	256Kb	256Kb
Time taken by peer to download file ( file size : 2MB)	$\infty$	8 min
Time taken by peer to download file ( file size : 175MB)	$\infty$	426 min
Time taken by peer to download file ( file size : 1.2GB)	$\infty$	2020 min

TABLE V. IMPROVED DOWNLOADING RATE FOR A LEECHER

Parameter	Multi-tracker collaboration disabled	Multi-tracker collaboration Enabled
Torrent size	5	5
No. of seeders	1	4
No. of Leechers	4	6
Ratio of seeders:Leechers	1 : 4	4 : 6
Peer pool size	5	10
Piece Length	256Kb	256Kb
Time taken by peer to download file ( file size : 2MB)	8 min	8 min
Time taken by peer to download file ( file size : 175MB)	453 min	425 min
Time taken by peer to download file ( file size : 1.2GB)	2112 min	2023 min

We evaluated their performances for three different file sizes: 2MB, 175MB, and 1.2GB. As seen from the table, when the file size is not small, the time taken to

download the same file is shorter when multi-tracker collaboration is present, compared to the time taken in a standalone torrent network.

Currently, the internal incentive mechanisms work in a way that a peer in a torrent network gets rewarded with more downloading bandwidth when it contributes and uploads more packets to other peers in the same torrent network. We haven't modified the incentive schemes when peers in two torrent networks collaborate together.

Ideally, when a peer A contributes more bandwidth to peers in the torrents that are collaborating together, whether the peers are inside his own small torrent network or belong to other collaborative torrents, peer A should be rewarded by more downloading bandwidth. When this incentive algorithm is introduced into the collaborative torrent networks, we will expect to see much bigger improvements in peers' downloading time with torrent collaboration.

## VI. FUTURE ENHANCEMENTS

In the current experiments conducted, each tracker had a configured list of trackers participating in multi-tracker collaboration. This kind of static configuration is not a practical option in real world scenarios. This can be made more feasible by one of the following options:

**Auto-discovery of trackers:** This technique requires the BitTorrent client to store the addresses of the trackers it has come across in its lifetime. The client can share this information with the tracker as part of the 'announce' message or a new message type. The tracker can then use this information to build up a database of trackers supporting multi-tracker collaboration, thus eliminating the need of any kind of pre-configuration. This is a heavy-duty solution, as it requires extensions to the BitTorrent protocol, as well client and tracker software.

**Centralized directory of trackers:** This technique requires maintaining a central repository of trackers participating in multi-tracker collaboration. The central directory is built by each participating tracker registering with it.

## I. CONCLUSION

In this paper, we have proposed to introduce tracker collaboration in BitTorrent protocol. Multi-tracker collaboration not only revives starved torrent networks, but also boosts performance in healthy torrent networks by extending the peer pool available to BitTorrent peers. Being client agnostic, multi-tracker collaboration feature can be incorporated in existing BitTorrent networks by upgrading the tracker software only.

## REFERENCES

- [1] BNBT EasyTracker, <http://bnbteasytracker.sourceforge.net/>.
- [2] BitTorrent Protocol, [http://www.bittorrent.org/beps/bep\\_0003.html](http://www.bittorrent.org/beps/bep_0003.html).
- [3] J. A. Pouwelse, P. Garbacki, D. H. J. Epema, and H. J. Sips. The bittorrent p2p file-sharing system: Measurements and analysis. In Proceedings of 4<sup>th</sup> International Workshop on Peer-to-Peer Systems (IPTPS), 2005.
- [4] R. Sherwood, R. Braud, B. Bhattacharjee. Slurpie: a cooperative bulk data transfer protocol. In Proceedings of 23th IEEE International Conference on Computer Communications (INFOCOM 2004), pp. 941-951 vol.2, 2004.
- [5] Lei Guo, Songqing Chen, Zhen Xiao, Enhua Tan, Xiaoning Ding, and Xiaodong Zhang. A performance study of bittorrent-like peer-to-peer systems. IEEE Journal on Selected Areas in Communications, vol.25, No.1, pp. 155-169, 2007.
- [6] Dongyu Qiu and R. Srikant. Modeling and performance analysis of bittorrent-like peer-to-peer networks. SIGCOMM Computer Communications Review, vol.34., No. 4, pp.367-378, October 2004.
- [7] Y. Tian, D. Wu, and K. W. Ng. Modeling, analysis and improvement for bittorrent-like file sharing networks. In Proceedings of 25th IEEE International Conference on Computer Communications (INFOCOM 2006), pp.1-11, April 2006.
- [8] M. Izal, Guillaume Urvoy-Keller, Ernst W. Biersack, P. A. Felber, Al, and L. Garces-Erice. Dissecting bittorrent: Five months in a torrent's lifetime. In Proceedings of 5<sup>th</sup> International Workshop on Passive and Active Network Measurement, pp.1-11. 2004.
- [9] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM), volume 31, pp. 149-160, New York, NY, USA, October 2001. ACM.
- [10] Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph, and John D. Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. IEEE Journal on Selected Areas in Communications, 22(1):41-53, 2004.
- [11] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. Lecture Notes in Computer Science, 2218:329-350, 2001.

# An Evaluation of Tie-Breaking Strategies for Fault Localization Techniques

Xiaofeng Xu<sup>a,b,†</sup>  
xxfeng@xmu.edu.cn

Vidroha Debroy<sup>b</sup>  
vxd024000@utdallas.edu

W. Eric Wong<sup>b</sup>  
ewong@utdallas.edu

Donghui Guo<sup>a,c,\*</sup>  
dhguo@xmu.edu.cn

<sup>a</sup>Department of Physics, Xiamen University, China

<sup>b</sup>Department of Computer Science, The University of Texas at Dallas, USA

<sup>c</sup>School of Information Science and Technology, Xiamen University, China

## Abstract

*Software fault localization techniques typically rank program components (such as statements or predicates) in descending order of their suspiciousness (likelihood of being faulty). During debugging, programmers may examine these components, starting from the top of the ranking, in order to locate faults. However, the assigned suspiciousness to each component may not always be unique, and thus, some of them may be tied for the same position in the ranking. In such a scenario, the total number of components that a programmer needs to examine in order to find the faults may vary considerably – as in the best case one would examine a faulty component before all other components that are tied for the same rank; and in the worst case, only after. The greater the variability, the harder it is for a programmer to decide which component to examine first, and the harder it is to accurately compute the expected effectiveness of a fault localization technique. Our first case study (based on three fault localization techniques across four sets of programs) reveals that the phenomenon of assigning the same suspiciousness to multiple components is not limited to any technique or program in particular. Thus, to reduce variability and alleviate this problem, four tie-breaking strategies are evaluated empirically in our second case study across the programs of the Siemens suite, using the Tarantula fault localization technique. Results indicate that some strategies can not only reduce the number of ties in the rankings but also improve the average effectiveness of Tarantula.*

**Keywords:** program debugging, fault localization, critical ties, suspiciousness

## 1. Introduction

Among all the program debugging activities, software fault localization (hereafter referred to simply as fault localization) has been recognized to be one of the most expensive [7]. This realization has catalyzed the proposal and development of many fault localization techniques over the recent years [1,2,3,5,6,8,9,10,11,12]. Several such techniques use execution traces collected at runtime during program testing as well as test execution results (i.e., whether an execution succeeds or fails) to compute the suspiciousness of each statement<sup>1</sup>. This ‘suspiciousness’ represents the likelihood of

the corresponding statement being faulty. Once the statements have been ranked in descending order of their suspiciousness (from most suspicious to least), they are examined one-by-one starting from the top of the ranking until a faulty statement is identified. A good fault localization technique should place a faulty statement towards the top of its ranking (if not at the very top). Ideally, a faulty statement should have a much higher suspiciousness than a non-faulty statement.

However, an important observation is that the suspiciousness of a statement assigned by a fault localization technique is not always unique when compared with the suspiciousness of other statements. Multiple statements may be assigned the same suspiciousness and therefore, are tied for the same position in the ranking.<sup>2</sup> The existence of such tied statements implies that the fault localization technique in question cannot distinguish these statements from one another in terms of their likelihood of being faulty. Programmers are offered no guidance on what to examine first, given a set of statements that are tied, and are thus, forced to rely on their own judgment or intuition.

For the discussion purpose, let’s consider the case where in a ranking there is a group of statements with the same suspiciousness and only one of them is faulty. In the very best case, the programmer shall examine the faulty statement first and there will be no need to examine other statements in the group (as we would stop examining after finding the fault). But in the worst case, we would examine all non-faulty statements in the group before we could reach the faulty statement. Thus, in terms of the number of statements that need to be examined to find the fault, we have two different levels of effectiveness – the *best* and the *worst*. The actual effectiveness may be anywhere in between the best and the worst. The larger the gap between these two effectiveness (i.e., the more statements that are tied with the faulty statement), the more difficult it is to accurately estimate what the actual effectiveness would be.

Thus, ties are undesirable in a ranking of program statements. The immediate question then arises, as to how these ties might be broken such that the problem mentioned above is alleviated, and yet the effectiveness of a fault

---

<sup>†</sup>Xiaofeng Xu is an exchange PhD student at the University of Texas at Dallas under the supervision of Professor W. Eric Wong. This research is sponsored by the China Scholarship Council (CSC) and the University of Texas at Dallas.

---

<sup>1</sup> Fault localization techniques can be used to locate different types of faulty program components. For the purposes of this paper we consider program components to be statements with the understanding that they could have just as easily been other components such as functions, blocks, predicates, etc.

<sup>2</sup> Refer to Section 2.1 for a more detailed discussion on tied statements.

localization technique is not affected adversely. To this effect, four tie-breaking strategies are discussed and evaluated. We take this opportunity to point out that for the purposes of this paper, ties between groups of statements, where none of them is faulty, are considered immaterial. Further discussion on this is presented in Section 2.1.

Another important question that comes up is whether ties between statements are a common occurrence, and whether they are prevalent regardless of which fault localization techniques or subject programs are used. If a faulty statement is not expected to be tied with other non-faulty statements, then why would we need to worry about ties? To address such concerns, experiments were performed using three fault localization techniques (Tarantula [3], Ochiai [1], and Heuristic III [9]) across four sets of programs (the Siemens Suite, and the `grep`, `gzip` and `make` programs). Results indicate that ties involving faulty statements are quite frequent, and, they occur regardless of the choice of fault localization technique or subject program.

The contributions of the paper can be summarized as:

- 1) Using three fault localization techniques – Tarantula [3], Ochiai [1], and Heuristic III [9] on several different programs (the Siemens suite, `grep`, `gzip` and `make`), the existence, and scale of the problems involving ties is revealed. Furthermore, we show that this is not a concern limited to any technique or subject program in particular.
- 2) Four strategies are proposed to break ties. Empirical data shows that some of the strategies can break ties among statements to a considerable extent, yet do so without adversely affecting the effectiveness of the techniques.

The remainder of this paper is organized as follows: Section 2 describes in detail the problem that is addressed in this paper, and describes the fault localization techniques and subject programs evaluated herein. Empirical evidence is presented to illustrate the scale of the problem. Section 3 presents four tie-breaking strategies designed to alleviate the problem, and they are subsequently evaluated in Section 4. Section 5 discusses our threats to validity. Related work is reviewed in Section 6, and we present our conclusions in Section 7.

## 2. Illustrating the Problem

In this section we further discuss ties among statements in the rankings that are produced by fault localization techniques, and investigate, via case studies, the degree to which this may occur. For each of the fault localization techniques that are used, a succinct overview is also provided.

### 2.1 Ties and Critical Ties

There is a subtle distinction between the types of ties that may occur among statements in a ranking. While it is possible that a faulty statement may be tied with non-faulty statements; it is also quite likely that non-faulty statements are tied with one another for the same position in a ranking. The former scenario is of greater concern to us than the latter. To better understand why, let us again consider a

hypothetical ranking where only one of the statements is faulty. Given that there may be a set of statements  $S$  none of which is faulty, yet they are all assigned the same suspiciousness, we can construct two scenarios: First, it is possible the statements in  $S$  have a higher suspiciousness than the actual faulty statement. In this case, every statement in  $S$  will have to be examined before the faulty statement regardless of the internal order in which the statements of  $S$  are examined. Thus, the breaking of ties in the case of statements in  $S$  is of no immediate gain with respect to the total fault localization effectiveness (in terms of the number of statements that must be examined to find the fault).

Now consider the case where the statements in  $S$  are assigned a suspiciousness that is less than that of the faulty statement. In this scenario, the faulty statement will be examined before the statements in  $S$ ; and once we have located the faulty statement, we have no reason to continue examining the ranking. Therefore, again the internal order of examination of the statements in  $S$  is irrelevant with regards to the fault localization effectiveness. The effectiveness in fact, only undergoes change when ties are broken with respect of a group of statements (all of which have the same assigned suspiciousness) that contains the faulty statement. Thus, even though the number of ties for groups that do not contain the faulty statement may have been altered, when evaluating the tie-breaking strategies (which shall be presented in Section 3), we disregard the change in ties for such groups, and focus on tie-breaks among groups that contain faulty statements. To facilitate subsequent discussion, we present two definitions:

**Definition 1:** Ties. A Tie  $\mathcal{T}$  is defined as a set of statements, each of which has been assigned the same suspiciousness (and therefore, share the same position in a ranking) with respect to the fault localization technique used.

**Definition 2:** Critical Ties and critically tied statements: A Critical Tie  $\mathcal{CT}$  is a tie which contains a faulty statement. The statements in a critical tie are called critically tied statements. Note, that under the assumption that there is only one faulty statement in a program, there can be only one critical tie.

### 2.2 Fault Localization Techniques

We utilize three fault localization techniques for our experiments – Tarantula [3], Ochiai [1], and Heuristic III [9]. Each one of these techniques was chosen with a specific reason in mind. Tarantula because it is simple and well known, and Ochiai and Heuristic III because they have reported better fault localization effectiveness than Tarantula, yet are quite different with one another in terms of their construction. Further discussion on our choice of fault localization techniques is presented in Section 5.

**Tarantula** thus, assigns a suspiciousness value to each statement based on the following formula:

$$susp(s) = \frac{\frac{failed(s)}{totalfailed}}{\frac{failed(s)}{totalfailed} + \frac{passed(s)}{totalpassed}} \quad (1)$$

Where  $susp(s)$  is the suspiciousness of statement  $s$ ;  $passed(s)$  and  $failed(s)$  are the number of successful and failed test cases that execute  $s$  respectively; and  $totalpassed$  and  $totalfailed$  are the number of successful and failed test cases respectively.

**Ochiai** assigns a suspiciousness value to each statement based on the following formula:

$$susp(s) = \frac{failed(s)}{\sqrt{totalfailed \times (failed(s) + passed(s))}} \quad (2)$$

Where  $susp(s)$ ,  $passed(s)$ ,  $failed(s)$ , and  $totalfailed$  are the same as in Equation (1).

**Heuristic III**, assigns a suspiciousness value to each statement based on the following formula:

$$susp(s) = \sum_{i=1}^{S_F} w_{Fi} \times n_{Fi} - \sum_{i=1}^{S_S} w_{Si} \times n_{Si} \quad (3)$$

Where  $S_F$  and  $S_S$  is the number of failed and successful groups,  $n_{Fi}$ ,  $n_{Si}$  is the number of failed and successful tests in group  $i$ , and  $w_{Fi}$ ,  $w_{Si}$  is the contribution of failed and successful groups  $i$  respectively. As per the experiments in [9], we use  $S_F = S_S = 3$ ,  $n_{F1} = n_{S1} = 2$ , and  $n_{F2} = n_{S2} = 4$ , implying the third failed group contains all the remaining failed tests, and the third successful group contains all the remaining successful tests. Also we set  $w_{F1} = 1$ ,  $w_{F2} = 0.1$ ,  $w_{F3} = 0.01$ ,  $w_{S1} = 1$ ,  $w_{S2} = 0.1$  and  $w_{S3} = \alpha \times \chi_{F/S}$ , where  $\alpha = 0.001$  and  $\chi_{F/S}$  is the ratio of total failed and total successful test cases. Readers interested in further details of this technique are referred to [9].

## 2.3 Subject Programs

Four sets of programs are used for these experiments – the programs of the Siemens suite, and the grep, gzip, and make programs. For further details regarding the programs (including where they were downloaded from), and for data collection and environment information, readers are referred to Section 3.3 of [9]. Each faulty version used contains only one fault. For a discussion on programs with multiple faults, please refer to Section 5.

## 2.4 The Scale of the Tie-Related Problems

Prior to coming up with a solution to a problem, one must first decide whether the problem itself is worth fixing. Recall our discussion from Section 1, where we asked ourselves if ties in the rankings (critical ties for our purposes) were even worth breaking. To answer this, we need to determine the scale of the tie-related problems. The term ‘scale’ is used in a general sense and encompasses multiple factors such as the frequency of occurrence of critical ties; the size of a critical tie, etc. Also investigated is the question of whether the presence or absence of critical ties has something to do with the choice of fault localization technique, or the subject programs under study. We address these issues through empirical data collected using the fault localization

techniques in Section 2.2 across (all the faulty versions of) the subject programs in Section 2.3. Figures 1 to 4 (corresponding to the Siemens suite, gzip, grep and make programs respectively) present this data in the form of box plots. Each of the box plots depicts the chosen fault localization techniques on the x-axis and provides the ratio of – the size of the critical tie<sup>3</sup>, to the total number of executable statements in the program – on the y-axis; with respect to each of the techniques. They thus, present the fraction of the program that is critically tied.

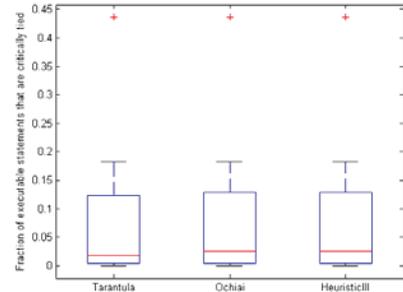


Figure 1. Fraction of program that is critically tied for the Siemens suite

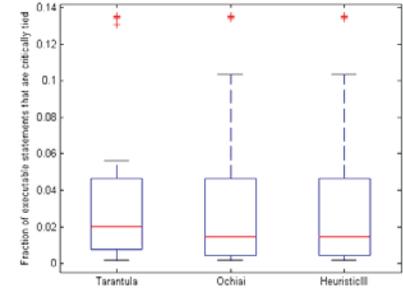


Figure 2. Fraction of the program that is critically tied for gzip

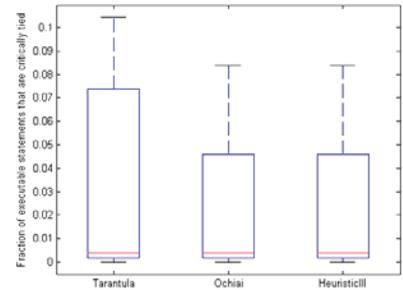


Figure 3. Fraction of the program that is critically tied for grep

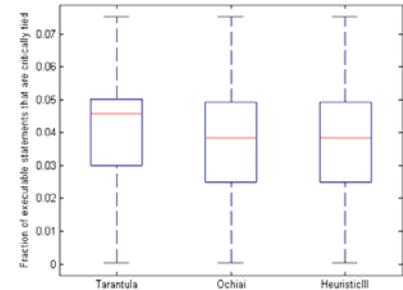


Figure 4. Fraction of the program that is critically tied for make

<sup>3</sup> Note, that we only have one critical tie per faulty version as each of the faulty versions studied has a single fault in it.

We are immediately able to draw several conclusions based on the box plots. Firstly, we observe that all of the studied fault localization techniques do indeed result in critical ties, and furthermore, this is true regardless of which subject program is under consideration. With regard to the scale or significance of the critical ties, we observe that on the Siemens suite programs (Figure 1), the maximum recorded fraction is almost as high as 20%, which means that on at least one of the faulty versions, almost 20% of the code (executable statements) is critically tied. This is the case for all three of the fault localization techniques studied (i.e., Tarantula, Ochiai, and Heuristic III). Furthermore, if we consider the maximum outliers, then we observe that almost 45% of the code can be critically tied. In fact, in the case of the Siemens suite programs (regardless of which fault localization technique is used), only 26 of the 129 faulty versions (i.e., 20.16%) are free of critical ties. Critical ties are similarly observed in case of the `gzip`, `grep`, and `make` programs as well, even though the scale does not seem as significant as in the case of the Siemens suite. This illustrates that critical ties are not a rare occurrence, and when they do occur, a significant portion of the code might be critically tied. Thus, this justifies the need for tie-breaking strategies.

However, we note that for the most part, the box plots for the fault localization techniques are quite similar to one another, independent of subject program. While they are slightly different in the case of the `make`, `grep` and `gzip` programs, they are near identical in the case of the Siemens suite (even with regard to outliers). For this reason, we believe it is representative and sufficient to make use of only one fault localization technique (Tarantula), and one set of subject programs (the Siemens suite); for the sake of evaluating the tie-breaking strategies (that are to appear in Section 3). This is with the understanding that any results derived, shall in all likelihood, also apply to the other techniques and subject programs.

### 3. The Tie-Breaking Strategies

Four tie-breaking strategies are considered in our experiments that can be classified as (1) Statement order based (SOS); (2) Confidence based (two strategies are presented, annotated as CS1 and CS2); and (3) Data-dependency based (DDS). We remind the reader that fault localization techniques already sort statements in decreasing order of their suspiciousness, and the tie-breaking strategies are only used to resolve ties. Furthermore, for the purposes of evaluating a tie-breaking strategy we only focus on critical ties (i.e., ties involving a faulty statement). For the purposes of discussion, let us assume  $CT$  is a critical tie consisting of  $k$  statements, and  $\Omega(st)$  – where statement  $st \in CT$  – represents a metric based on which ties among these  $k$  statements can be resolved; such that a statement with a smaller  $\Omega$  value will be examined before a statement with a larger  $\Omega$  value.

Statement order based strategy (SOS): This strategy calls for tied statements to be internally sorted in the order in which they naturally appear in the code, i.e., their statement number.

The order thus, depends on the way in which the program was written. In this scenario,  $\Omega(st)$  = the statement number of  $st$ . This strategy assumes that when programmers debug, the intuitively, follow the order in which the program may naturally appear (say should a source file be opened in an Integrated Development Environment or a plain text editor).

Confidence based strategies (CS1 and CS2): Jones et al. proposed the use of a tie-breaking metric in [13], which they refer to as the ‘confidence’ of a statement. The metric is named so as it is designed to measure the degree of confidence in the given suspiciousness. The metric is given as follows, and the strategy that utilizes it is referred to as CS1.

$$confidence(s) = \max\left(\frac{failed(s)}{totalfailed}, \frac{passed(s)}{totalpassed}\right) \quad (4)$$

Along similar lines, we propose a new strategy (abbreviated as CS2) which is based on the intuition that statement suspiciousness should be more dependent on the information extracted from failed test cases than passed ones. Thus, regarding the suspiciousness computation of a statement – the contribution of the failed tests that execute it with respect to all failed tests; should be greater than the contribution of the passed test cases that execute it, with respect to all the passed test cases. We thus, use a new confidence metric for CS2:

$$confidence(s) = \frac{failed(s)}{totalfailed} - \frac{passed(s)}{totalpassed} \quad (5)$$

Regardless of whether CS1 or CS2 is used to break-ties, we have:  $\Omega(st) = confidence(st)$

Data dependency based strategy (DDS): The use of data dependency analysis has been proposed for the purposes of fault localization before in [10]. Let us define a data dependency relation  $\Delta$  between two statements  $s_i$  and  $s_j$ , such that  $s_i \Delta s_j$  if and only if  $s_i$  defines some variable that is used in  $s_j$ , or  $s_j$  defines some variable used in  $s_i$ . Now for each statement  $st$  in the critical tie  $CT$ , the  $\Omega$  value is defined as the suspiciousness value of the most suspicious statement that is not in the critical tie, but is data dependent on  $st$ .

$$\Omega(st) = \max(susp(s) \mid \forall s.(s \notin CT) \wedge (s \Delta st), s \in \mathcal{P}) \quad (6)$$

Where  $susp(s)$  returns the suspiciousness assigned to statement  $s$  by the fault localization technique in question, and  $\mathcal{P}$  is the set of all the executable statements in the program. The rationale behind the strategy is that even though a fault may not be in a statement  $\alpha$  that has a high suspiciousness, it is very likely that the fault has something to do with a statement that is directly data dependent on statement  $\alpha$ .

### 4. Evaluation of the Tie-Breaking Strategies

As per the discussion in Section 2.4, we evaluate the tie-breaking strategies based on the Tarantula fault localization technique and programs of the Siemens Suite. However, prior to the evaluation, we first present details on how the strategies are to be evaluated.

## 4.1 Evaluation Metric

It is important to recognize that the breaking of ties can affect the effectiveness of a fault localization technique, and for a tie-breaking strategy to be useful, it should not affect the fault localization effectiveness adversely. Thus, this becomes a fundamental requirement of any tie-breaking strategy.

For the purposes of this paper, fault localization effectiveness is measured in terms of the average number of statements that need to be examined to find the first faulty statement (for a discussion on other metrics, please refer to Section 6). Thus, for any given faulty version, the average effectiveness can be computed as the mean of the best and worst effectiveness. Recall that the best effectiveness is achieved when we examine the faulty statement before all other statements tied for the same suspiciousness, and the worst effectiveness is achieved when the faulty statement is examined only after. Note that the worst effectiveness is in fact the sum of the best effectiveness and the size of the critical tie. Along with measuring the average effectiveness after the application of a tie-breaking strategy, we also define a variable called *Tie-Reduction* that quantifies the extent to which critical ties have been broken. For any tie-breaking strategy, the corresponding Tie-Reduction is computed as:

$$Tie-Reduction = \left( 1 - \frac{\text{size of critical tie after applying the strategy}}{\text{size of critical tie without applying the strategy}} \right) \times 100\% \quad (7)$$

A good tie-breaking strategy should strive to at least maintain the same average effectiveness (if not improve it), while resulting in a high Tie-Reduction, i.e., considerably reducing the size of the critical tie.

## 4.2 Results

Table 1 presents data regarding the average effectiveness (in terms of the average number of statements that need to be examined to reach a faulty statement); the number of critically tied statements (i.e., the size of the critical tie), and the Tie-Reduction (as defined in Section 4.1), when each of the strategies are used in conjunction with Tarantula on the Siemens suite. The data is presented from a cumulative perspective, i.e., the data corresponds to all 129 faulty versions of the Siemens suite. For example, based on the first row of the table we observe that when no strategy is used, on average, Tarantula requires the examination of 2882 executable statements (of 16507 executable statements in all) to locate all the faults, out of which 858 statements constitute critical ties. We observe that strategies CS1 and CS2 do not alter the average effectiveness, and are also not able to break any critical ties. This suggests that they may be of limited or no use for the purposes of breaking critical ties, at least when fault localization techniques such as Tarantula are used.

As far as the other strategies are concerned, we note that in terms of Tie-Reduction, the DDS strategy is able to reduce critical ties by almost 10% while maintaining the same effectiveness (in fact it is more effective by 12 statements). This suggests that the DDS strategy may hold potential as a practical tie-breaking strategy. Not surprising is the fact that

the SOS strategy is able to achieve a Tie-Reduction of 100% in that all ties are broken. This is easy to explain as in the case of SOS, statement numbers are used to break critical ties; and as long as each statement number is unique, there shall never be any ties. What is surprising however is the fact that on average SOS results in a better fault localization effectiveness, because a fewer number of statements need to be examined in order to find all the faults, than had the strategy not been applied to begin with. Interesting and appealing as this may be, we refrain from drawing any conclusions about the use of SOS to improve fault localization effectiveness, as our observations may be related to external factors, such as how the faults in the programs of the Siemens suite have been seeded. Investigating the use of SOS as an aid to improve fault localization effectiveness is deferred to future studies. However, a conclusion that can be safely drawn is that the fault localization effectiveness is not affected adversely, yet all ties have been broken, which is an encouraging result as far as the use of SOS as a tie-breaking strategy is concerned.

Table 1. Evaluation of the Tie-breaking strategies using proposed metric

Strategy (Based on Tarantula)	Cumulative Average Effectiveness	Total Number of Critically Tied Statements	Tie-Reduction
None	2882	858	0%
SOS	2706	0	100%
CS1	2882	858	0%
CS2	2882	858	0%
DDS	2870	773	≈10%

## 5. Threats to Validity

There are several identified threats to validity which include but are not limited to the following. A threat to external validity is our choice of fault localization techniques and subject programs which may limit our ability to generalize our results. Three fault localization techniques – Tarantula, Ochiai, and Heuristic III were evaluated across four sets of subject programs – the programs of the Siemens suite and the grep, gzip and make programs (10 programs in total). While we cannot claim that our results apply everywhere, this extensive combination of techniques and programs does allow us to have confidence in the validity of our results. For the purposes of evaluating the tie-breaking strategies, only Tarantula has been considered across the Siemens suite programs. However, this has primarily been due to space limitations, and because based on our results in Section 2.4, Tarantula seems to be a fairly representative technique, and the tie-related problems seem independent of subject program.

A threat to the internal validity is the way in which the faults may have been inserted in order to create the faulty versions that have been used for our experiments. Even though the results in Section 4.2 indicate that the statement order based strategy (SOS) can potentially improve the effectiveness of a fault localization technique, we are unable to attribute a reason as to why. This may have something to do with the faults themselves. A threat to the construct validity is the metric used to evaluate fault localization effectiveness (i.e., the total number of statements that need to

be examined to locate a fault). However, such a metric has also been used in studies such as [9].

Another threat to the validity is that the subject programs used for our experiments have all been single-fault programs (i.e., each faulty version has a single fault in it). However, tie-breaking strategies may easily be applied to programs with multiple faults as well via the procedure described in [4] where *fault-focused* clusters are constructed such that failed tests in each cluster are associated with the same fault. The successful test cases are then combined with the fault-focused clusters to produce specialized test suites, whose information can be fed to a fault localization technique to locate the fault associated with that cluster. Once such fault-focused clusters are constructed (the discussion of which is beyond the scope of this paper), regardless of which fault localization technique is used, statements may still be tied for the same suspiciousness, and therefore, the tie-breaking strategies are still of great value.

## 6. Related work

There are a great number of fault localization techniques (in addition to the ones discussed by us) that have been proposed over the recent past, and while this paper is directly related to fault localization, we cannot hope to cover them all. For readers interested in finding out more about fault localization and the current state of the art fault localization techniques, we refer them to [8]. With the exception of the confidence metric (used in strategy CS1) proposed by Jones et al. in [13], we are unaware of any other research work that has dealt explicitly with the problem of resolving ties (especially critical ties) in the context of fault localization.

As explained in Section 4.1, a good tie-breaking strategy must reduce ties without adversely affecting fault localization effectiveness. For the purposes of this paper, effectiveness has been measured in terms of the total number of executable statements that must be examined to locate all faults under study. Other metrics also exist that evaluate fault localization effectiveness using various means. Renieris et al. [6] suggested the assignment of a score to each faulty version of a subject program, which is defined as the percentage of the program that need not be examined to find a faulty statement in the program or a faulty node in the corresponding program dependence graph (PDG). Cleve and Zeller also adopted the same score (i.e., effectiveness measure) in [2]. In order to make their computations comparable to those based on the PDG, the authors of Tarantula [3] consider only executable statements, but essentially use the same score. Wong et al. [9] define an *EXAM* score which expresses the effectiveness of a fault localization technique in terms of the percentage of code that needs to be examined, as opposed to the percentage of code that need not. A similar modification is made by Liu et al. in [5] except that their score is again based on the PDG.

## 7. Conclusions

Fault localization techniques strive to assign suspiciousness (likelihood of being faulty) values to program components, such that these components can then be examined by

programmers in decreasing order of their suspiciousness during debugging. However, statements might be tied for the same suspiciousness and thus, programmers must use their own intuition or judgment to decide which components to examine first. Furthermore, the more number of ties that involve faulty statements, the harder it is to precisely estimate at what point during the examination, the faulty statement will be encountered. In this paper we systematically analyze the problems associated with ties involving faulty statements and reveal that the problem is quite frequent and is not limited to any particular fault localization technique or subject program. To alleviate the problem, four tie-breaking strategies are considered, and evaluated via case studies. Results indicate that some of the strategies can indeed reduce ties without having an adverse impact on fault localization effectiveness. Furthermore, our observations indicate that the tie-breaking strategies also evidence some potential for improving fault localization effectiveness on average.

## References

1. R. Abreu, P. Zoetewij, and A.J.C. van Gemund, "An evaluation of similarity coefficients for software fault localization," in *Proceedings of the 12th Pacific Rim International Symposium on Dependable Computing*, pp. 39-46, Riverside, California, USA, December 2006
2. H. Cleve and A. Zeller, "Locating causes of program failures," in *Proceedings of the 27th International Conference on Software Engineering*, pp. 342-351, St. Louis, Missouri, USA, May 2005
3. J. A. Jones and M. J. Harrold, "Empirical evaluation of the Tarantula automatic fault-localization technique," in *Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering*, pp. 273-283, California, USA, November 2005
4. J. A. Jones, J. Bowring, and M. J. Harrold, "Debugging in parallel," in *Proceedings of the 2007 International Symposium on Software Testing and Analysis*, pp. 16-26, London, UK, July 2007
5. C. Liu, L. Fei, X. Yan, J. Han, and S. P. Midkiff, "Statistical debugging: a hypothesis testing-based approach," *IEEE Transactions on Software Engineering*, 32(10):831-848, October 2006
6. M. Renieris and S. P. Reiss, "Fault localization with nearest neighbor queries," in *Proceedings of the 18th IEEE International Conference on Automated Software Engineering*, pp. 30-39, Montreal, Canada, October 2003
7. I. Vessey, "Expertise in debugging computer programs," *International Journal of Man-Machine Studies: A Process Analysis*, 23(5):459-494, 1985
8. W. E. Wong and V. Debroy, "A survey on software fault localization," Technical Report UTDCS-45-09, Department of Computer Science, University of Texas at Dallas, November 2009
9. W. E. Wong, V. Debroy, and B. Choi, "A Family of Code Coverage-based Heuristics for Effective Fault Localization," *Journal of Systems and Software*, 83(2):188-208, February 2010
10. W. E. Wong and Y. Qi, "Effective program debugging based on execution slices and inter-block data dependency," *Journal of Systems and Software*, 79(7):891-903, July 2006
11. W. E. Wong, Y. Shi, Y. Qi, and R. Golden, "Using an RBF Neural Network to Locate Program Bugs," in *Proceedings of the 19th IEEE International Symposium on Software Reliability Engineering*, pp. 27-38, Seattle, USA, November 2008
12. W. E. Wong, T. Wei, Y. Qi and L. Zhao, "A Crosstab-based statistical method for effective fault localization, in *Proceedings of the First International Conference on Software Testing, Verification and Validation*, pp. 42-51, Lillehammer, Norway, April 2008
13. Y. Yu, J. A. Jones, and M. J. Harrold, "An empirical study of the effects of test-suite reduction on fault localization", in *Proceedings of the International Conference on Software Engineering*, pp.201-210, Leipzig, Germany, May 2008

# An Ontology Model to Support the Automated Evaluation of Software

Raúl García-Castro\*, Miguel Esteban-Gutiérrez\*, Mick Kerrigan<sup>†</sup> and Stephan Grimm<sup>‡</sup>

\*Ontology Engineering Group. Facultad de Informática, Universidad Politécnica de Madrid, Spain

<sup>†</sup>STI Innsbruck. Universität Innsbruck, Austria

<sup>‡</sup>FZI Research Center for Information Technology, Germany

**Abstract**—Even though previous research has tried to model Software Engineering knowledge, focusing either on the entire discipline or on parts of it, we lack an integrated conceptual model for representing software evaluations, and we also lack the information related to them that supports their definition and enables their automation and reproducibility.

This paper presents an extensible ontology model for representing software evaluations and evaluation campaigns, i.e., worldwide activities where a group of tools is evaluated according to a certain evaluation specification using common test data.

During the development of the ontologies, we have reused current standards and models and have linked these ontologies with some renowned ones.

## I. INTRODUCTION

The SEALS European project<sup>1</sup> is developing an infrastructure for the evaluation of semantic technologies, the *SEALS Platform*, that will provide independent computational and data resources for evaluating such technologies. This platform will allow users to define and execute evaluations on their own and will support the organization and execution of evaluation campaigns, i.e., worldwide activities where a group of tools is evaluated according to a certain evaluation specification with common test data.

One core component in the development of any infrastructure supporting software evaluations is the definition of the data model to be used for representing software evaluations, evaluation campaigns and the rest of the entities managed by such a platform.

Nevertheless, even if there is some progress in reaching consensus in the Software Engineering discipline and in the content of its knowledge areas – the best example is the initiative that led to the Guide to the Software Engineering Body of Knowledge (SWEBOK) [1] – we still lack formal and reusable models for representing information common in the software evaluation area.

Previous research has tried to model Software Engineering knowledge related to software evaluation, both in general (e.g., [2]) or focusing on parts of the discipline (e.g., software quality characteristics as covered in [3], [4], [5] and software measurement in [6], [7]), but we lack an integrated model for representing software evaluations, the different information related to them, and software evaluation campaigns.

The main contribution of this paper is an explicit conceptual model for representing software evaluations and evaluation campaigns. Our design principles when defining this model were that it is

- machine-processable to support the automation of the evaluation process,
- exhaustive to allow evaluations to be reproducible,
- interoperable to allow interchanging evaluation-related information between different systems, and
- extensible because it will have to be expanded to be used in concrete evaluations and evaluation campaigns.

To cover these requirements we decided to use ontologies for representing such a model. Ontologies are formal and explicit specifications of a conceptualization [8] that allow representing consensual knowledge, are easily extensible, and support interoperability at the knowledge level.

Furthermore, we encourage the use of the ontologies presented in this paper to represent software evaluation information. To facilitate this, we have reused current standards and models and linked our ontologies to some renowned ones.

This paper is structured as follows. Sections II and III present the main entities related to a software evaluation activity and the life cycle of these entities, respectively. Section IV introduces the upper ontology used to represent common entities and their properties and Sections V, VI and VII discuss the ontologies used to represent software evaluations, evaluation execution requests and evaluation campaigns, respectively. Section VIII refers to previous work related to the one presented in this paper and, finally, Section IX draws conclusions from this work and proposes future lines of research.

## II. SOFTWARE EVALUATION ENTITIES

Our model revolves around the notion of *evaluation*, which is largely inspired by the notion of evaluation module as defined by the ISO/IEC 14598 standard on software product evaluation [9]. However, it is not our intention to fully cover this standard but to focus on the entities required to describe software evaluations for their automated execution.

As illustrated in Figure 1, in any *evaluation* a given set of *tools* are exercised, following the workflow defined by a given *evaluation description* and using determined *test data*. As an outcome of this process, a set of *evaluation results* is produced.

<sup>1</sup><http://www.seals-project.eu/>

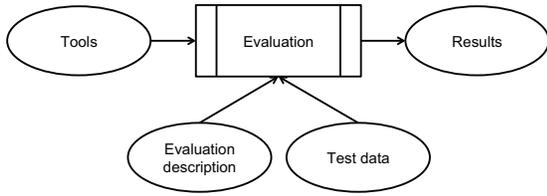


Fig. 1. Main entities in a software evaluation scenario

This high-level classification of entities can be further refined as needed. For example, in the context of SEALS, tools are classified into different types of semantic technologies according to their functional scope, namely, *ontology engineering tools*, *ontology storage systems*, *matching tools*, etc.

Similarly, it is also possible to distinguish different types of test data: *persistent test data* (those whose contents are stored in and physically managed by the evaluation platform), *external test data* (those whose contents reside outside the evaluation platform and whose life cycle is not controlled by it), and *synthetic test data generators* (pieces of software that can generate synthetic test data on-demand according to some determined configuration).

In accordance with the approach followed in the IEEE 1061 standard for a software quality metrics methodology [10], evaluation results are classified according to their provenance, differentiating *raw results* (those evaluation results directly generated by tools) from *interpreted results* (those generated from other evaluation results).

Besides, our entities include not only the results obtained in the evaluation but also any contextual information related to such evaluation, a need also acknowledged by other authors [6]. To this end, we also represent the information required for automating the execution of an evaluation description in the platform that, with the other entities presented, allows obtaining traceable and reproducible evaluation results.

Finally, another type of entities are *evaluation campaigns*, which represent the information needed to support the organization and running of campaigns for the evaluation of different (types of) participating tools. An evaluation campaign contains one or more *evaluation scenarios*, which include the evaluation description and test data to be used for carrying out the evaluation and the tools that will be evaluated.

Each of the abovementioned entities is composed of two different elements: the *data* that define the entity itself and the *description* of the entity, that is, the set of metadata that characterizes the entity (both generally and specifically) and enables the provision of the discovery mechanisms required for entity integration, consumption, and administration by the evaluation platform.

### III. ENTITIES LIFE CYCLE

Different entities have different life cycles in the evaluation platform. This section describes the life cycles of the most relevant entities.

Tools, test data, and evaluation descriptions are defined in the platform as *artifacts*, which are a collection of *artifact*

*versions*; for example, a particular tool can have a number of different tool versions that evolve over time.

Figure 2 shows state diagrams for artifacts and artifact versions, including the possible states, the operations that alter the state, and the operations that retrieve the entity information (data and metadata) in dotted arrows. It can be observed that, once registered in the platform, artifacts can always be retrieved and have a single state until they are deregistered. On the other hand, artifact versions have two states, published and unpublished; in the former state artifact versions can only be retrieved, and in the latter state they can only be updated. In this way, evaluations can only be performed using fixed (i.e., published) artifact versions.

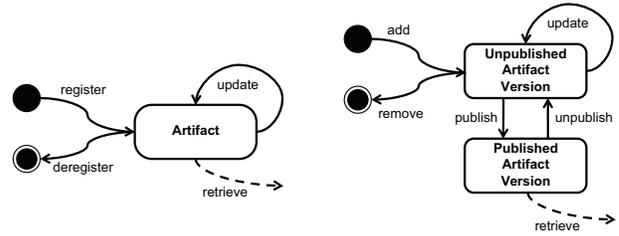


Fig. 2. Life cycle of artifacts (left) and artifact versions (right).

Evaluation results (raw results and interpretations) are defined as artifacts with no version information. Additionally, once registered they cannot be updated.

Evaluation descriptions are processed by the evaluation platform through *execution requests*. An execution request encapsulates the execution needs that a particular user has at some point in time, i.e., which evaluation description is to be executed, which tools shall be evaluated, which test data shall be used for its evaluation, etc.

During its life cycle, an execution request transits among eight different states, as shown in Figure 3. The starting state of an execution request is that of “*pending*”, which takes place whenever a new execution request is created.

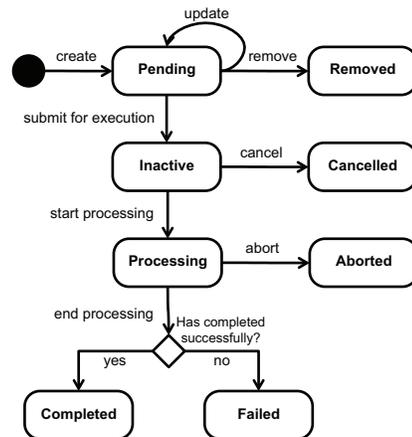


Fig. 3. Life cycle of an execution request.

At this point, the execution request can be updated, removed, or submitted for execution. Whereas the first operation

does not change the state of the execution request, the other two do change it: on the one hand, when the execution request is removed, the state transits to the “*removed*” state, a state in which no further operations are possible<sup>2</sup>; on the other hand, when the execution request is submitted to execution, the state transits to the “*inactive*” state. Beyond this point, the execution request shall not be further modified.

While the execution request is inactive, two possible courses of action can take place: it can be cancelled or it can start being processed. In the first case, the state transits to the “*cancelled*” state, a state in which, again, no further operations are possible. The latter case takes place once the execution requirements of the execution request are fully satisfied and, then, the state transits to the “*processing*” state.

Once the execution request is being processed, three possible outcomes may occur: (1) The execution request may be completed successfully, and thus the state transits to the “*completed*” state. (2) Some failure might prevent completing the execution of the execution request, causing the state to transit to the “*failed*” state. (3) It is also possible that the processing of the evaluation request is aborted (e.g., due to an abnormal duration time), thus forcing the state to transit to “*aborted*”. Regardless of the course of action, no further operations over the execution request will be carried out.

As can be seen, execution requests are not disposed by the evaluation platform. On the contrary, regardless of the execution request’s internal state, its information is available to the user at any time, providing a complete and historical view of the evaluation activities over time.

#### IV. UPPER ONTOLOGY

The entities presented above share a number of common properties. In this section we explore these properties and design the upper ontology that will be used to represent them. We have reused the Dublin Core vocabulary [11] because it already defines a consensual set of properties for describing resources.

Every entity managed by the evaluation platform can be described in terms of a top-level entity, which can be further specialized. As mentioned in section III, tools, test data, evaluation descriptions, and results are described in terms of an artifact, which is a collection of artifact versions. In the case of results, only an artifact is used to describe them since they do not include version information.

Given this information, we define three classes within the upper ontology, namely, *Entity*, *Artifact*, and *ArtifactVersion*, where the last two are specializations of the first one, as can be seen in Figure 4.

Entity descriptions include their creator, creation time, name, identifier, and description. Regarding artifacts, we also store the person responsible for the artifact, an URL with further information about it, and the collection of artifact versions (identifying the current version). Each artifact version

<sup>2</sup>That is, the state of the execution request will not be changed beyond this point.

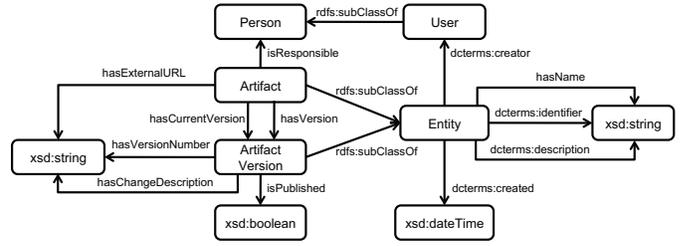


Fig. 4. Overview of the upper ontology.

is described by a version number, a description of the changes related to the version, and a flag to indicate whether it is published or not (as explained in Section III).

We have to note that we differentiate between a person and a platform user as a specific type of person, since only the latter can create entities in the platform. For describing them, we extend the FOAF [12] and VCard [13] vocabularies.

#### V. MODELING SOFTWARE EVALUATIONS

Figure 5 provides an overview of the main classes and properties defined for modeling software evaluations. Although they are not shown in the figure, between each type of artifact and its version we have defined subproperties of *hasVersion* (e.g., “*Tool hasToolVersion ToolVersion.*”). Next, we describe the main entities represented in the figure.

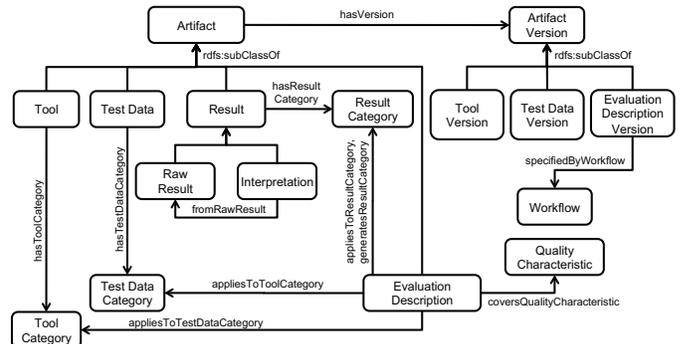


Fig. 5. Main classes and properties for modeling software evaluations.

Tools are defined by classifying them into a certain category, and the definition of each tool version specifies its particular capabilities, the hardware platform and operating system upon which it can be executed, and its execution requirements.

As described in Section II, we cover three types of test data, namely, persistent test data, external test data, and synthetically generated test data. In order to allow the evaluations to be reproducible, only persistent test data will be used in them. For external test data, further details need to be specified, such as the location where they can be accessed and when they are persisted into the platform as a test data version, or the date when they were retrieved. Similarly, when persistent test data is synthetically created by running a test data generator, the test data generator used for creation must be stored, as well as the configuration used to execute the generator.

Besides specifying test data using this hierarchy, these test data must be categorised to facilitate their future retrieval (e.g., interoperability test data, scalability test data, etc.) and each test data version must define the tool capabilities that the test data version can exercise. Also, we group evaluation test data in test suites for a meaningful analysis of the produced evaluation results, as suggested by [6].

We also mentioned in Section II that results specialize into raw results and interpretations. Regardless of the type of result, we need to categorise any result generated and to specify which raw result was used to create a certain interpretation.

The definition of an evaluation description includes the categories of tools and test data that can be used with the evaluation description as well as the category of results that the evaluation description produces when it is executed. An evaluation description also specifies which software quality characteristics can be measured with it. We do not impose any software quality model to define these quality characteristics. Nevertheless, we suggest to use the quality characteristics defined in the quality model of the ISO/IEC 9126 standard on software product quality [14].

An evaluation description also includes an execution contract that defines what the evaluation description needs to be executed and the type of results that will be produced. The workflow with the executable specification of an evaluation description is included in the evaluation description versions.

## VI. MODELING EXECUTION REQUESTS

An evaluation execution request is associated to a certain evaluation description version. Thus, an execution request is totally coupled with the contract specified by the evaluation description to which the version belongs (see Figure 6). In this way, the evaluation platform is able to verify whether the evaluation description version can be enacted.

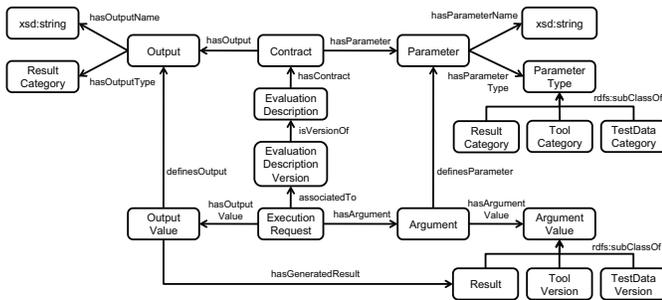


Fig. 6. Main classes and properties for modeling execution requests.

An execution contract defines what an evaluation description needs in order to be executed and the output that it will produce with the specified input. The inputs are specified by their parameter names and by the types of artifacts that can be used within these parameters (i.e., tool, test data and result categories) and the outputs are specified by the output names and the types of artifacts that can be used within these outputs (which are always result categories).

To execute a contract, the execution request must specify the arguments that define each parameter of the contract and the particular values of each argument. Upon execution, the platform will indicate the results that were generated by specifying the output values that define each output of the contract and the particular result generated.

Besides, the platform will also manage information about the life cycle of execution requests (the states presented in Section III and the moments of time when they change) and the computing resources that were used for the execution of evaluation descriptions.

In this way, an insight is provided about the configuration of the evaluation platform that was used for carrying out the evaluation with the objective of enabling the reproducibility of the evaluations afterwards, since the platform configuration may have a direct effect on the results obtained (e.g., in efficiency or scalability evaluations).

## VII. MODELING EVALUATION CAMPAIGNS

As mentioned above, we also want to model information about evaluation campaigns. As shown in Figure 7, these campaigns are activities organized by some users and contain a set of evaluation scenarios that will be executed over some participating tools. For each evaluation scenario, it is necessary to identify the evaluation description version and test data version used, as well as who will be participating (both the user participating and the tool version). Finally, it is necessary to identify the execution requests by which the various evaluation scenarios of a campaign are executed.

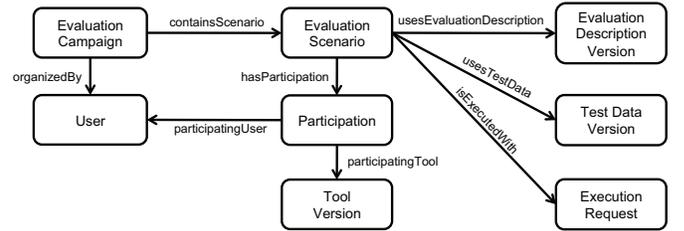


Fig. 7. Main classes and properties for modeling evaluation campaigns.

## VIII. RELATED WORK

By the time of writing this paper, we have not found any ontology or other conceptual model that allows representing all the entities involved in a software evaluation activity in an integrated way. What we have found are some models that cover parts of this activity or are related to it that are worth mentioning.

Next, we present an overview of the models that can be used to represent software quality characteristics, software measurement concepts, measurement and evaluation results, and software projects; we also discuss the main similarities and differences between these models and the one presented in this paper.

### A. Software Quality Characteristic Ontologies

Different researchers have dealt with the development of ontologies for representing software quality characteristics. For example, [4] includes an analysis over different software quality documentation aspects (standards, publications, etc.) with the goal of identifying common software quality attributes and extracting the relevant concepts and relationships along with their frequency of use; also, [3] and [5] describe ontologies to represent quality characteristics in domain engineering and in the software product audit areas, respectively. Unfortunately, no implementation of any of these ontologies is available to be reused.

In our case, as Section V mentions, we allow defining the quality characteristics that an evaluation description covers although we do not impose any quality model.

### B. Software Measurement Ontology

The Software Measurement Ontology (SMO) [7] has been defined by analysing software measurement standards and research proposals and is composed of a set of OWL ontologies for representing software measurement concepts and properties.

The idea behind SMO is similar to the one in our work. In SMO, a measurement result is produced by the measurement performed over an entity following some measurement approach; in our case, an evaluation result is produced by the execution request performed over some tool following an evaluation description.

However, the different scope of both works (SMO tries to cover the whole terminology involving software measurement, whereas we focus on the automated evaluation of software products) shows some differences in the resulting models.

Similarly as in our model, SMO differentiates between the results produced with base measures (raw results) and those produced with derived measures or indicators (interpretations). Nevertheless, in our case we do not differentiate between derived measures and indicators because we do not have a specific interest in the type of measurement approach used to obtain the interpretations.

Moreover, in our model we do not deal with details regarding the concrete tool attributes evaluated, measurement scales, and units of measurement; we leave this information open to be modeled in concrete evaluations as needed.

On the other hand, we define in detail the measurement method (evaluation description) to be used by specifying an executable evaluation workflow, inputs (test data), and a contract.

In the future we will analyse whether to extend our model with some of the general concepts of software measurement included in SMO in order to increase its applicability.

### C. The MiniSQUID Model

The MiniSQUID model [6] allows storing the measurements obtained from software measurement programs in companies.

The main differences between this entity-relationship model and our way of representing results are that the MiniSQUID

model covers the measurement of any type of software entity in the software development process whereas our model tackles only software products.

Moreover, measurement units and scales are a cornerstone of the MiniSQUID model, whereas we currently leave this aspect open, so that evaluation designers can freely decide the units and scales to be used.

### D. Evaluation and Report Language

The Evaluation and Report Language (EARL) [15] defines a vocabulary for expressing test results in RDF(S) [16]. By the time of writing this paper, it is a working draft in the W3C.

The approach for representing results in EARL is similar to our approach. EARL states that a result is obtained when an assessor (i.e., who runs the test) tests some subject (i.e., what is being tested) according to a certain test criterion (i.e., a requirement or test case), and in our case we state that an evaluation result is obtained when some tool is evaluated according to certain test data.

Even if the primary goal of EARL was the exchange of test results between Web accessibility evaluation tools, it is designed to be flexible enough to cover other types of test results. This flexibility makes the scope of EARL broader than ours since it takes into account multiple types of assessors and subjects, whereas in our case the only assessor is the evaluation platform and the only type of subject is software.

However, the main difference between EARL and the metadata presented in this paper is their focus; while EARL is mainly centered on testing, our work is mainly focused on evaluation. In EARL test results are expected to inform about whether a subject passes a test or not and to provide pointers to the parts of the subjects relevant to the result, whereas in our case evaluation results describe a subject according to a set of metrics and disregard which parts of the subject are relevant to the results.

While EARL (as well as the previously-mentioned MiniSQUID model) only covers the representation of results, we cover the representation of all the resources involved in the evaluation life cycle (from the description of the evaluation and test data to the evaluation execution) so that the evaluation can be reproduced at a later stage. Another main difference is that we expect the entities involved in the evaluation (tools, evaluation descriptions, and test data) to change over time; therefore, we model these artifacts and their sets of versions in a different way.

However, once EARL becomes a W3C recommendation, we will study the feasibility of defining an alignment between our vocabulary and the one proposed in EARL in order to be able to export the results produced by the SEALS Platform in terms of the EARL vocabulary.

### E. Description of a Project

The Description of a Project (DOAP) vocabulary<sup>3</sup> can be used to describe software projects (in particular open

<sup>3</sup><http://trac.usefulinc.com/doap>

source ones), different versions of a project, and any related specifications and repositories. Similarly to our case, DOAP also differentiates between a software project and the different versions (releases) of it.

We do not reuse this vocabulary for describing tools because the DOAP terms that are relevant to our case are already included in our current descriptions of the Tool and ToolVersion classes (and their super-classes). Because of this, the alignment between our proposal and DOAP is straightforward in case interchanges between the two vocabularies are required.

## IX. CONCLUSIONS AND FUTURE WORK

This paper presents a first version of an ontology model that can be used to represent any information required for evaluating software or for organising an evaluation campaign over software products.

Our work has been guided by the need to obtain an interoperable and extensible model that supports automated and reproducible software evaluations. Even if no standard or model fully covers these needs (mainly because standards cannot cover every specific application need), we have been inspired by existing efforts and have tried to reuse accepted and well-known standards and proposals as much as possible during the development of the ontologies.

The ontologies presented here are generic by design, and concrete evaluations will have to extend them according to their needs. While these extensions will support each specific evaluation, the generic ontologies will allow aggregating multiple evaluations and evaluation results for further processing.

This ontology model has been implemented in terms of lightweight OWL [17] ontologies, which are available in the Web<sup>4</sup>. We decided to use lightweight ontologies because we agree with the authors of [18] when they advice to use ontologies as light as possible to cover project requirements.

We also plan to publish all the data about evaluations and evaluation campaigns as RDF data in the SEALS Platform and we encourage other practitioners to reuse the ontologies presented in this paper in their own evaluations to allow an easier integration of software evaluations and their results.

Even if these ontologies have been obtained through an extensive analysis of the literature and of our current requirements, we expect that new needs are identified and, therefore, these ontologies will change in the future. For example, currently, the ontologies are mainly focused on automated software evaluations, but in the future, they will be extended to allow the insertion of results produced by manual evaluations.

Furthermore, if we want to consistently compare results across different evaluations, we need to model information regarding the scales and units of evaluation results. This information is already covered in other models and right now we leave it open so users can model it according to their specific evaluations. Nevertheless, when needed, this information will be specified in detail in our model.

Finally, as mentioned in the previous section, since software evaluation is highly related to other areas (e.g., software

measurement, software testing, etc.) we will analyse whether our ontologies could be applied to these areas and which extensions are required to achieve this goal.

## ACKNOWLEDGMENT

This work is supported by the SEALS European project (FP7-238975). Thanks to Rosario Plaza for reviewing the grammar of this paper.

## REFERENCES

- [1] A. Abran, J. Moore, P. Bourque, and R. Dupuis, Eds., *SWEBOK: Guide to the Software Engineering Body of Knowledge*. IEEE Press, 2004.
- [2] A. Abran, J. J. Cuadrado, E. García-Barriocanal, O. Mendes, S. Sánchez-Alonso, and M. A. Sicilia, "Engineering the ontology for the SWEBOK: Issues and techniques," in *Ontologies for Software Engineering and Software Technology*, C. Calero, F. Ruiz, and M. Piattini, Eds. Springer, 2006, ch. 3, pp. 103–121.
- [3] R. A. Falbo, G. Guizzardi, and K. C. Duarte, "An ontological approach to domain engineering," in *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering (SEKE 2002)*. New York, NY, USA: ACM, 2002, pp. 351–358.
- [4] A. Kayed, N. Hirzalla, A. A. Samhan, and M. Alfayoumi, "Towards an ontology for software product quality attributes," in *Proceedings of the 2009 Fourth International Conference on Internet and Web Applications and Services (ICIW 2009)*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 200–204.
- [5] R. C. de Boer and H. van Vliet, "QuOnt: an ontology for the reuse of quality criteria," in *Proceedings of the 2009 ICSE Workshop on Sharing and Reusing Architectural Knowledge (SHARK 2009)*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 57–64.
- [6] B. A. Kitchenham, R. T. Hughes, and S. G. Linkman, "Modeling software measurement data," *IEEE Trans. Softw. Eng.*, vol. 27, no. 9, pp. 788–804, 2001.
- [7] M. F. Bertoa, A. Vallecillo, and F. García, "An ontology for software measurement," in *Ontologies for Software Engineering and Software Technology*, C. Calero, F. Ruiz, and M. Piattini, Eds., 2006, ch. 6, pp. 175–196.
- [8] R. Studer, V. R. Benjamins, and D. Fensel, "Knowledge engineering: Principles and methods," *IEEE Transactions on Data and Knowledge Engineering*, vol. 25, no. 1-2, pp. 161–197, 1998.
- [9] *ISO/IEC 14598-6: Software product evaluation - Part 6: Documentation of evaluation modules*. ISO/IEC, 2001.
- [10] *IEEE 1061-1998. IEEE Standard for a Software Quality Metrics Methodology*. IEEE, December 1998.
- [11] DCMI Usage Board, "DCMI Metadata Terms," Dublin Core Metadata Initiative, Recommendation, 14 January 2008. [Online]. Available: <http://dublincore.org/documents/dcmi-terms/>
- [12] D. Brickley and L. Miller, "FOAF vocabulary specification 0.97," FOAF Project, Namespace Document - 3D Edition, 1 January 2010. [Online]. Available: <http://xmlns.com/foaf/spec/>
- [13] H. Halpin, R. Ianella, B. Suda, and N. Walsh, "Representing vCard objects in RDF," W3C, Member Submission, 20 January 2010. [Online]. Available: <http://www.w3.org/TR/vcard-rdf/>
- [14] *ISO/IEC 9126-1. Software Engineering - Product Quality - Part 1: Quality model*. ISO/IEC, 2001.
- [15] S. Abou-Zahra and M. Squillace, "Evaluation and Report Language (EARL) 1.0 Schema," W3C, Last Call WD, Oct. 2009. [Online]. Available: <http://www.w3.org/TR/2009/WD-EARL10-Schema-20091029/>
- [16] Brickley, D., Guha, R.V. (editors), "RDF Vocabulary Description Language 1.0: RDF Schema," W3C, Recommendation, 10 February 2004. [Online]. Available: <http://www.w3.org/TR/rdf-schema/>
- [17] D. McGuinness and F. van Harmelen, "OWL Web Ontology Language Overview," W3C, Recommendation, 10 February 2004. [Online]. Available: <http://www.w3.org/TR/owl-features/>
- [18] F. Ruiz and J. R. Hilerá, "Using ontologies in software engineering and technology," in *Ontologies for Software Engineering and Software Technology*, C. Calero, F. Ruiz, and M. Piattini, Eds. Springer, 2006, ch. 2, pp. 49–102.

<sup>4</sup><http://www.seals-project.eu/ontologies/>

# A Review of Parametric Effort Estimation Models for the Software Project Planning Process

Pablo Rodríguez-Soria<sup>a</sup>, J.J. Cuadrado-Gallego<sup>a,b</sup>, J.A. Gutiérrez de Mesa<sup>a</sup>, Borja Martín-Herrera<sup>a</sup>

<sup>a</sup> Universidad de Alcalá, Departamento de Ciencias de la Computación, 28805 Alcalá de Henares, Madrid, Spain

<sup>b</sup> Ecole de Technologie Supérieure (ETS) – Université du Québec à Montreal, Canada

{pablo.rsoria, jjcg, borja.martinh}@uah.es

## Abstract

*The aim of this research paper is to present a review of thee of the main software effort estimation methods, focused on Parametric models, that have been developed and then commercialized across the software engineering history. These models, among others, have been considered as the basis of the recent software project effort estimation and today conforms the nucleus of some of the most important companies of software effort estimation.*

*For each model is shown its main features, publications and equations that allow us to see as a whole the operation and implementation of each of these effort estimation methods.*

## 1. Introduction to Software Cost Estimation

The more software becomes important in almost every human activity, the more it becomes complex and difficult to implement. Even if modern software technologies render easier the development of certain types of software products, increased user demands and new application domains produce additional problems. It is not surprising that software project management activities are becoming increasingly important.

One of the most critical activities during the software life cycle is that of estimating the effort and time involved in the development of the software product under consideration. This task is known as Software Cost Estimation (see *Figure 1*).

Estimations may be performed before, during and after the development of software. The cost and time estimates are necessary during the first phases of the software life cycle, in order to decide whether to proceed or not (feasibility study). Accurate estimates are obtained with great difficulty since, at this point, available data may not be precise, wrong assumptions may be made, etc. During the development process, the cost and time estimates are

useful for the initial rough validation and the monitoring of the project's progress. After completion, these estimates may be useful for project productivity assessment.

Estimation methods fall in three main categories, namely expert judgment, machine learning and algorithmic cost estimation. Expert judgment [13] relies purely on the experience of one or more experts. Machine Learning estimation [19, 4], compares the software project under consideration with few (e.g. two or three) similar historical projects (i.e. projects with known characteristics, effort and schedule) using different automated or iterative rules. Algorithmic cost estimation involves the application of a cost model, i.e. one or more mathematical formulas which, typically, have been derived through statistical data analysis.

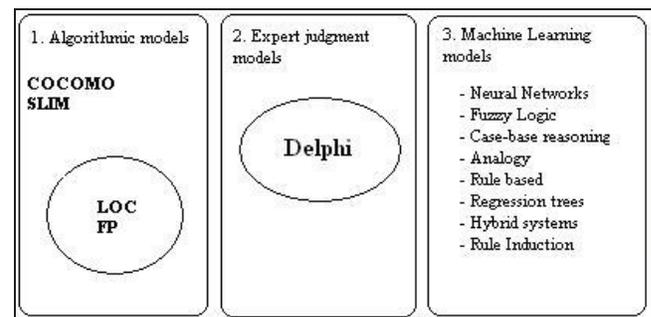


Figure 1. Software Cost Estimation Models

All of the three approaches have known advantages and disadvantages. Expert judgment is easy to apply and produces fast evaluation but suffers from the difficulty to find real experts and is exposed to wrong subjective assessment. Machine Learning models concentrate on a concrete, well-defined estimation framework provided that, suitable projects of the past may be easily found and the mechanism applied is correct. Parametric models are very useful when they are used correctly after they have been calibrated with historical data reflecting the characteristics of the estimated project.

It is important to remark that no single technique is best for all situations, and that a careful comparison of the results of several approaches is most likely to produce realistic estimates. In this survey, we are going to focus onto Algorithmic Models, also known as Parametric Models.

The rest of this paper is structured as follow: Section 2 introduces the reader to Parametric Models and its features. Section 3 presents three Parametric Commercial Models that we have chosen due to their historical relevance. Finally, Section 4 outlines the Conclusions and the main objectives raised in this research.

## 2. Cost Estimation focused on Parametric Models

Since the 1970's, a considerable amount of the software cost estimation research has been focused on the development of new and improved cost estimation models. New models have been proposed and existing models have been compared and validated.

Parametric Models were the most used techniques in the beginning of Software Cost Estimation. These models correspond to the box 1 in *Figure 1*. Then we are going to describe some issues about how these models work in Section 2.1; and how these models have been commercialized in section 2.2.

### 2.1. Parametric Models

The parametric, or statistical, method uses regression analysis of a database of two or more similar systems to develop cost estimating relationships (CERs) which estimate cost based on one or more system performance or design characteristics (e.g., speed, range, weight, thrust). The parametric method is most commonly performed in the initial phases of product description and development. Although during this phase an acquisition program is unable to provide detailed information (like drawings and standards), the program can specify top-level system requirements and design characteristics. In other words, estimating by parametric is a method to show how parameters influence cost.

Parametric estimating is used widely in government and industry because it can yield a multitude of quantifiable measures of merit and quality (i.e., probability of success, levels of risk, etc.). Additionally, CERs developed using the parametric method can easily be used to evaluate the cost effects of changes in design, performance, and program characteristics.

The fast changing nature of software development has made it very difficult to develop parametric models that yield high accuracy for software development in all domains. Software development costs continue to increase and practitioners continually express their concerns over

their inability to accurately predict the costs involved. Accurate software cost estimates are critical to both developers and customers. They can be used for generating request for proposals, contract negotiations, scheduling, monitoring and control. Underestimating the costs may result in management approving proposed systems that then exceed their budgets, with underdeveloped functions and poor quality, and failure to complete on time. Overestimating may result in too many resources committed to the project, or, during contract bidding, result in not winning the contract, which can lead to loss of jobs.

Accurate cost estimation is important because:

- It can help to classify and prioritize development projects with respect to an overall business plan.
- It can be used to determine what resources to commit to the project and how well these resources will be used.
- It can be used to assess the impact of changes and support re-planning.
- Projects can be easier to manage and control when resources are better matched to real needs.
- Customers expect actual development costs to be in line with estimated costs.

Software cost estimation involves the determination of one or more of the following estimates:

- effort (usually in man-months)
- project duration (in calendar time)
- cost (in dollars)

Most cost estimation models attempt to generate an effort estimate, which can then be converted into the project duration and cost. Although effort and cost are closely related, they are not necessarily related by a simple transformation function. Effort is often measured in man/months (MM) of the programmers, analysts and project managers. This effort estimate can be converted into a dollar cost figure by calculating an average salary per unit time of the staff involved, and then multiplying this by the estimated effort required.

Most cost models are based on the size measure, such as Lines of Code (LOC) [2] and Function Points (FP) [1], obtained from size estimation. The accuracy of size estimation directly impacts the accuracy of cost estimation.

### 2.2. Commercial Tools

Since the mid 1990's there have been about 50 commercial software cost estimation tools marketed in the United States and another 25 in Europe, although not all at the same time. Many of these tools are "black boxes" and their methods of operation are proprietary and regarded as trade secrets by their owners [12]. However, while these estimating tools were developed by different companies and

are not identical, they do tend to provide a nucleus of common functions and public equations.

The software cost estimation market was created by researchers who were employed by large enterprises that built large and complex software systems: IBM, RCA, TRW, and the U.S. Air Force were the organizations whose research which led to the development of commercial cost estimating tools.

Commercially available cost estimation tools try to offer the user greater utility by packaging the parametric model with a user interface, database of completed projects, some way of estimating the size of the project, and/or context-sensitive help.

Whatever features any tool may have, most parametric models are likely to employ one or more of three methodologies; Putnam methodology [15] is based on the insight that efficiently run software projects follow well-defined patterns that can be modeled with a set of exponential equations. COCOMO II [5] is a continuation of work begun by Dr. Barry Boehm at USC. Monte Carlo simulation models complex interactions in the face of uncertain estimating assumptions.

As of 2009, some of these estimating tools include COCOMO II, CoStar, CostModeler, CostXpert, KnowledgePlan, PRICE S, SEER, SLIM, and SoftCost. Some older automated cost estimating are no longer being actively marketed but are still in use, such as CheckPoint, COCOMO, ESTIMACS, REVIC, and SPQR/20. Since these tools are not supported by vendors, usage is in decline. The major features of commercial software estimation tools include these attributes:

- Sizing logic for specifications, source code, and test cases
- Phase-level, activity level, and task-level estimation
- Adjustments for specific work periods, holidays, vacations, and overtime
- Adjustments for local salaries and burden rates
- Adjustments for various software projects such as military, systems, commercial, etc.
- Support for function point metrics, lines of code metrics, or both.
- Support for “backfiring” or conversion between lines of code and function points
- Support for both new projects and maintenance and enhancement projects

Some estimating tools also include more advanced functions such as:

- Quality and reliability estimation
- Risk and value analysis
- Return on investment (ROI)
- Sharing of data with project management tools
- Measurement modes for collecting historical data

- Cost and time to complete estimates mixing historical data with projected data
- Support for software process assessments
- Statistical analysis of multiple projects and portfolio analysis
- Currency conversion for dealing with overseas projects

### 3. Review of 3 Parametric Commercial Models

In this section, three of the most relevant models in software cost estimation history will be treated. We have reviewed these tools trying to show their main features, publication dates and central equations. The models in review are: SLIM, SEER-SEM and SPR-Knowledge Plan.

#### 3.1. SLIM – Putnam – 1979

**SLIM;** Software Lifecycle Management.

**First Publication:** Putnam, 1978 [15]

**Patent:** Quantitative Software Management (QSM).

**Tools:**

1. SLIM-Estimate. It is a project planning tool.
2. SLIM Control. It is a project tracking and control tool.
3. SLIM Metrics. It is a software benchmarking tool.

Larry Putnam and Ann Fitzsimmons founded Quantitative Software Management (QSM) and build the first version of SLIM in 1979. It became the second commercial software cost estimation tool on the market.

This model is based on the software lifecycle analysis of Putnam in terms of the size distribution of the development team of a software product against the time that follows a distribution of Rayleigh and it is based on the work of Norden [14] and Aron [3]. Norden observed through the graphical representation of the personnel distribution frequencies during the development and maintenance phases of many projects implemented in IBM, that the curves resembled quite to the distribution curves of Rayleigh since 90% of the project was completed in two-thirds of the total time, while the remaining 10% needed a third of the total time remaining to be completed. Although this distribution was purely empirical, Norden found no theoretical basis for it.

SLIM supports the widespread methods of size estimating, including the source lines of code and function points. It can predict the size of the project, the effort, the development time and the proportion of defects.

**Equations:** The equations of the model have not been edited for the public domain, although the central algorithms of the model were published by Putnam [16]. These are the ones collected here:

1. Size:  $e = c \cdot (E_d)^{1/3} \cdot (t_d)^{4/3}$

Where  $e$  is the size in SLOC,  $E_d$  is the total effort needed to complete the project, selected from a database of previous projects,  $c$  is a constant of the project called by Putnam *Technological Factor*, this factor reflects the effect of numerous cost drivers as the constraints of hardware, the complexity of the program, the personnel levels of experience and the programming environment.  $t_d$  is the total development time of the project.

2. Effort:  $E(t) = E_d (1 - e^{-at^2})$ ;  $a = \frac{1}{2(t_d)^2}$

Where  $E(t)$  is the effort that has been consumed in MM in order to develop the project during  $t$  months,  $a$  is a constant of the project that determines the curve, it is also obtained from previous projects.

### 3.2. SEER-SEM – 1989

**SEER-SEM**; System Evaluation and Estimation Resources – Software Estimation Model.

**First Publication:** Jensen, 1983 [10]

**Patent:** Galorath Associates Inc.

**Tools:**

- SEER-SEM.

It is based on the Model of Jensen of 1979 [9], which is based on the Model of Putnam of 1977 [15].

The scope of the model is broad and covers all phases of the lifecycle, from the first specifications, until design, development, delivery and maintenance. It manages a wide variety of development environment configurations and types of applications, such as client – server, distributed, graphics, etc. It manages the development methods and languages more used. The development methods include objects oriented, reuse, development in spiral, cascade, of prototypes and incremental. Languages include both the 3rd and 4th generation (C++, FORTRAN, COBOL, Ada, etc.) as well as application generators. It allows taking as constraints the capacity of the development team, the design standards and process required, and the levels of an acceptable development risk. Among the characteristics of the model, the following are included:

- It allows that the level of the estimation probabilities, the development team and the development time are inputs as independent variables.
- It allows an extended sensitive analysis and a monitoring of the input parameters of the model.
- It shows the project cost drivers.
- It allows and interactive adjustment of the schedule of the project elements through Diagrams of Gantt.
- It builds the estimates by a knowledge base of existing projects.

The model specifications include:

1. *Parameter:* Size, personnel, complexity, development environment, method of development and acquisition, applicable standards.
2. *Predictions:* Effort, development time, development team, defects, costs. The estimates may be based in development time or effort. The constraints can be specified at the development time or development team.
3. *Risk Analysis*
4. *Methods for Size Estimating:* Function Points, approved by IFPUG (International Function Points User Group) [7] in addition to an increased set and lines of code, both new and existing ones.

Figure 2 is adapted from a Galorath illustration and shows gross categories of model inputs and outputs, but each of these represents dozens of specific input and output possibilities and parameters. The reports available from the model cover all aspects of input and output summaries and analyses numbers in the hundreds.

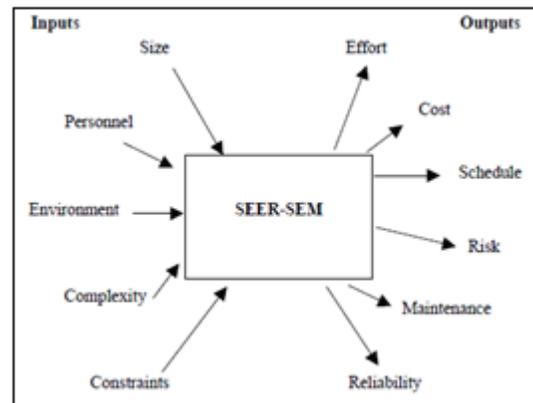


Figure 2. SEER-SEM Inputs and Outputs

As part of this effort, Galorath maintains a software project repository of approximately 6,000 projects (and growing). About 3,500 projects containing effort and duration outcomes are stored in a unified repository that can be readily accessed for studies. SEER is also available with data repositories from The International Software Benchmarking Standards Group (ISBSG) [8]. ISBSG provides the largest open repository of software project history and includes standardized, verifiable data from over 4,000 software projects.

These projects are from both defense and commercial sources representing many development organizations, permitting calibration of the model to a wide array of potential projects. Additional project outcomes, in the hundreds, are also available to the company, which has also collected sizing and other information on thousands of additional projects. Analysis involves running project data through SEER-SEM using a special calibration mode. The

model is essentially run backwards to find calibration factors.

Productivity factors are evaluated across different data attributes (e.g. platform, application, etc.) to detect trends. A variety of methods are used to mitigate outlier data points and control for variation. The variance in the data set is also used to establish default parameter ranges; nearly all settings accommodate risk. Model settings are updated as new trends are established.

SEER technology provides project results by generating a virtual project based on:

- *The SEER Modeling Engine:* SEER mathematical models are derived from extensive software project histories, behavioral models, and metrics. SEER for Software (SEER-SEM) employs a multi-faceted approach to project estimating, leveraging industry and/or company project histories and proven formulaic cost relationships.
- *SEER Knowledge Bases:* Serve as a virtual “in-house expert,” providing default values, ranges, and calibrations based on comparable software project histories.

Together, these capabilities enable users to develop first-look estimates when very little information is known, and to those estimates as details become available over time.

**Equations:** The equations of the model have not been edited for the public domain, although a few of the central algorithms of the model were published by Jensen [9] and collected here:

1. Size:  $s = c (td) (Ed)^{1/2}$

Where  $s$  is the size in SLOC,  $E_d$  is the total effort needed to complete the project, selected from a database of previous projects,  $c$  is a constant of the of the project called Technological Factor of Jensen, this factor reflects the effect of numerous costs drivers such as the hardware constraints, complexity of the program, the levels of the team experience and the programming environment.  $t_d$  is the total development time of the project.

2. Effort:  $e = 0.4 \left(\frac{s}{c}\right)^2 \left(\frac{1}{t^2}\right)$

Where  $e$  represents the Effort measured in MM and  $t$  is the time consumed since it began development.  $K$  is the total effort of the lifecycle.

### 3.3. SPR-Knowledge Plan – 1997

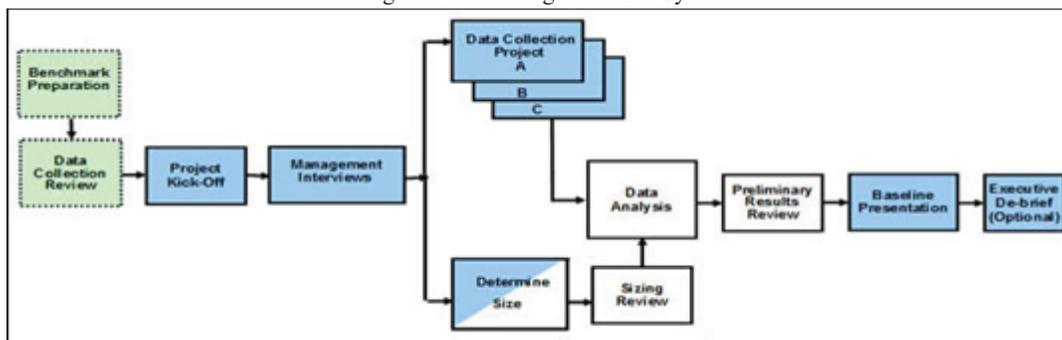
**Patent:** SPR – Software Productivity Research

SPR Knowledge PLAN is a commercial Windows-based software estimating tool from Software Productivity Research (SPR).

Launched in 1997, SPR Knowledge PLAN was the first powerful, knowledge-based software estimation tool to combine project estimation and scheduling in a flexible environment. This tool quickly creates and refines detailed project plans for seamless export to Microsoft Project or other enterprise project management systems.

SPR Knowledge PLAN is a software tool designed to help in planning software projects. With this tool, the user can effectively size each project and then estimate work, resources, schedule, and defects. User can even evaluate project strengths and weaknesses to determine their impact on quality and productivity.

Figure 3. Knowledge Plan LifeCycle



SPR Knowledge PLAN provides a complete and rational view of all tradeoffs among features, schedules, quality and costs. User can explore the cost/value implications of additional resources, more powerful languages, development tools, improved methods and other technical changes. User can also track milestones, schedules, resources, actual work effort, and defects found.

This estimation tool provides a bi-directional interface with project management applications to create an integrated, full life-cycle solution. For convenience, some project management functions such as critical path scheduling are offered.

Knowledge Plan collects information at the project level, using a “representative sample approach.” The

sample projects selected reflect the work patterns of user's organization itself, e.g., a mix of new projects, enhancements, and those with special factors such as high usage of contractors or integration of packages. In the analysis, the project data is summarized to create a complete and accurate picture of the organizational.

SPR uses its expertise in measurement to assist clients in establishing meaningful portfolio and project baselines. Then, SPR draws on its extensive industry knowledge base, derived from more than 14,531 completed software projects of all types, as a reference point against which to compare (benchmark against) client baseline data. It is the actual organizational data that forms the baseline and the comparative analysis against industry data that constitutes the benchmark.

#### 4. Conclusions

Software estimating is simple in concept, but difficult and complex in reality. The difficulty and complexity required for successful estimates exceeds the capabilities of most software project managers to produce effective manual estimates. The commercial software estimating tools can often outperform human estimates in terms of accuracy, and always in terms of speed and cost effectiveness.

However, no method of estimation is totally error-free. As mentioned before, the current "best practice" for software cost estimation is to use a combination of software cost estimating tools coupled with software project management tools, under the careful guidance of experienced software project managers and estimating specialists.

The fundamental objective raised in this research was the study and the analysis of three of the main software effort estimation methods, focused on Parametric models, that have been developed and then commercialized across the software engineering history.

With this review, we have tried to show how these estimation models work and which their main features are. Taking a clear view of these estimation models as a whole, we will be able then to understand how actual software project estimation companies has gained an important "piece" of the today software industry market.

#### Acknowledgement

We would like to thank the University of Alcalá for supporting this research (Ph.DC researchers support programme).

#### References

[1] Albrech, A. "Measuring Application Development Productivity". Proceedings of the IBM Application

Development Symposium, GUIDE/SHARE, California, USA, pp. 83-92, 1979.

[2] Albrecht, A.J., and Gaffney, J.E. "Software function, source lines of code, and development effort prediction: A software science validation," IEEE Transactions on Software Engineering (SE-9:6), pp 639-648, 1983.

[3] Aron, J. "Estimating Resources for Large Systems", In NATO Conference Report on Software Engineering Techniques, Eds. J.N. Buxton y B.Randel, Rome (Italy), 1969.

[4] Bisio, R. and F. Malabocchia. 'Cost estimation of software projects through case base reasoning', in Proc. 1st Intl. Conf. on Case-Based Reasoning Research & Development . Springer-Verlag, 1995.

[5] Boehm, B., Clark, B., Horowitz, E., Madachy, R., Selby, R. and Westland, C. "Cost Model for Future Software Life Cycle Processes: COCOMO 2.0". Annals of Software Engineering Special Volume on Software Process and Product Measurement, Eds. J.D. Arthur, S.M. Henry and J.C. Baltzer, Ed. AG Science Publishers, Amsterdam (Netherlands), Vol. 1, 1995.

[6] Freiman, F.R., and Park, R.E. "The PRICE software cost model," Proceedings of the IEEE National Aerospace and Electronics Conference NAECON, p. 500. New York, USA, 1979.

[7] IFPUG, International Function Points Users Group, "Function points counting practices manual 4.1.1". Ohio, USA, 1999.

[8] ISBSG, International Software Benchmarking Standards Group repository, Release 10. <http://www.isbsg.org>

[9] Jensen, R.W. "A macro-level software development cost estimation methodology". Conference Record of the Fourteenth Asilomar Conference on Circuits Systems & Computers, p. viii+520, 1979.

[10] Jensen R. "An improved Macrolevel Software Development Resource Estimation Model". Proceedings 5th ISPA Conference, pp. 88-92, 1983.

[11] Jones, C. "Programming Quality and Programmer Productivity", IBM Technical Report TR-02-764, pp. 39-63, 1977.

[12] Jones, C. "How software estimation tools work". SPR Technical Report, Version 5 – February 27, 2005.

[13] Jørgensen, M. "A Review of Studies on Expert Estimation of Software Development Effort". Journal of Systems and Software 70 (1-2): pp. 37-60, 2004.

[14] Norden, P.V. "Curve fitting for a model of applied research and development scheduling," A-IBM Systems Journal (2:3), 1958.

[15] Putnam, L.H. "A general empirical solution to the macro software sizing and estimating problem," IEEE Transactions on Software Engineering (SE-4:4), pp 345-361, 1978.

[16] Putnam, Lawrence H., and Ware Myers. "Measures for Excellence: Reliable Software on Time" Within Budget, Englewood Cliffs, NJ: Yourdon Press, 1992.

[17] Shepperd, M.J., C. Schofield, and B.A. Kitchenham. 'Effort estimation using analogy', in Proc. 18th Intl. Conf. on Softw. Eng. Berlin: IEEE Computer Press, 1996.

# Information-Theoretic Metrics for Project-Level Scattering and Tangling

Erik Linstead<sup>1</sup>, Lindsey Hughes<sup>1</sup>, Cristina Lopes<sup>2</sup>, Pierre Baldi<sup>2</sup>

<sup>1</sup> Department of Math and Computer Science. Chapman University, Orange, CA.

<sup>2</sup> School of Information and Computer Sciences. University of California, Irvine.

linstead@chapman.edu, hughe120@chapman.edu, lopes@ics.uci.edu, pfbaldi@ics.uci.edu

## Abstract

*We develop and apply unsupervised statistical topic models to identify functional components of source code and introduce new information-theoretic techniques for measuring and monitoring software complexity in the form of scattering and tangling entropies. This approach formalizes the basic scattering and tangling concepts of Aspect-Oriented Programming by considering the distributions of topics over files and files over topics, and formulates project-level metrics for crosscutting. These proposed metrics, Area-Under-Scattering-Curve and Area-Under-Tangling-Curve provide a probabilistic basis for modeling facets of crosscutting in isolation or as components of more sophisticated complexity models. We demonstrate our approach through an empirical analysis of scattering and tangling of five well-known open source projects, in addition to studying the evolution of scattering and tangling for multiple versions of Eclipse and ArgoUML.*

## 1. Introduction

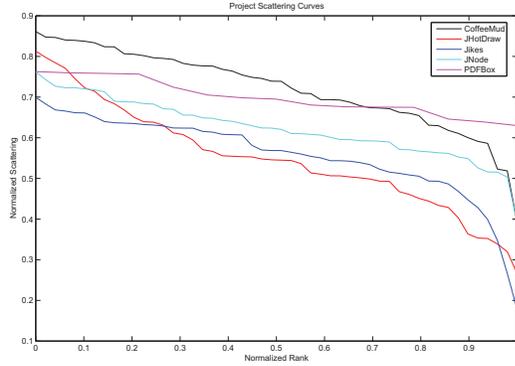
Repositories of software, available publicly over the Internet or privately within organizations, create several new challenges and opportunities for statistical data mining and machine learning. One fundamental challenge is to develop approaches for automatically measuring, monitoring, and better understanding software complexity, with the long-term goal of informing and guiding software development and engineering. One key articulation of software complexity is provided by the field of Aspect-Oriented Programming (AOP) [7] which, over the last decade, has shaped the way the software engineering community evaluates the design, modularity, and complexity of software artifacts.

Central to AOP is the hypothesis that software consists of concerns, or aspects, that represent the various functional concepts represented in source code, and that these concerns can transcend package, class, and source file boundaries. Concerns (functional concepts) that are manifested

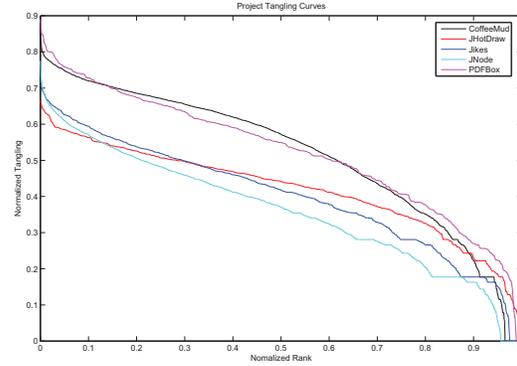
in many source files are said to be “scattered.” Similarly, source files that contain many concerns are said to be “tangled.” While the existence of scattering and tangling is widely accepted throughout the software engineering community, and their anecdotal presence used as a major indicator of software complexity, statistically-grounded, unsupervised data mining techniques for extracting concerns and quantifying scattering and tangling at the component and project level have so far remained elusive.

In our previous work [9] we have partially addressed this challenge by adapting statistical topic modeling methods, in particular Latent Dirichlet Allocation (LDA) [2], to automatically extract concerns from source code and derive a precise, operational, definition of scattering and tangling at multiple scales of software granularity, from individual projects to thousands of projects. While our results demonstrated the effectiveness of probabilistic text mining in serving as a basis for identifying and quantifying crosscutting for individual concerns and files, unified metrics for project-level scattering and tangling were not considered. In this paper we extend our work in this area to formulate two new metrics, Area-Under-Scattering-Curve (AUSC) and Area-Under-Tangling-Curve (AUTC), to succinctly measure these facets of AOP at the project-level. In addition to providing an information-theoretic means for capturing software complexity, either directly or as part of more sophisticated models, our preliminary results indicate substantial promise in applying AUSC and AUTC to quantify aspect-oriented software characteristics in support of automated software quality analysis, such as the analysis of software complexity evolution.

The remainder of the paper is organized as follows. In section 2 we briefly describe the data used in the empirical validation of our methods. Section 3 gives an introduction to LDA-based topic models, and provides formal definitions of scattering and tangling that can be derived from file-topic and topic-file distributions as well as definitions for the AUSC and AUTC metrics themselves. This is followed by a discussion of the results obtained in applying our techniques to our data set in section 4, including the ap-



**Figure 1. Scattering Curves for Five Open Source Projects**



**Figure 2. Tangling Curves for Five Open Source Projects**

plication of AUSC and AUTC to the problem of summarizing complexity evolution. Finally, we briefly discuss related and future work in section 5.

## 2. Data

To exercise our approach, we apply our topic modeling technique to five open source projects: JHotDraw, Jikes, PDFBox, JNode, and CoffeeMud. Analysis of these projects is facilitated by Sourcerer [8], an open source Java software repository and mining infrastructure. Together these projects represent a collection of well-known, non-trivial software products of varying size and complexity. Additionally, multiple versions of Eclipse and ArgoUML are employed to apply our technique to the study of software complexity evolution.

## 3. Methodology

In the LDA model for text, the data consists of a set of documents. The length of each document is known and each document is treated as a bag of words. Let  $D$  be the total number of documents,  $W$  the total number of distinct words (vocabulary size), and  $T$  the total number of topics present in the documents. While non-parametric Bayesian and other methods exist to try to infer  $T$  from the data, here we assume that  $T$  is fixed.

The model assumes that each topic  $t$  is associated with a multinomial distribution  $\phi_{\bullet,t}$  over words  $w$ , and each document  $d$  is associated with a multinomial distribution  $\theta_{\bullet,d}$  over topics. More precisely, the parameters are given by two matrices: a  $T \times D$  matrix  $\Theta = (\theta_{td})$  of document-topic distributions, and a  $W \times T$  matrix  $\Phi = (\phi_{wt})$  of topic-word distributions. Given a document  $d$  containing  $N_d$  words,

for each word the corresponding  $\theta_{\bullet,d}$  is sampled to derive a topic  $t$ , and subsequently the corresponding  $\phi_{\bullet,t}$  is sampled to derive a word  $w$ . A fully Bayesian model is derived by putting symmetric Dirichlet priors with hyperparameters  $\alpha$  and  $\beta$  over the distributions  $\theta_{\bullet,d}$  and  $\phi_{\bullet,t}$ . For instance, the prior on  $\theta_{\bullet,d}$  is given by

$$D_{\alpha}(\theta_{\bullet,d}) = \frac{\Gamma(T\alpha)}{(\Gamma(\alpha))^T} \prod_{t=1}^T \theta_{td}^{\alpha-1}$$

and similarly for  $\phi_{\bullet,t}$ .

The probability of a document can then be obtained in a straightforward manner by integrating the likelihood over parameters  $\phi$  and  $\theta$  and their Dirichlet distributions. The posterior can be sampled efficiently using Markov Chain Monte Carlo Methods (Gibbs sampling) and the  $\Theta$  and  $\Phi$  parameter matrices can be estimated by maximum a posteriori (MAP) or mean posterior estimate (MPE) methods.

Once the model has been formalized, applying LDA to software requires tools to pre-process source code and convert it into compatible representations for the algorithm, required as input parameters. Of these parameters, the most important is the word-document matrix, which represents the occurrence of words in individual documents. To produce the word-document matrix for our input data we have developed a comprehensive tokenization tool tuned to the Java programming language. This tokenizer includes language-specific heuristics that follow the commonly practiced naming conventions.

Once the topics of a project are identified, the quantification of scattering and tangling can be easily derived from the corresponding distributions, for instance in terms of entropy. For example, if the distribution of topic  $t$  across modules  $m_0 \dots m_n$  is given by  $p^t = (p_0^t \dots p_n^t)$  then scattering of topic  $t$  can be measured by the entropy

$$H(p^t) = - \sum_{j=0}^{j=n} p_j^t \log p_j^t \quad (1)$$

Likewise, if the distribution of the module  $m$  across the topics  $t_0 \dots t_r$  is given by  $q^m = (q_0^m \dots q_r^m)$  then tangling in module  $m$  can be measured by the entropy

$$H(q^m) = - \sum_{j=0}^{j=r} q_j^m \log q_j^m \quad (2)$$

Scattering and tangling entropies may be normalized to values between 0 and 1 by dividing by the log of the number of source files and number of topics, respectively. The resulting values, however, are only interpretable at the level of individual concerns and files. To quantify scattering and tangling at the project level while still taking advantage of the unsupervised, information-theoretic framework of LDA, additional metrics are required. Here we propose two such metrics: Area-Under-Scattering-Curve (AUSC) and Area-Under-Tangling-Curve (AUTC).

To produce scattering and tangling curves we plot normalized entropy measures for each concern or file in a project in descending order, also normalizing the domain from zero to one. The resulting curves provide an intuitive visualization for scattering and tangling behavior within a project, and also allow for comparative evaluation of scattering and tangling between projects. Figure 1 and figure 2 depict the scattering and tangling curves for the five projects selected for our empirical analysis.

Once the curves are acquired, AUSC and AUTC are obtained by integrating over the desired plot, easily achieved in practice by applying numerical integration techniques such as trapezoidal approximation, and essentially analogous to the common data mining practice of calculating area-under-curve (AUC) for receiver operating characteristics (ROC). AUSC is then a single value between 0 and 1.0 representing the degree to which the project is scattered, with a value of 0 denoting a project with no crosscutting, and 1.0 representing a project whose every concern is uniformly distributed across all source files (maximum entropic scattering). Similarly, an AUTC of 0 represents a project whose files are each assigned to only a single concern, and an AUTC of 1.0 representing a project where every source file is a uniform distribution over all concerns (maximum entropic tangling).

Unlike the scattering and tangling curves themselves, AUSC and AUTC provide a direct, numerical means for quantifying project-level scattering and tangling without the need for visual interpretation. This simplification also facilitates the incorporation of these metrics into other complexity models, allowing them to be used in conjunction with other indicators of software quality, such as the well-known measures proposed by Halstead [6].

**Table 1. AUSC and AUTC Metrics for Selected Data Set.**

Project Name	AUSC	AUTC
CoffeeMud	0.7221	0.5174
JHotDraw	0.5492	0.4191
Jikes	0.5593	0.4008
JNode	0.6244	0.3597
PDFBox	0.699	0.5227

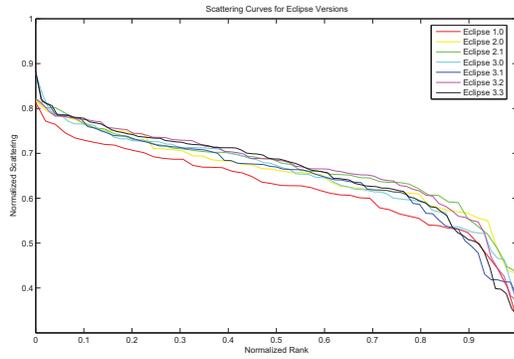
## 4. Results

Using LDA we extracted concerns for each of the five projects in our benchmark data set, and quantified scattering and tangling for each. The number of concerns to be extracted for each project was determined empirically, with the upper limit being 100 topics for the larger projects. The file-topic and topic-file distributions were then leveraged to compute entropies per equations 1 and 2 in section 3, and in turn plotted as scattering and tangling curves.

Figure 1 contains the scattering profiles for our data, with the y-axis denoting the normalized scattering entropy of each concern, and the x-axis denoting the normalized rank of each concern when ordered by the magnitude of crosscutting. From the figure we see that CoffeeMud and PDFBox exhibit the most scattering of all projects. This is especially telling in the case of PDFBox as it is the smallest project in the collection and contains only 370 source files compared to CoffeeMud's 2,900 files. Continuing to examine projects, we see that JHotDraw and Jikes appear comparable in terms of scattering, though it is not easily determined from the visualization which project demonstrates higher crosscutting and hence motivating the need for a unified project-level metric.

One can interpret the tangling curve in figure 2 in a similar fashion. Again, CoffeeMud and PDFBox compete for the largest tangling profiles with noticeably higher curves than the remaining projects, though inspection of the curves themselves does not easily allow for computing exact differences or summary statistics. Despite being in the middle of the field for scattering, JNode demonstrates the least amount of tangling of all five projects, sitting noticeably below JHotDraw and Jikes. These remaining projects again appear to contain similar amounts of tangling, but as above an exact ordering remains elusive leveraging the visualization alone. In examining PDFBox and CoffeeMud, for example, it is not immediately clear which project claims the highest score.

Despite the convenience of our scattering and tangling curves in providing general insight into project complex-

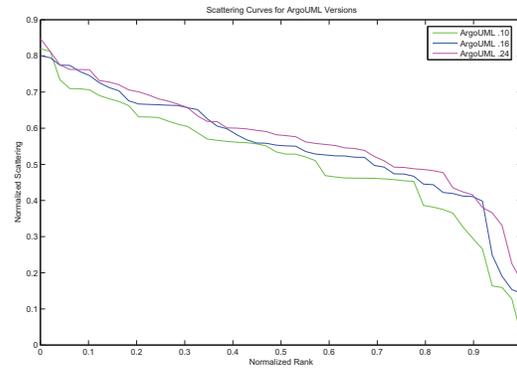


**Figure 3. Scattering Curves for Several Eclipse Versions**

ity, these curves alone are insufficient for precisely quantifying these aspect-oriented characteristics so as to allow a total ordering to be derived. To this end, we apply our area-under-curve metrics to gain an exact understanding of scattering and tangling for our sample data set. Table 1 provides AUSC and AUTC scores for each project. Looking at the column for AUSC we clearly see that CoffeeMud has the highest scattering at 0.72, with PDFBox close behind at 0.70. Moreover, the ambiguity between JHotDraw and Jikes is easily resolved, with Jikes demonstrating only slightly higher scattering at 0.56 compared to JNode’s 0.55. Computing summary statistics is made simple for the purpose of empirical analysis, and across all projects one observes a mean AUSC of 0.631, with a standard deviation of 0.079.

Turning to AUTC, we find a mean score across all projects of 0.444 with a standard deviation of 0.073. As with scattering CoffeeMud and PDFBox top the list with the highest scores, though in the case of tangling PDFBox scores slightly higher at 0.523 than CoffeeMud at 0.517. The scores of Jikes and JHotDraw are again easily differentiated using this metric, and one sees that tangling manifests itself to a slightly greater extent in JHotDraw’s source files with a score of 0.419 compared to 0.401 for Jikes.

Together AUSC and AUTC provide a simple and intuitive method for capturing scattering and tangling behavior at the project level. A key benefit is that the approach itself is based on unsupervised machine learning, and does not require any pre-processing to identify concerns in the source code or the relations between them. The metrics themselves are grounded in information theory, which in turn yields ease of interpretation, and, if desired, the ability to leverage additional statistical methods to refine the underlying concern mixture model. While the development of more sophisticated models of software complexity is beyond the scope of this paper, it is our intent that AUSC and AUTC



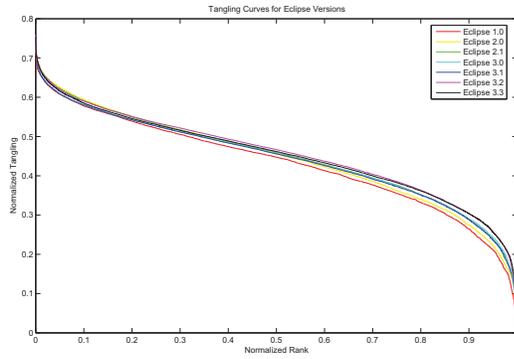
**Figure 4. Scattering Curves for Several ArgoUML Versions**

serve as convenient parameters for such models, and facilitating empirical analysis and validation of software quality and maintainability. To this end, in the next section we demonstrate how our metrics can be applied to the study of software evolution.

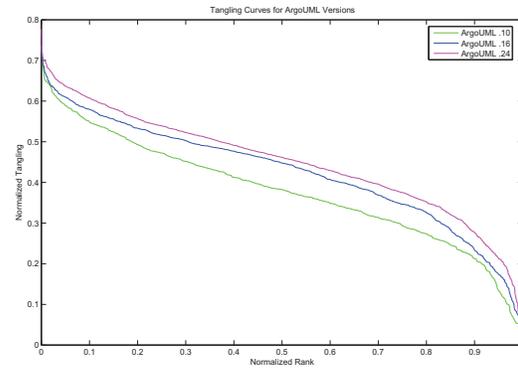
#### 4.1 AUSC and AUTC as Metrics for Software Evolution

In recent years, an increasing trend in program understanding is the mining and analysis of version history. Not only has the open-source movement made available a large number of projects for empirical study, but the relative ease in which researchers can acquire multiple releases of the same project has fueled new directions in automated methods for summarizing software evolution. Recent techniques have focused on learning the relationships among developers over time, discovering code changes responsible for introducing bugs, and predicting the number and types of bugs for new releases based on historical data. In many cases it is the metadata from configuration management systems (such as CVS and Subversion) that is leveraged for mining purposes, rather than the source code itself.

When studying software evolution a natural question that arises is whether the complexity of a project increases, decreases, or remains constant as the project matures. Feature integration, feature deprecation, and code refactoring are common events on the software release timeline, and these events impact the overall complexity of the product in addition to the form and function of the source code itself. Our AUSC and AUTC metrics provide a simple method with which to track changes in complexity, and we demonstrate that here by analyzing several versions of Eclipse and ArgoUML. Eclipse is a widely used software development platform that was initially developed as an IBM product



**Figure 5. Tangling Curves for Several Eclipse Versions**



**Figure 6. Tangling Curves for Several ArgoUML Versions**

before being reincarnated as an open-source project. ArgoUML is a popular software design tool that started life as a university research project before moving to the open-source domain.

Figures 3 and 4 present the scattering curves for 7 major versions of Eclipse and 3 major versions of ArgoUML, respectively. At first glance one can see that over time ArgoUML appears to exhibit an increase in scattering complexity, compared to Eclipse which exhibits a noticeable increase after version 1.0 but appears relatively stable thereafter. This observation is precisely quantified with the AUSC metrics provided for each project in tables 2 and 3. Indeed, we see that for ArgoUML project-level scattering complexity is monotonically increasing over the release timeline, with a rise from approximately 0.51 to 0.58. Such monotonic behavior is not exhibited by Eclipse, which experiences several increases and decreases in scattering complexity over the release timeline, with a maximum AUSC score of 0.6713 being measured in version 3.2 before sinking to 0.6618 in version 3.3.

A similar analysis can be provided for the tangling curves provided in figures 5 and 6. Once again we see that the curves for the seven releases of Eclipse remain relatively constant for each release, with no clear trend of increase or decrease over time. ArgoUML, on the other hand, exhibits a substantial increase in tangling from version .10 to version .24. Turning to the AUTC values in table 3, we verify this observation with a project-level tangling score that rises from approximately 0.38 to 0.45. This is in contrast to Eclipse, from which we measure a difference of only .03 from a minimum AUTC score of 0.43 in version 1.0 and a maximum score of 0.46 in version 3.2.

Due to its early history as a commercial IBM product, one would expect that Eclipse development was driven by formal processes and best practices from its inception, man-

dating a modular and stable architecture amenable to substantial feature additions. As such, one may hypothesize that, over time, scattering and tangling would remain relatively constant compared to a more “grass-roots” open source project like ArgoUML. This hypothesis is indeed supported by comparing AUSC and AUTC metrics for both projects over time.

If Eclipse seems to fair better than ArgoUML in terms of managing complexity over time, why then is it that the AUSC scores for all versions of Eclipse are substantially higher than those of ArgoUML? The answer lies in the fact that Eclipse consists of over 16,000 source files, compared to a mere 1,400 for ArgoUML. Moreover, Eclipse is a general-purpose integrated development environment, containing components for a multitude of development tasks, as opposed to ArgoUML which provides facilities only for software design. Thus, the concerns manifested in the Eclipse codebase do tend to exhibit a greater degree of cross-cutting than those in the ArgoUML codebase, an observation which further validates the scores given by the AUSC metric.

## 5. Related and Future Work

While a variety of techniques exist for mining aspects from source code, space limitations prevent a detailed review here. The reader may refer to [1] for a detailed evaluation of related work pertinent to our use of LDA as a basis for concern identification. At the highest level the greatest advantage of our technique is its unsupervised nature, which removes the need for human participation in the identification and mapping phases of concern extraction. Additionally, Kellens et al. offer a detailed survey of seven different code-based aspect mining techniques [3]. Our approach

**Table 2. AUSC and AUTC Metrics for Eclipse Versions.**

Version	AUSC	AUTC
1.0	0.6272	0.4322
2.0	0.6604	0.4436
2.1	0.6702	0.4483
3.0	0.6583	0.4461
3.1	0.6525	0.4439
3.2	0.6713	0.4556
3.3	0.6618	0.4520

can be characterized with the attributes taken from their framework, and exhibits a strong advantage of substantial empirical analysis in our past and current work, as well as the fact that no strong preconditions are required.

Metrics used in aspect mining usually deal with measures of crosscutting, scattering and tangling. These metrics are used in two different ways: to measure the precision and accuracy of the aspect mining technique in capturing crosscutting concerns [4], and to derive various conclusions regarding the effect of crosscutting, by correlating them with selected quality attributes for software [5]. While our previous work focused on the former, the AUSC and AUTC metrics defined here support studies of the latter type, providing an additional empirical measure to researchers and practitioners in need of project-level indicators of scattering and tangling. As such these metrics may be used for correlation with existing software metrics, leveraged as components of more broadly-scoped metrics or models, and to continue the study of the fundamental hypotheses of AOP. Moreover, the data calculated as part of the scattering and tangling curves themselves may be used to support finer-granularity studies at the class or component level, the goal of many previous metrics such as those discussed in [10].

We are currently in the process of expanding the preliminary work presented here by analyzing the over 12,000 projects currently indexed by Sourcerer. From our initial observations we are building parametric models for scattering and tangling which can be leveraged for more precise analytical quantification of AUSC and AUTC, as well as provide a functional basis for understanding crosscutting at the large and small scale. In the future we hope the metrics described here, in addition to the parametric models currently being explored, will facilitate an Internet-scale analysis of the effects of scattering and tangling on software quality, maintainability, and evolution.

**Table 3. AUSC and AUTC Metrics for ArgoUML Versions.**

Version	AUSC	AUTC
.10	0.5126	0.3791
.16	0.5539	0.4248
.24	0.5779	0.4595

## References

- [1] P. F. Baldi, C. V. Lopes, E. J. Linstead, and S. K. Bajracharya. A theory of aspects as latent topics. In *OOPSLA '08*, pages 543–562, New York, NY, USA, 2008. ACM.
- [2] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, January 2003.
- [3] G. S. Cojocar and G. Şerban. On some criteria for comparing aspect mining techniques. In *LATE '07: Proceedings of the 3rd workshop on Linking aspect technology and evolution*, page 7, New York, NY, USA, 2007. ACM.
- [4] G. S. Cojocar and G. Şerban. On some criteria for comparing aspect mining techniques. In *LATE '07: Proceedings of the 3rd workshop on Linking aspect technology and evolution*, page 7, New York, NY, USA, 2007. ACM.
- [5] M. Eaddy, T. Zimmermann, K. D. Sherwood, V. Garg, G. C. Murphy, N. Nagappan, and A. V. Aho. Do crosscutting concerns cause defects? *IEEE Trans. Softw. Eng.*, 34(4):497–515, 2008.
- [6] M. H. Halstead. *Elements of Software Science (Operating and programming systems series)*. Elsevier Science Inc., New York, NY, USA, 1977.
- [7] G. Kiczales, J. Lamping, A. Menhdhekar, C. Maeda, C. Lopes, J. Loingtier, and J. Irwin. Aspect-oriented programming. In M. Akşit and S. Matsuoka, editors, *Proceedings European Conference on Object-Oriented Programming*, volume 1241, pages 220–242. Springer-Verlag, Berlin, Heidelberg, and New York, 1997.
- [8] E. Linstead, S. Bajracharya, T. Ngo, P. Rigor, C. Lopes, and P. Baldi. Sourcerer: mining and searching internet-scale software repositories. *Data Min. Knowl. Discov.*, 18(2):300–336, 2009.
- [9] E. Linstead, P. Rigor, S. Bajracharya, C. Lopes, and P. Baldi. Mining internet-scale software repositories. *NIPS 2007: Advances in Neural Information Processing Systems 20*, 0, 2008.
- [10] R. E. Lopez-herrejon and S. Apel. Measuring and characterizing crosscutting in aspect-based programs: Basic metrics and case studies. In *In Proc. Intl Conf. Fundamental Approaches to Software Engineering*, page 422437. Springer, 2007.

# Synchronization Complexity Metric

Peter Yastrebenetsky  
The Open University of Israel  
pyastreb@ieee.org

Mark Trakhtenbrot  
Holon Institute of Technology  
The Open University of Israel  
markt@hit.ac.il

## Abstract

A new metric is introduced that characterizes complexity of a program based on the kind and amount of means used for synchronization between its concurrent components. Similar to McCabe's metric for single-thread programs, the new Synchronization Complexity metric allows for assessment of the amount of tests needed to achieve a proper coverage in testing of a concurrent program. It also enables comparison between different application's implementations based on their synchronization complexity.

## 1. Introduction

Software developers are often faced with questions such as: *How to make a better quality product? How to make it testable? How to make it maintainable?* These questions were valid in 1970's, and they're valid still. There are numerous ways to answer these questions, depending on the used definition of "testable" or "maintainable", or on how the overall product quality is assessed.

One of the popular approaches is the use of software code metrics. It is based on selecting measures that are believed to be predictive of an aspect of application quality, and are used as an aid to requirements, design, test, and code reviews [1]. Many industry coding standards (like [5, 6]) specify requirements for the values of these measures.

McCabe's Cyclomatic Complexity (CCN) metric [1, 10, 12] is most popular for assessment of complexity of a single thread program. This metric counts the number of branches in the code, and provides the lower bound for the amount of tests needed to achieve the *full coverage* of the application (under the branching coverage model).

However, CCN is not enough for multi-thread programs. The reason is that during different executions of the same test scenario, different *interleavings* might occur, and this is not taken into the account in the CCN calculation.

While CCN allows estimating the effort needed to test a single-process program, no metric has been suggested so far to efficiently evaluate the impact of the concurrency on the efforts needed to thoroughly test a program.

The trivial estimation of the cost incurred by concurrency to the required test efforts would be

proportional to the amount of all possible execution sequences (i.e. interleavings) in the concurrent program. This estimation is exponential with respect to the amount of states in program's threads. However, in reality, there is no need to consider all possible interleavings. The main goal of this work is a more accurate estimation for interleavings that should actually be considered.

A number of tools support estimation of the actual interleavings coverage during the test process. ConTest [3,4] does this based on internal counters of the synchronization points' accesses in the program under test. CHESS [11] controls the scheduling and preemption of tasks, and allows for systematic coverage of all possible interleavings. However, such tools are used during the actual testing of a program, and cannot be used for planning and estimating efforts, or for comparing implementation options during the development. In this work we attempt to fill this void. The suggested metric allows a more controlled use of these tools so that at each point of the test execution the tester would be able to know how many paths were covered, and more importantly – how many are still remain to be covered.

Section 2 discusses the different types of interleavings and provides an alternative definition for the term "*interleaving*". This allows us to define in section 3 a novel Synchronization Complexity metric that is an extension of CCN for multi-thread programs. The new metric provides a way to estimate the amount of so called *intentional interleavings* in concurrent programs. Section 4 presents data on calculation of the relevant cost parameters for various types of synchronization statements. Finally, section 5 discusses the results of analysis of a real life concurrent application with regards to the new metric.

## 2. Interleavings

### 2.1 Intentional and unintentional interleaving

We distinguish between two different types of interleavings – "intentional" and "unintentional".

*Unintentional interleaving* is characterized by its unpredictability: it is caused by an interrupt or another unpredictable external event (such as buffer overflow). Moreover, a hardware interrupt may cause interleaving on

the machine instructions level even during execution of a single-thread program; this happens due to activation of the relevant interrupt handler.

In this work we ignore unintentional interleavings, and only consider intentional ones. *Intentional interleaving* is caused by an explicit synchronization statement (placed by designer in the code) which by its nature serves as a potential switch between the various threads of the multithread application. It thus can cause several alternative execution paths even when the same input is considered. The various types of synchronization statements are discussed in section 2.2 below.

## 2.2 Synchronization Points

Every instance of a synchronization statement (e.g. of `fork`) in code defines a separate *synchronization point*. In a sense, this is similar to a branching statement in a single-thread application, except that the result of this “concurrency branching” is a different order of state changes in the program's threads.

In Table 1 below we consider a sub-set of what we call *basic* synchronization statements. Table 1 is based on the list of synchronization statements types considered in [3] as part of the wider discussion there on synchronization test coverage and ConTest implementation. Statements in the first group (items 1 to 5) have been chosen based on the coverage definition in [3]. In order to complete the list, we added more statements for cases not addressed in [3]. Namely, these are: volatile access for cases where the application is not encapsulated to the software, thread/task creation and voluntary preemption (see items 6, 7 and 8).

Each synchronization statement may have a different function name in different programming languages; instead, we use generic names describing the statement functionality.

Statements in Table 1 are basic in the sense that they are common and behave similarly in different operating environments. For example, synchronization point of type *wait* may be implemented as a call to one of the functions `pthread_cond_wait`, `sem_wait`, or `mq_receive` in `pthread` library. Each of these implements different functionality, but the synchronization effect is the same: block until an event occurs.

## 2.3 Interleaving definition based on the execution paths graph model

According to the definition in [9], an interleaving is a total order relation on the set  $P_E$  of shared data access events in the application  $P$ . For each set of events  $P_E$  (for the given program  $P$  under a given input), there may be several different interleavings.

Below we suggest an alternative definition of

interleaving. It can be shown that it is equivalent to that in [9]; but it is more convenient for complexity measuring.

**Table 1. Types of basic synchronization statements (by functionality)**

	Synchronization statement types	Description
1	<b>Try-lock</b>	Try to enter a critical section; a mutex acquire call that can fail (without blocking the caller) or succeed. Example: a C pthread library <code>pthread_mutex_trylock</code> call
2	<b>Lock</b>	Entrance to a critical section; a mutex acquire call that can block the caller (till the mutex is unlocked) or succeed instantly. Example: a C pthread library <code>pthread_mutex_lock</code> call.
3	<b>Unlock</b>	Exit from a critical section; a mutex release call. E.g. a C pthread library <code>pthread_mutex_unlock</code> call.
4	<b>Wait</b>	Wait on a condition. Examples: <code>sleep</code> , a <code>select</code> call in a POSIX system, or <code>sem_wait</code> call.
5	<b>Notify</b>	Inform a waiting (on <b>Wait</b> ) thread that the condition it is waiting for has been met. E.g. : POSIX <code>signal</code> call, or <code>sem_post</code> call.
6	<b>Pass Control</b>	Release the CPU control by a thread. For example, the Java method <code>yield</code> .
7	<b>Volatile access</b>	Access to a non-synchronized variable with concurrent access, for example a <code>volatile</code> variable in C.
8	<b>Task/Thread Initiation</b>	Creation of a new thread. Examples: a <code>pthread_create</code> call, a POSIX <code>fork</code> system call or C <code>exec</code> call.

In the *execution paths graph*  $G(S,E)$ , set  $S$  of nodes is the product of states in all possible threads existing throughout the application life:  $S = S_1 \times S_2 \times \dots \times S_n$ . This means that a state of the entire application is in fact an  $n$ -tuple consisting of current states in all  $n$  threads.

Each edge  $e \in E$  connects two states  $p, q \in S$ , so that  $p$  and  $q$  differ by exactly one element of the tuple. In other words, each edge in the path changes state of exactly one thread in the application.

Each path in  $G(S,E)$  represents an execution of the multithreaded program. We call these paths *interleavings*.

Note that threads can be created and terminated dynamically during the application execution. However the amount of threads is always finite (for example, due to limitations of the operating system); throughout this paper we assume that a realistic estimation for this amount is known in advance. Clearly, thread's state (and hence the relevant component of the tuple) can change only when it

is alive, i.e. after its creation and before it is terminated.

### 3. Synchronization complexity metric

Synchronization Complexity metric (SCM) defined here is used to estimate the required amount of tests needed for full coverage under the synchronization coverage model [3]. Under this coverage criterion, each branch of code execution that includes a synchronization statement has to be tested at least once (i.e. for each synchronization point, all its interleaving options have to be covered at least once).

Below we first consider cost parameters used to calculate SCM, and then define the metric itself.

#### 3.1 Cost parameters definition

For each of the synchronization types  $T$  listed above, we define the following parameters:

**Interleaving potential (IP).** This is the *minimum* number of interleavings (branches in the execution paths graph) created by a synchronization point of type  $T$  in the given operating environment. This is the minimum number of other threads that can preempt the current one at the considered synchronization point.

For each operating environment the value of  $IP$  remains constant, and is not data dependent. For example, once  $IP$  is calculated for *Task Initiation* under certain operating environment, the obtained value can be used in evaluation of SCM for all programs that use this synchronization type and intended for that environment. E.g. consider a `fork` system call in UNIX, where tasks with the same priority are treated in a Round-Robin manner. In this case control can switch to the newly created task or remain in the calling task, or switch to some other ready-to-run task (if there are any). Thus, the interleaving potential of this synchronization type is 2: there are *at least* two possible interleavings – switch to the newly created task, or not. When the program is written, it is not known which of the two will occur at the run-time; in each run, a different interleaving may occur.

Situation changes when priorities are fixed and no two tasks can have the same priority, as in  $\mu$ C/OS operating system used in various embedded devices. Namely, of the two tasks (the calling and the spawned) control will be given to the one with a higher priority. The decision is deterministic (since the priority is defined in the spawn command by the caller), and will always be the same when the program is run under the same input. Therefore, in this case the  $IP$  of *Task Initiation* equals to 1.

Thus, in order to properly use this parameter, it is necessary to be familiar with specifics of the target operating environment for which the application is being tested. A sample analysis of such parameters for some environments is presented in Section 4.

**Competition potential (CP).** Value of this parameter equals to the total number of threads competing for the synchronization point, i.e. threads that can be in their ready-to-run state immediately after execution of the synchronization statement in this point.

It is data dependant and may change between various applications and synchronization points within a single application. In terms of the graph representation discussed above, this is the number of tuples immediately reached from the current state as the result of the synchronization point execution.

Recall that  $IP$  depends only on the underlying scheduling algorithm, but not on the program logic. For  $CP$  the situation is the opposite: it doesn't depend on the scheduling algorithm, but may vary between different instances of the synchronization statement in the code. This is because the amount of threads created by the program and competing for a synchronization point depends on the program logic.

For example, for *Task Initiation* under the Round-Robin scheduling algorithm the  $IP$  equals to 2 (see above), but the  $CP$  (i.e. the number of tasks to which the control may pass in this particular instance) is variable, and may change throughout the different places in the application.

#### 3.2 Formal Definition of Synchronization Complexity Metric

While the classic McCabe's CCN is the most common branching complexity metric for sequential programs, SCM can be viewed as its extension to the case of concurrent programs. SCM can be used to estimate the minimum amount of tests required to guarantee a full branching coverage for all synchronization points in a program, where branching appears as a result of concurrency.

According to [12], the minimum requirement to a metric in order to be used for software measurement is that it belongs to the so called ordinal scale category; such metric can be applied for ranking of programs.. It can be shown that SCM, as defined below, fulfils this requirement.

SCM is formally defined as follows:

$$SCM = \sum (CCN_{sp} * IP_{sp}^{CP_{sp}-1})$$

This formula combines the classic CCN for a single-thread program (when  $CP=1$ ) with the newly defined parameters (interleaving potential and competition potential). Here, for each synchronization point  $sp$  :

- $CCN_{sp}$  is the cyclomatic complexity number of the branch at which the synchronization point was detected (see definition and a detailed example below)

- $IP_{sp}$  is the interleaving potential of the synchronization point. This value depends on the synchronization type.

- $CP_{sp}$  is the competition potential of the synchronization point. This value is data-dependant.

In order to find  $CCN_{sp}$  for a given synchronization point  $sp$ , we first define a sub-graph  $G_{sp}(S_{sp}, E_{sp}) \subseteq G(S, E)$  as

follows:

$S_{sp}$  – all states in  $S$  reachable from state  $b_{sp}$  in which the application was *after* executing the *most recent* branching statement (for example `if` or `while`), on the execution path leading to  $sp$ .

$E_{sp}$  – all edges in  $E$  that are connected to (i.e. enter into or exit from) states in  $S_{sp}$

$CCN_{sp}$  is then defined as cyclomatic complexity of  $G_{sp}$

We illustrate the definitions with the following simple example of  $CCN_{sp}$  calculation.

**Example.** Consider the following program:

```
#include <unistd.h>
#include <stdlib.h>
int main() {
    int pid = fork();
    if (pid != 0) {
        sleep(1);
        printf("\nChild Process\n");
    }
    else {
        sleep(1);
        printf("\nParent Process\n");
    }
    return pid;
}
```

Here the `fork` call is a synchronization point on the top branching level, and hence its  $CCN_{sp}$  equals to the  $CCN$  for the entire program (in this case 2) if analyzed as a simple single threaded application.

The `sleep` calls reside inside the branches created by the `if` statement. Each branch is a linear sequence of statements:

```
sleep(1);
printf("\nChild Process\n");
```

and

```
sleep(1);
printf("\nParent Process\n");
```

Hence, in both cases the value of  $CCN_{sp}$  for the `sleep` statements is 1, and equals to the  $CCN$  value of the branches in which the synchronization point was found.

## 4. Cost parameters for various synchronization types

Below we show examples of  $IP$  calculation for one of the synchronization types and  $CP$  calculations for data independent synchronization points.

### 4.1 IP analysis for Lock synchronization point

As described above, to use SCM it is necessary to calculate the  $IP$  values for the underlying operating environment, so that its scheduling policy is adequately taken into account. This is done only once; the calculated values can be then reused throughout SCM calculations

performed for applications running in that environment. As an example, consider the *Lock* synchronization type.

**Priority based scheduling:** If the lock is released at the time of the call, the current thread will acquire the lock and will continue running, as in try-lock. If the lock is locked at the time of the call, the current thread will be blocked (will become waiting), and the highest priority ready thread will be given the CPU. Here the interleaving potential is the number of threads in the application, because in the worst case, any other thread can theoretically be in ready state while the calling thread will be blocked waiting.

**Round-robin based scheduling:** For the considered functionality the analysis is the same, and so is the interleaving potential.

Evaluation of interleaving potentials for other types of synchronization points listed in Table 1 is done in a similar way; the results are summarized in Table 2 below.

**Table 2. Interleaving potentials for various synchronization types**

Synchronization Type	Interleaving potential	
	Priority based scheduling (standard system call)	Round Robin based scheduling (standard system call)
Try-Lock	1	
Unlock	Number of threads accessing the lock, with priority higher than that of the caller	1
Wait	Number of threads in the application	
Notify	Same as Unlock, but not higher than number of threads accessing "Wait".	
Pass Control (Explicit)	1	2
Volatile Access	Maximum between number of threads accessing the data and number of processors in the application.	
Task Initiation	1	2

### 4.2 CP Analysis for Data Dependent Synchronization Points

For all synchronization types relying on specific data (i.e. they are "data dependant"),  $CP$  equals to the number of threads using that data. This relates, for example to *Lock*, *Unlock*, *Try-Lock*, *Wait*, *Notify*, *Volatile Access* synchronization types where  $CP$  equals to the number of threads using the lock/semaphore/mutex data.

The worst case for the *CP* is the total number of threads; however it is safe to assume that typically this will not be the case. There are metrics (e.g. the coupling metric supported by static code analysis tools [7, 8]), that allow to identify such problematic synchronization point instances.

### 4.3 CP analysis for data independent synchronization points

Other synchronization types (such as *Pass Control (Yield)* and *Task/Thread Initiation*) are “data independent”. They depend on the number of total threads/tasks in the application, but not on the usage of a particular shared variable.

**Pass Control.** When the control is passed, whether explicitly or implicitly, the thread given the CPU can be any of the threads available in the system, thus the competition potential is the number of threads in the system.

**Task Initiation.** Task initiation is essentially a pass control construct, except that each task initiation increases the following call competition potential. Initiating all the threads in the system in the initialization stage is a common practice in embedded software implementations with limited resources. Such systems acquire all the needed resources in the initialization stage, including threads initiations and memory allocations, in order to avoid resource deficit during the life time of the running system. In this case the competition potential for each thread initiation synchronization point will increase by 1 every call, so that for each new thread  $t$ :  $CP_t = t$  (with  $CP_0 = 0$ ).

A common practice is to initiate threads in one loop, in a code that looks similar to the following example (here  $n$  is the total amount of tasks to be created):

```
int i, maxTasks = n;
for (i = 0; i < maxTasks; i++)
    createTask(i);
```

In this usage pattern, the *CP* of *createTask* changes at each iteration, but for the static analysis the average value can be considered (calculated as the sum of all the numbers in the range, divided by the number of iterations of execution of this code). So in this case, the *CP* for each call equals to

$$\frac{1}{n} \sum_{t=1}^n (t+1) = \frac{n+1}{2}, \text{ where } n \text{ is the total number of threads.}$$

In another case, when thread initiations and exits are performed in an arbitrary order (e.g. Web Server which starts a separate thread on each connection, and exits when the connection ends), the competition potential would be the **maximum number** of threads allowed to run in the system concurrently (i.e.  $n$ ).

## 5. Real life application analysis

For the sample implementation we used the open source tool CCCC [8]. Through static analysis of C, C++ or Java

code, it is able to calculate a number of metrics, including McCabe’s CCN that is used in the definition of SCM. The CCN values are calculated per function in a single pass over the code. They are then summarized to obtain the CCN value per module (thus allowing to assess the total effort needed for unit testing of all functions in the module).

For a sample real-life applications analysis we have chosen three HTTP server implementations. These are open source programs: two versions of the BusyBox HTTP server implementation (from BusyBox versions 1.0 and 1.14.1, both are available under GPLv2 license from the BusyBox project at <http://www.busybox.net/>), and IKI implementation (version 1.915) available at <http://www.iki.fi/iki/src/httpd.c>.

The reasons to choose this particular software are:

- These are stand-alone applications with many synchronization points
- The BusyBox implementation has been improved significantly between the versions, so it is a good candidate for comparative analysis
- The BusyBox HTTP server implementations are used in embedded devices where task synchronization problems are of high importance, while schedulers are of a simpler design (for example simple priority based schedulers can be found in many applications using this implementation).
- There are many other HTTP server implementations which can be used for comparison (we chose to compare the BusyBox and the IKI implementations).

Results of the analysis are described in Table 3. It can be clearly seen that the synchronization complexity of the IKI implementation is less than that of the BusyBox implementation, both absolutely and relatively to the total CCN of the module. This suggests possibility of more efficient and careful usage of system calls and inter-process communication mechanism in the IKI implementation, and can be used as a basis for review of the implementations in order to find better ways (or identify better patterns) of doing things.

**Table 3. Analysis of synchronization complexity for various HTTP server implementations**

Module	CCN	SCM	SCM addition
Old BusyBox	374	1416	278%
New BusyBox	326	615	87%
IKI	312	487	56%

### 5.1 Growth of SCM values

Additional interesting observation seen in Table 3 is that SCM doesn’t grow exponentially, as might be expected based on the metric formula. This is due to very low de-facto competition potentials for each of the synchronization

points. The analyzed implementations used only two tasks: the main task and a listener task forked from it. Hence the *CP* value was equal to 2.

In real life applications the tendency is towards loose coupling between modules, thus creating very little dependencies and synchronization amongst threads in the system. For example, in [6] the requirement is that "Source code should be developed as a set of modules as loosely coupled as reasonably feasible".

Therefore, as we already mentioned, it can be assumed that most of the real-life applications have only handful of threads competing over the same resources, and thus require synchronization. For example, in [3], 16 out of 575 classes (i.e. 3%) having synchronization primitives is considered reasonable. Although it is not clear from [3] how many threads are in the system, it is reasonable to assume that in such large-scale system, the amount of threads sharing data is small relatively to the total number of the threads. In [11] it is also shown that there are very limited amounts of threads for even large scale projects.

## 6. Conclusions and future work

We defined a metric that provides valuable information to the developers and testers when considering different implementations of the same functionality, or usage of certain implementation in different execution environments.

SCM can be used to estimate the amount of unique execution paths required to cover the expected interleavings on the program language level. Thus, this number also represents the amount of unique tests required for proper coverage of these paths (based on the known branching and synchronization coverage models). This is important, for example, when trying to reach full coverage based on the concurrency coverage criteria with tools like ConTest [3] or CHESS [11]. Part of the future work is to show a correlation between the SCM and the effort required for testing using tools like CHESS or ConTest. Such correlation will help validating the SCM and verify that it can be used for estimation of the effort needed to achieve full coverage when using these tools.

Although there are possible cases where SCM will provide extremely large values which by themselves will appear not to be useful, in fact such values can still be useful and suggest that there are coupling issues in the code under analysis which should be checked. Even then, comparative analysis using SCM to choose the better option out of several "bad" (with high SCM values) options can be performed.

Thus the conclusion is that SCM is a measure which provides usable results for real life applications, and apart from it direct uses as test effort estimation and comparative analysis, it also provides an indication for the adequacy of

the coupling between the threads in the system.

Our opinion is that having a metric which allows comparing the impact of the synchronization on the overall program complexity between various possible solutions may be beneficial for software developers and testing engineers. As part of future work we plan to evaluate the effectiveness of SCM and its ability to prove itself useful for development and testing of real-world applications.

Also, further development needs to be done on the implementation of SCM in order to provide values precise enough to be used in pair with coverage calculations based on one of the models described in [3, 9, 11].

## 7. References

- [1]. B. M. Hetzel, The Complete Guide to Software Testing, 2nd ed. (John Wiley & Sons, 1993).
- [2]. M. Bilberstein, E. Farchi, S. Ur, Choosing among alternative pasts, Concurrency and Computation: Practice and Experience, Vol.19/3, John Wiley & Sons Ltd, pp.341-353, 2006.
- [3]. A. Born, E. Farchi, Y. Magid, Y. Nir, S. Ur, Applications of synchronization coverage, Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming, ACM New York, USA, pp.206-212, 2005.
- [4]. O. Edelstein, E. Farchi, Y. Nir, G. Ratzaby, S. Ur, Multithreaded Java Program Test Generation, IBM Systems Journal, 41, 3 (2002), pp. 111-125.
- [5]. "Guidelines for the use of the C language in vehicle based software" (MISRA-C:1998), Motor Industry Software Reliability Association, <http://www.misra-c2.com/index.htm>, 1998.
- [6]. Joint Strike Fighter Air Vehicle C++ Coding Standards, Lockheed-Martin Corporation, Document Number 2RDU00001, Rev. C, 12/2005.
- [7]. T. Littlefair, An Investigation into the use of Software Code Metrics in the Industrial Software Development Environment, PhD thesis, Faculty of Communications, Health, and Science, Edith Cowan University, Mount Lawley Campus, June 2001.
- [8]. T. Littlefair, C and C++ Code Counter, <http://cccc.sourceforge.net>, 2003.
- [9]. S. Lu, W. Jiang, Y. Zhou, A Study of Interleaving Coverage Criteria, POSTER SESSION: ESEC/FSE'07 posters, 2007, pp.: 533-536.
- [10]. T. McCabe, A Complexity Measure, IEEE Transactions on Software Engineering, SE-2(4), 1976, pp.308-320.
- [11]. M. Musuvathi, S. Qadeer, and T. Ball. CHESS: A Systematic Testing Tool for Concurrent Software. Microsoft Research Technical Report MSR-TR-2007-149, 2007.
- [12]. H. Zuse, "A Framework of Software Measurement", Walter de Gruyter, Berlin, New York, 1998.

# Measurement Model of Software Requirements Derived from System Maintainability Requirements

Alain Abran<sup>1</sup>, Khalid T. Al-Sarayreh<sup>2</sup>, Juan J. Cuadrado-Gallego<sup>3</sup>  
<sup>1,2</sup>Software Engineering Department, University of Quebec (ETS)  
1100 Notre-Dame west, Montréal, Québec H3C 1K3, Canada

<sup>3</sup>Departamento de Ciencias de la Computación, Universidad de Alcalá  
28805 Alcalá de Henares, Madrid, Spain

<sup>1</sup>alain.abran@etsmtl.ca, <sup>2</sup>khalid.al-sarayreh.1@ens.etsmtl.ca, <sup>3</sup>jjcg@uah.es

**Abstract**—Maintainability is typically described initially as a non functional requirement at the system level. Systems engineers must subsequently apportion these system requirements very carefully as either software or hardware requirements to conform to the maintainability requirements of the system. A number of concepts are provided in the ECSS, ISO 9126, and IEEE standards to describe the various types of candidate maintainability requirements at the system, software, and hardware levels. This paper organizes these concepts into a generic standards-based reference model of the requirements at the software level for system maintainability. The structure of this reference model is based on the generic model of software requirements proposed in the COSMIC – ISO 19761 model, thereby allowing the measurement of the functional size of such maintainability requirements implemented through software.

**Keywords**—Maintainability Requirements, Non functional requirements – NFR, Functional size, COSMIC – ISO 19761, ECSS International Standards, Software Maintainability Measurement.

## 1. Introduction

Non-functional requirements (NFR) play a critical role in system development, including as selection criteria for choosing among alternative designs and ultimate implementations. NFR may also have a considerable impact on project effort, and should be taken into account for estimation purposes and when comparing project productivity.

Typically, these NFR are described at the system level and not at the software level, and there is no consensus yet on how to describe and measure these system NFR. In practice, NFR can be viewed, defined, interpreted, and evaluated differently by different people, particularly when they are stated vaguely and only briefly [1-3]. Therefore, it is challenging to take them into account in software estimation and software benchmarking: NFR have received less attention in the software engineering literature and are definitely less well understood than other cost factors [3]. Without measurement, it is challenging to take them as quantitative inputs into an estimation process and productivity benchmarking.

In practice, the requirements are initially typically addressed at the system level [4-10] either as high-level system functional user requirements (system FUR) or as high-level system non-functional

requirements (system NFR). The latter must usually be detailed, allocated and implemented in either hardware or software, or both, as software FUR, for instance.

For example, a system FUR will describe the required functions in a system, while a system NFR will describe how the required functions must behave in a system. In the software requirements engineering step, system NFR can then be detailed and specified as software FUR to allow a software engineer to develop, test, and configure the final deliverables to system users.

The term "functional" refers to the set of functions the system (including the software) has to offer, while the term "non-functional" refers to the manner in which such functions are performed. FUR is typically phrased with subject or predicate constructions (i.e. noun/verb), such as: "The system must print 5 reports". NFR, by contrast, are typically phrased with adverbs or modifying clauses, such as: "The system will print 5 reports quickly" or "The system will print 5 reports with a high degree of reliability".

In the ECSS (European Cooperation on Space Standardization) standards for the aerospace industry [11-14], the ISO 9126 [15] and IEEE 830 [16] standards, a number of concepts are provided to describe various types of candidate maintainability requirements at the system, software, and hardware levels. However, these standards vary in their views, terminology, and coverage of maintainability.

Currently, there exists no generic model for the identification and specification of software FUR for implementing system maintainability requirements (system NFR) based on the various views documented in international standards and in the literature [1-14]. Consequently, it is challenging to measure these maintainability-related software FUR, and take them into account quantitatively for estimation purposes.

This paper focuses on a single type of NFR, that is, system maintainability requirements, and reports on the work carried out to define an integrated view of software FUR for system maintainability NFR, on the basis of international standards including the use of the generic COSMIC – ISO 19761 [17] model of software FUR.

The paper is organized as follows. Section 2 presents the view of software FUR in ISO 19761. Section 3 identifies the standards describing

maintainability requirements. Section 4 presents a standards-based definition of a generic model of requirements for software to implement system maintainability NFR. Finally, a discussion is presented in section 5.

## 2. A Generic view of software FUR in ISO

ISO 14143-1 [18] specifies that a functional size measurement (FSM) method must measure the software functional user requirements (FUR). In addition, the COSMIC – ISO 19761 [17] model proposes a generic model of software FUR that clarifies the boundary between hardware and software. Fig. 1 illustrates the generic flow of data from hardware to software from a functional perspective. From this generic model of software FUR, depicted in Fig. 1, the following observations can be made:

- Software is bounded by hardware. In the so-called “front end” direction (i.e. center of Fig. 1), software used by a human is bounded by I/O hardware such as a mouse, a keyboard, a printer, or a display, or by engineered devices such as sensors or relays. In the so-called “back end” direction (i.e. the right-hand side of Fig. 1), software is bounded by persistent storage hardware like a hard disk, or RAM or ROM memory.
- Software functionality is embedded within the functional flows of data groups. Such data flows can be characterized by four distinct types of data movements. In the “front end” direction, two types of movements (ENTRIES and EXITS) allow the exchange of data with users across a boundary. In the “back end” direction, two types of movements (READS and WRITES) allow the exchange of data with the persistent storage hardware.

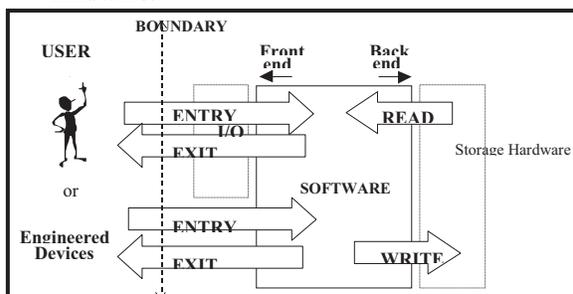


Fig. 1: Generic flow of data groups through software from a functional perspective in COSMIC – ISO 19761

- Different abstractions are typically used for different measurement purposes. In real-time software, the users are typically the engineered devices that interact directly with the software, that is, the users are considered the I/O hardware. For business application software, the abstraction commonly assumes that the user is one or more humans who interact directly with the business application software across the boundary; the I/O hardware is ignored.

As an FSM method, COSMIC is aimed at measuring the size of software based on identifiable FUR. Once identified, those requirements are allocated to hardware and software from the unifying perspective of a system integrating these two “components”. Since COSMIC is aimed at sizing software, only those requirements allocated to the software are considered in its measurement procedure.

## 3. Identification of standards describing maintainability requirements

This section presents a survey of the maintainability-related views, concepts, and terms in the ECSS, ISO 9126, and IEEE-830 standards. This section identifies which standards currently address aspects of the software FUR derived from system maintainability FUR and NFR – see Fig. 2. The expected outcome is the identification of the various elements that should be included in the design of a standards-based framework for modelling software FUR for system maintainability. The elements of maintainability are dispersed in various system views throughout various ECSS standards and are expressed as either:

- System maintainability functional user requirements (system maintainability FUR), or
- System maintainability non-functional requirements (system maintainability NFR)

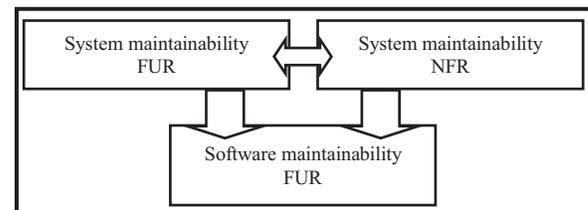


Fig. 2: Mapping system NFR into software FUR for maintainability

### 3.1 ECSS: views and concepts for maintainability

Maintainability in the ECSS standards is considered as part of the integrated support requirements in system engineering, including related activities and procedures. Table 1 presents a list of concepts and vocabulary used in the ECSS standards to describe system-related maintainability requirements. For instance, the ECSS specifies that, for system maintainability, analysis of failure modes, effects, and criticality (FMECA) must be carried out. ECSS does not specify, however, if such requirements must be implemented in software, hardware, or a combination of the two.

While conducting the survey of all the maintainability concepts and terms described in the ECSS-E-40 and ECSS-Q series and in ECSS-ESA as the integrated standard for ECSS-E and ECSS-Q, it was observed that:

- These various maintainability elements are described differently, and at different levels of detail;

- The maintainability elements are dispersed throughout the various documents: there is, therefore, no integrated view of all types of candidate maintainability requirements;
- There is no obvious link between the maintainability requirements in ECSS-ESA [19] as the integrated standard and all the other ECSS standards that describe maintainability requirements.

Table 1: Maintainability view and vocabulary in ECSS

Key view	Concepts and Vocabulary
Part of the integrated support requirements in system engineering, including activities and procedures	<ul style="list-style-type: none"> <li>• Maintainability activities and procedure.</li> <li>• Maintainability operations</li> <li>• Environment control and life support systems design (ECLSS)</li> <li>• Failure modes, effects, and criticality analysis (FMECA)</li> <li>• Failure modes and effects analysis (FMEA)</li> <li>• Mean-time-to-repair and System down-time</li> <li>• Fault detection and isolation capability</li> <li>• System malfunction.</li> </ul>

It is also to be noted that the ECSS does not propose a way to measure such software maintainability requirements, and, without measurement, it is challenging to take such an NFR either as a quantitative input to an estimation process or in productivity benchmarking.

### 3.2 IEEE: views and concepts for maintainability

IEEE-830 [16] lists maintainability as one of the NFR type, but does not define it, nor does it provide guidance on how to describe and specify the maintainability requirements; of course, it does not provide guidance on how to measure any of these NFR either.

IEEE-14764 [20] and IEEE-982.1 [21] only define the maintainability requirement as the capability of the software product to be modified, without mentioning how to describe and specify the maintainability requirements.

### 3.3 ISO9126: views and concepts for maintainability

The key view on maintainability in the ISO 9126 series is from the perspective of the quality of the software product: maintainability is presented as a ‘quality characteristic’, which is decomposed into quality sub characteristics and then into proposed derived measures to quantify those quality sub characteristics. The inventory of related concepts and vocabulary on software maintainability, such as analyzability, changeability, is presented in Table 2.

Table 2: Maintainability view & vocabulary in ISO 9126

Key views	Concepts and Vocabulary
Maintainability quality characteristic: The capability of the software product to be modified.	<ul style="list-style-type: none"> <li>• Analysability</li> <li>• Audit Trial Capability</li> <li>• Failure Analysis Capability</li> <li>• Status Monitoring Capability</li> <li>• Diagnostic Function Support</li> <li>• Changeability</li> <li>• Change Efficiency</li> <li>• Software Change Control Capability</li> </ul>
Modifications may include corrections, improvements, or adaptation of the software to changes in environment	<ul style="list-style-type: none"> <li>• Modifiability</li> <li>• Stability</li> <li>• Modification Impact</li> <li>• Change Success Ratio</li> <li>• Testability</li> <li>• Availability of Built-in Test Function</li> <li>• Retest Efficiency</li> <li>• Test Restart Ability</li> </ul>

A large number of measures are proposed in ISO 9126, but none addresses software FUR, only the maintainability NFR of the software itself. However, nothing precludes the use of these concepts at the system level or looking at what functions must be performed at the software level (i.e. FUR allocation to software) to implement these system level NFR.

## 4. A standards-based definition of a generic model of software-FUR for system maintainability requirements

This section identifies and assembles the terminologies and concepts of maintainability dispersed throughout the ECSS, IEEE, and ISO standards. These terminologies are mapped next into a proposed model of maintainability software FUR using the generic FUR model proposed in COSMIC. This COSMIC-based generic model then becomes a framework for describing the software FUR from system maintainability requirements based on the ECSS standards.

### 4.1 Mapping maintainability views and vocabulary from standards

Table 3 presents the system maintainability requirements that are present either as system requirements in the ECSS standard or as maintainability-related concepts in ISO 9126: each of these could be interpreted, and specified, at times as software FUR.

### 4.2 Types of maintainability requirements

Next, four types of system-related maintainability requirements can be derived:

- System Analyzability
- System Changeability
- System Stability
- System Testability

Table 3: Maintainability in ECSS & ISO 9126

System Maintainability Requirements
Failure Data Operation Failure Data Monitoring Failure Data Control System Failure Tasks Failure Isolation Failure Detection Correct Data Faults Correct System Defects Fault Prevention of Data Control Fault Prevention of System Functions Fault Allocation Time

Table 4 presents various typical procedures (middle column) for system maintainability requirements and corresponding software functions (right-hand side column) that may be specified to implement such procedures for the four types of system maintainability requirements.

Table 4: System maintainability requirements and related software functions

System Maintainability Requirements	Software Procedure for System Maintainability	Software functions
System Analysability	Maintainability Procedure	System Diagnostic Function Failure Data Operation Failure Data Monitoring Failure Data Control System Failure Tasks
System Analysability & Changeability	Registered Failures	Failure Isolation Failure Detection
System Changeability	System Malfunction	Correct Data Faults Correct System Defects
System Testability		System Time Fault Allocation Time
System Stability		Fault Prevention of Data Control Fault Prevention of System Function

### 4.3 Software maintainability functions to be specified

The maintainability functions to be specified (and corresponding entities to be measured) are divided into external and internal maintainability function that may be allocated to software – see Table 5: External maintainability refers to the expected failures that could occur in the system, while internal maintainability refers to the expected correction of the failures occurring in the system.

Table 5: Maintainability functions allocated to software

Type	Maintainability functions
<b>External Maintainability</b>	<ul style="list-style-type: none"> <li>Failure Data Operation function</li> <li>Failure Data Monitoring function</li> <li>Failure Data Control function</li> <li>System Failure Tasks function</li> <li>Failure Isolation function</li> <li>Failure Detection function</li> </ul>
<b>Internal Maintainability</b>	<ul style="list-style-type: none"> <li>Correct Data Faults function</li> <li>Correct System Defects function</li> <li>Fault Prevention of Data Control function</li> <li>Fault Prevention of System Functions</li> <li>System Time function</li> <li>Fault Allocation Time function</li> </ul>

### 4.4 Identification of the maintainability function types

This section identifies the function types and functional relationships in the software FUR for system maintainability.

#### Maintainability Function Type 1: Failure Procedure - Fig. 3

- The system diagnostic function sends a data group to the failure data operation, monitoring, control, and system failure tasks.
- Failure data operation, monitoring, control, and system failure tasks functions send a data group to each other.

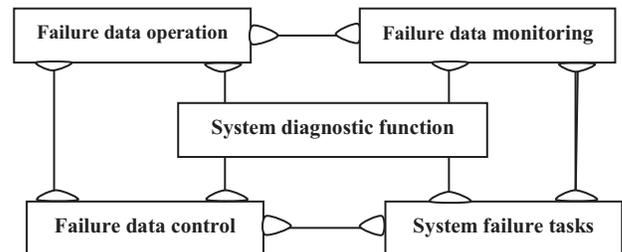


Fig. 3: Maintainability failure procedure

#### Maintainability Function Type 2: Registered failures – Fig. 4

- Failure detection function sends/receives a data group to/from failure isolation function and vice versa.

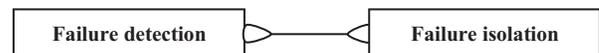


Figure 4: Registered failures

#### Maintainability Function Type 3: System malfunction – Fig. 5

- Correct data faults function sends/receives a data group to/from correct system defects function, and vice versa.

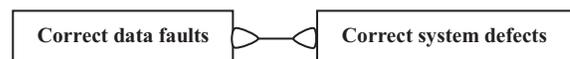


Figure 5: System malfunction

**Maintainability Function Type 4: System stability**

– Fig. 6

- Fault prevention of data control function sends/receives a data group to/from fault prevention of system functions, and vice versa.

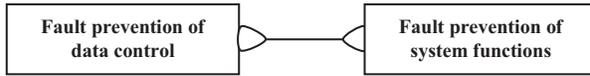


Fig.6: System stability

**Maintainability Function Type 5: System testability** – Fig. 7

- The system time function sends/receives a data group to/from fault allocation time, and vice versa.
- The system time and fault allocation time function sends/receives a data group to/from system stability function, and vice versa.

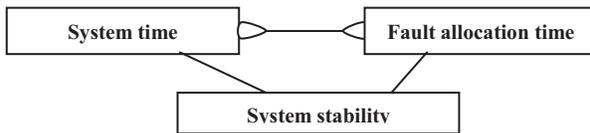


Fig.7: System testability

**4.5 Identification of the functional relationships**

Fig. 8 presents an overview of the relationships between the function types in the maintainability software FUR, using COSMIC for graphical representation. Specifically:

- The sub model of the Maintainability Function Type 1 can be used to specify and measure the

functional size of the external maintainability for the maintainability failure procedure from the received/sent data movements from/to registered failures and system stability function – see Fig. 8.

- The sub model of the Maintainability Function Type 2 can be used to specify and measure the functional size of the external maintainability for the registered failure from the received/sent data movements from/to registered failures and system malfunctions and stability functions – see Fig. 8.
- The sub model of the Maintainability Function Type 3 can be used to specify and measure the functional size of the internal maintainability for the system malfunction Maintainability Function type from the received/sent data movements from/to registered failures and system stability functions – see Fig. 8.
- The sub model of the Maintainability Function Type 4 can be used to specify and measure the functional size of the internal maintainability for the system stability Maintainability Function type from the received/sent data movements from/to malfunction system, system testability and system failure procedure – see Fig. 8.
- The sub model of the Maintainability Function Type 5 can be used to specify and measure the functional size of the internal maintainability for the system testability entity type from the received/sent data movement from/to stability functions – see Fig. 8.

This model is referred to here as a generic model of software FUR for system maintainability.

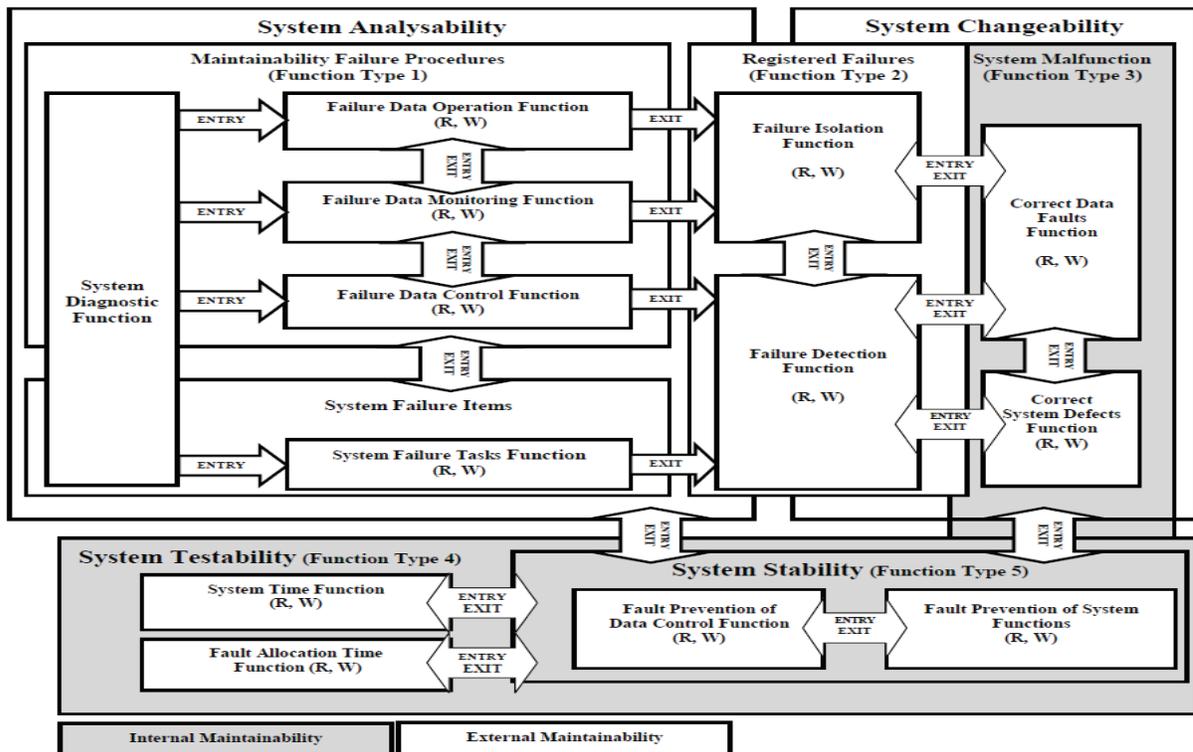


Fig. 8: COSMIC reference model of maintainability requirements allocated to software

## 5 Discussion

This paper has introduced a procedure for specifying and measuring software requirements for the internal and external maintainability needed to address the system's maintainability requirements.

The main contribution of this paper is our proposed Generic Model of software FUR for system maintainability. This generic model can be considered as a kind of reference model for the identification of system maintainability requirements and their allocation to software functions implementing such requirements. System requirements allocated to hardware have not been addressed in this paper. Since the structure of the generic model is based on the generic model of software adopted by the COSMIC measurement standard, the necessary information for measuring their functional size is readily available.

Specifically, the generic model of maintainability presented in this paper is based on:

- the ECSS standards for the description of the NFR for system maintainability; and
- The COSMIC measurement model of functional requirements.

The model is independent of the software type and the languages in which the software FUR will be implemented. The proposed generic maintainability model (i.e. reference model) provides:

- A specification model for each type, or all types, of maintainability requirements: for example, the requirements to be allocated to software for the maintainability failure procedures for system analyzability, the registered failures and software/system malfunction for system changeability, and for system/software stability and testability.
- A specification measurement model for each type, or all types, of maintainability requirements.

Future work includes documentation of the traceability of the elements of this generic model to the detailed elements of the ECSS standards, as well verification of this generic model to ensure full coverage of maintainability requirements. Discussions with groups of experts will be necessary to ensure its usefulness across various communities and to develop a consensus on further refinements of such a generic model which could be proposed eventually as a candidate for standardization.

## REFERENCES

- [1] L. Chung and J. Cesar Prado Leite, "On Non-Functional Requirements in Software Engineering", in "Conceptual Modeling: Foundation and Applications, Essays in Honor of John Mylopoulos", Springer, 2009.
- [2] L. Chung, B. Nixon, E. Yu, J. Mylopoulos, "Non-Functional Requirements in Software Engineering", Springer, Heidelberg, 1999.
- [3] J. Mylopoulos, L. Chung, B. Nixon, "Representing and Using Nonfunctional Requirements: A Process-Oriented Approach", IEEE Transactions on Software Engineering vol. 18, pp. 483-497, 1992.
- [4] M. Shaw, "Larger Scale Systems Require Higher-Level Abstractions", Software Specification and Design, IEEE Computer Society, vol. 14, pp. 143-146, 1989.
- [5] A. M. Davis, "Software requirements: objects, functions, and states", Prentice-Hall, 1993.
- [6] I. Jacobson, G. Booth, J. Rumbaugh, "Excerpt from the Unified Software Development Process: The Unified Process", IEEE Software, vol. 16, pp. 96-102, 1999.
- [7] K. Wiegers, "Software Requirements", 2nd edition, Microsoft Press, 2003.
- [8] G. Roman, "A Taxonomy of Current Issues in Requirements Engineering", IEEE Computer, pp. 14-21, 1985.
- [9] B. W. Boehm, "Characteristics of software quality", North-Holland Pub. Co., American Elsevier, 1978.
- [10] A. I. Antón, "Goal identification and refinement in the specification of software-based information systems", PhD Thesis, Georgia Institute of Technology, 1997.
- [11] ECSS-E-40-Part-1B, "Space Engineering: Software - Part 1 Principles and Requirements", European Cooperation for Space Standardization, The Netherlands, 2003.
- [12] ECSS-E-40-Part-2B, "Space Engineering: Software-part 2 Document Requirements Definitions", European Cooperation for Space Standardization, The Netherlands, 2005.
- [13] ECSS-Q-80B, "Space product assurance: Software product assurance", European Cooperation for Space Standardization, The Netherlands, 2003.
- [14] ECSS-E-ST-10C, "Space engineering: System engineering general requirements", Requirements & Standards Division, Noordwijk, The Netherlands, 2009.
- [15] ISO/IEC-9126, "Software Engineering - Product Quality - Part 1: Quality Model", International Organization for Standardization, Geneva (Switzerland), 2004.
- [16] IEEE-Std-830, "IEEE Recommended Practices for Software Requirements Specifications", IEEE, 1993.
- [17] ISO/IEC-19761, "Software Engineering - COSMIC v 3.0 - A Functional Size Measurement Method", International Organization for Standardization, Geneva (Switzerland), 2003.
- [18] ISO/IEC-14143-1, "Information technology-Software measurement - Functional size measurement Part 1: Definition of concepts", International Organization for Standardization, Geneva (Switzerland), 1998.
- [19] ECSS-ESA, "Tailoring of ECSS, Software Engineering Standards for Ground Segments, Part C: Document Templates", ESA Board of Standardization and Control (BSSC), 2005.
- [20] ISO/IEC-14764, "Standard for Software Engineering—Software Life Cycle Processes—Maintenance", ISO, Geneva (Switzerland), 2006.
- [21] IEEE-982.1, "IEEE Standard Dictionary of Measures of the Software Aspects of Dependability", Software Engineering Standards Committee, IEEE Computer Society, New York, USA, 2005.

# Evolution Styles to Capitalize Evolution Expertise within Software Architectures

Olivier Le Goer, Dalila Tamzalit and Mourad Oussalah  
Université de Nantes, LINA - CNRS UMR 6241  
2, rue de la Houssinière, Nantes cedex 03, France  
{olivier.le-goer, dalila.tamzalit, mourad.oussalah}@univ-nantes.fr

## Abstract

*Evolution is an increasingly important aspect in the software architecture community. At this level of abstraction, evolution is mainly concerned with changes brought to specifications of components and connectors and to their configuration. Such an evolution activity may be very hard or easy; the degree of difficulty largely relying on the competence and skill of the architect who performs it. In order to decrease time and cost associated to evolution, this paper suggests a new evolution model relying on styles to capitalise recurring evolutions in order to re-apply them whenever a similar problem arises again within architectures. Ultimately, this paper argues in favor of off-the-shelf evolutions to tackle the complex problem of architectural evolution.*

## 1. Introduction

Most work on software evolution focuses on the code level. However, software architecture permits system evolution at a level of abstraction where quality and business tradeoffs can be understood and analyzed. We consider that “Evolving-in-the-large” is an approach for software evolution that shifts the user’s focus away from lines-of-code to coarse-grained components and their overall interconnection structure.

Architects have almost no assistance in evolving their architecture. Evolution is typically done in an ad hoc manner, guided only by the competence of the architect performing it. We believe that evolution is generalizable and repeatable, i.e., not specific to a particular instance. Thereby, we argue that it should be possible to define kinds of evolution patterns or styles, reusable over a family of related applications.

Taking a step back, we observe that in the field of software architecture, the problem of complexity underlying

the architectural design was successfully tackled in the past through styles and patterns. Indeed, there was a large body of work on the study and the classification of such reusable assets dedicated to architectural designs [12, 13, 3]. In the meantime, these efforts contributed to provide a common vocabulary to the community. Arguably the most important to us, there was an attempt to build a repository for storing and retrieving architectural styles in order to provide guidance to software architects [10]. From there on, the question is straightforward: why not leverage these key ideas to address the problem of architectural evolution?

In this paper, we present an evolution model in which a recurring evolution is modeled as a first-class entity called an evolution style. This evolution model considers that the competences of the software architects should be durably capitalized into a repository, termed a “shelf”, for their subsequent (re)use. Ultimately, reuse becomes effective when selecting a style from a shelf and instantiating it on a particular architecture. We propose that an evolution style (or style, for short) has a two part specification, called *header* and *competence*. Furthermore, evolution styles are related to one another with several semantic relationships to improve their organization on the shelf. Specifically, their organisation provides a natural taxonomy that is a foundation for a classification-driven reasoning. The type of the reasoning performed, either static or dynamic, determines the value-added features of our evolution model.

The rest of this paper is organized as follows. First, in Section 2 we briefly present the background information that sets the stage for the ensuing discussion. We survey the related work in Section 3. Section 4 introduces our style-based evolution model and Section 5 explains how architectural evolution is achieved according to this model. The off-the-shelf approach derived from our evolution model is described in Section 6, where we give the key elements to build a shelf for evolution styles. We conclude this paper in Section 7.

## 2. Background on Software Architecture

The evolution model developed in this paper relies on concepts from the software architecture field. This section briefly discusses these concepts.

As software systems grew more complex, their design and specification in terms of coarse-grain building blocks became a necessity. The field of software architecture addresses this issue and provides high-level abstractions for representing the structure, behavior, and key properties of a software system. Software architectures involve descriptions of the elements from which systems are built, interactions among those elements, patterns that guide their composition, and constraints on these patterns [11]. In general, a particular system is defined in terms of a collection of *components*, their interconnections (*configuration*), and interactions among them (*connectors*).

To date, many architecture description languages (ADLs) have been developed to aid architecture-based development [9]. ADLs provide formal notations for describing and analyzing software systems. They are usually accompanied by various tools for parsing, analysis, simulation, and code generation of the modeled systems. Examples of ADLs include C2SADEL [8], Darwin [7], and Wright [1]. A number of these ADLs also provide extensive support for modeling behaviors and constraints on the properties of components and connectors [9]. In contrast, ADLs provide a limited support for architecture-based evolution.

## 3. Related Work

Traditionally, it is admitted that Lehman's laws [6] laid the foundations of software evolution. This seminal work was based on empirical observations that have shown that changes in a software system are inevitable. From there on, there was the need for new engineering activities dedicated to software evolution. These efforts led to important evolution techniques such as: code refactoring, code slicing, and model transformation. Nevertheless, architecture evolution, by itself, has received little attention.

We review two approaches: architecture description languages (ADLs) and architectural styles. By providing formal syntax and semantics, ADLs facilitate analysis, which provide input for engineered evolution. However, existing ADLs do not provide any techniques to specify architectural change in the context of evolution [2]. An architecture style defines a vocabulary of components and connector types, and a set of constraints on how they can be combined. Architecture styles provide a way to capture knowledge about common classes of systems, and a leverage for analyzing their properties. Recently, several researchers (including us [5]) have proposed to extend styles to architecture evolution

and defined the concept of evolution styles. Particularly, Garlan et al. [4] defined an evolution style as a set of evolution paths among classes of systems, e.g., evolutions from a web-based architecture to J2EE. This point of view requires an even greater abstraction level to be able to describe such paths. From our concern, an evolution style is rather a local evolution path related to any architectural element. In short term, our respective visions could converge towards a single solution able to deal uniformly with different abstraction levels. For the time being, it's our own vision which is detailed in the rest of this paper.

## 4. Modeling Architecture Evolution

We argue that evolution of architectural elements should be captured in an abstraction amenable to being reused. In this section, we present our vision for such a model for evolution. We assume that the evolution-oriented task of architects is a combination of the knowledge of the elements involved and the conditions for their change, along with the method used to perform the change. To separate between these two views of an evolution, we suggest that an evolution style has two parts: a *header* and a *competence*. In addition, our evolution model is enhanced with a number of relational elements.

### 4.1. Header and Competence

As mentioned above, an evolution style is defined with a header plus a competence. In other words, a header provides an interface, and the competence provides an implementation of that interface. As a direct consequence of that, the header is mandatory while the competence is not. An abstract style is a style that does not define any competence to realize the header. On one hand, the header declares a textual description of the goal, a context, a list of parameters and assertions. Context is provided by the kind of architectural element where the style can be used (component, connector, configuration, port, etc.). The range of possibilities directly depends on the architectural elements reified by a given ADL. Parameters are further elements that may be required during the process. Finally, assertions are typically a pre- and a post-condition used to control the state of architecture before and after evolution. On the other hand, the competence specifies logic and control (i.e., the algorithm) to achieve the goal stated by the header.

One concretisation of this abstraction is shown in Figure 1. The goal of the evolution style depicted is to manage properly the suppression of a component located on the border of a configuration, that is, maintaining bindings<sup>1</sup> with it. We see that the header declares the context (a component)

<sup>1</sup>also known as delegation connectors

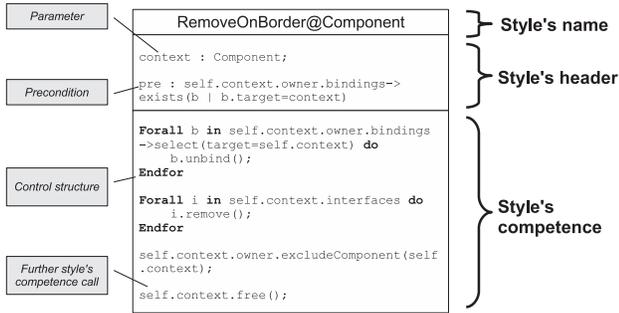


Figure 1. Specimen of evolution style.

and the initial state of the architecture (there exist bindings between the components and its owning configuration). The competence specifies what must be done in this case and can involve further evolution styles. Indeed, evolution styles know each other due to some relational elements introduced in the next section.

#### 4.2. Relational Elements

A key aspect of our evolution model is to provide some relational elements (in the broad sense) in order to improve the practical usage of evolution styles. Instantiation, specialization, composition and utilization are important relationships that we describe below:

1. Instantiation (*is-a*): makes possible to reuse a style on a given architecture. Logically, an abstract style cannot have direct instances. Each instantiation produces a particular evolution process (or simply evolution, for short).
2. Specialization (*is-kind-of*): enables to define more specific styles by inheriting from a parent. As a result, it is possible to build a conceptual hierarchy of evolution styles.
3. Composition (*is-composed-of*): is an important technique to combine styles into more complex ones. A composite style delegates its competence to its sub-styles. Above all, composition suggests a strong coupling between the evolutions modeled.
4. Utilization (*uses*): allows styles to collaborate in an optional manner. Hence, this relational element suggests a loose coupling between the evolutions modeled.

The Figure 2 gives an illustration of the relational elements explained above. Due to the object-oriented spirit of our relational elements we used the graphic notation of UML to depict them.

Basically, each of these four relational elements underpins a specific operational mechanism. Instantiation (1) is

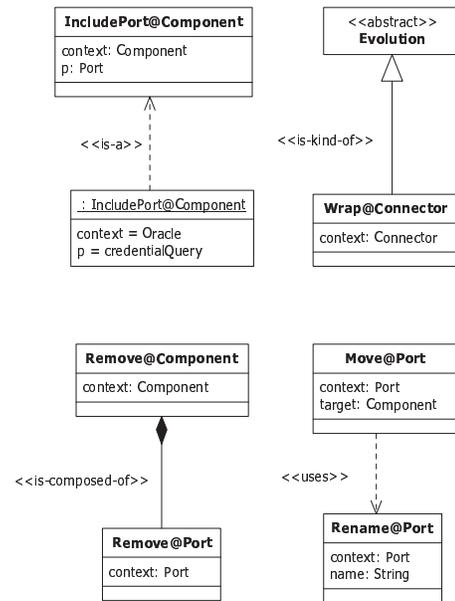


Figure 2. Illustration of relational elements supported by evolution styles.

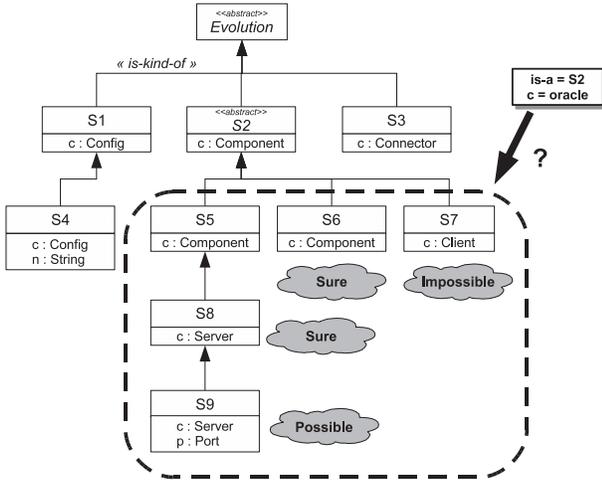
concerned with binding effective parameters (i.e., elements of an architecture) with formal parameters and with checking assertions. In the case where the evaluation of the precondition succeeds and the style is not an abstract style, the execution of the competence is triggered. Specialization (2) is concerned with an inheritance principle that supports *header overloading* and *competence overriding*. The overloading technique stipulates that the header of a sub-style may add new or more specific parameters, weaken the precondition and strengthen the postcondition. The overriding technique stipulates that the competence of a substyle can replace the competence of its superstyle. Composition (3) and utilization (4) are both concerned with a delegation principle by which the competence of an evolution style can be called from another evolution style.

### 5. Achieving Architecture Evolution

This section deals with the runtime aspect of the evolution process, whenever an evolution style is instantiated. The architect selects an element of her architecture and a style to apply to it, and (ideally) properly fulfills the header's parameters. If the evaluation of the header's precondition succeeds, then the style is elected. From there, two cases are possible: either the style is concrete and its competence is executed, or it is abstract and a dynamic competence lookup is started. Let us have a closer look at this mechanism that supports *competence polymorphism*.

## 5.1. Dynamic Competence Lookup

Dynamic competence lookup is an inference mechanism that relies on a dynamic classification-driven reasoning. Its purpose is to discover what competence matches with a given state of the architecture. The lookup process is based on an in-depth traversal of the specialization hierarchy in order to find the most specific style to which the instance could be attached via the *is-a* relationship.



**Figure 3. Illustration of dynamic competence lookup mechanism.**

The dynamic classification-driven reasoning is driven by the values of the effective parameters provided by the architect. In practice, it is frequent that the instance to be classified is incomplete because some parameters are missing. To deal with this issue, we use an “uncertain” classification technique. According to this technique, the attachment of an instance to a concrete evolution style can be labeled:

- **Sure:** if the instance satisfies all the constraints (i.e., parameters’ types and precondition) of the style.
- **Possible:** if the instance does not violate any constraint but is incomplete.
- **Impossible:** if the instance violates at least one constraint.

The exemple given by Figure 3 sketches a situation where an instance of the style *S2* is demanded and where the formal parameter *c* (abbreviation of *context* due to space limitation) is bound with the server component “Oracle”. Consequently of the abstract nature of *S2*, the dynamic lookup mechanism then tends to determine which sub-style of *S2* would be relevant to this instance. As a result, *S5*, *S8* and *S6* are labeled “sure”. However, *S8* is more suitable

than *S5* (i.e., most specific<sup>2</sup>). Meanwhile, *S7* is labeled “impossible” due to a type incompatibility. The “possible” evolution style is *S9* because the single effective parameter “Oracle” does not suffice in classifying the instance into it.

## 5.2. Validation of the Evolution

As already mentioned, the execution of a style’s competence can trigger the execution of other styles, and so on, until there is no more competence to execute on the architecture. When dynamic competence lookup occurs and when several styles were labeled “sure” or “possible”, the architect must intervene to make a decision. Her duty concerns the choice among the list of evolution styles so that the evolution process could continue, and the provision of missing parameters needed by styles labeled possible.

At the end of the evolution process, the consistency of the architecture is checked. It is worth mentioning that the constraints defined on the different elements of the architecture are viewed as natural invariants for their evolution. Thus, the consistency checking is concerned with the diagnostic of the constraints that are violated in the architecture after its evolution. If no inconsistency is detected, then the evolution is validated. Otherwise, the architects have to correct the problems. The list of violated invariants is shown to the architect in order to facilitate her work. We then leave to the architect the opportunity to return to any previous state by canceling the execution of some competences. She can also make the choice of other styles on the basis of the list of styles labeled “sure” and “possible”.

## 6. Evolution Off-The-Shelf

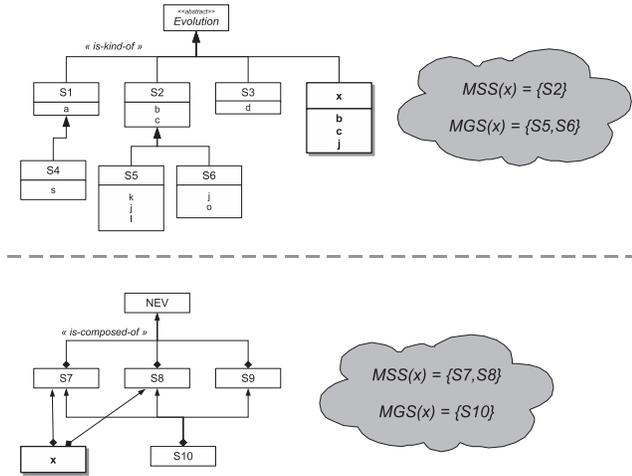
Given the principles outlined above, our aim is to provide a shelf where evolution styles could be properly managed. Storage and retrieval are the two key features of the shelf, that both rely on the hierarchical relations – specialization and composition – existing between evolution styles. The shelf works with a single central inference mechanism called static classification-driven reasoning. The purpose of such a reasoning is to infer new *is-kind-of* or *is-composed-of* relationships, respectively.

### 6.1. Basic Concepts on Classification

Generally speaking, the aim of the algorithm underlying the classification-driven reasoning is to compute the most specific subsuming styles ( $MSS_x$ ) and the most general subsumed styles ( $MGS_x$ ) of a given evolution style  $x$ . The algorithm works as well with the specialization hierarchy as with the composition hierarchy. The difference simply

<sup>2</sup>This requires to infer the exact type of a given architectural element.

holds in the subsumption test used and on the root on which the process is started, namely,  $\langle \sqsubseteq_s, Evolution \rangle$  for the specialization hierarchy and  $\langle \sqsubseteq_c, NEV \rangle$  for the composition hierarchy. This process is schematized in Figure 4.



**Figure 4. Illustration of the classification-driven reasoning performed on an evolution style  $x$  into the specialization hierarchy (top part) and on the composition hierarchy (bottom part).**

*Evolution* and *NEV* (No EVolution) are roots that are artificially introduced by the system to support single-rooted hierarchies. Each subsumption test (signature:  $A \times B \rightarrow Boolean$ ) makes possible to discover if a particular semantic relationship holds between a couple of evolution styles. This is done considering the specialization subsumption test  $\sqsubseteq_s$  that checks if the B's header is overloading the A's header, and the composition subsumption test  $\sqsubseteq_c$  that checks if B owns at least the same compositions as A.

## 6.2. Storage Feature

Acquiring new know-how is a very important step to ensure their subsequent reuse. Without assistance, the consistency of the shelf's content can not be guaranteed over time. This is particularly true when the size of the shelf reaches a certain threshold. Thus, the purpose of the storage is to take into account the existing evolution styles whenever the insertion of a new style occurs.

By default, any new specification of an evolution style is related to the *Evolution* style and to the *NEV* style, roots of the specialization hierarchy and the composition hierarchy, respectively. Then, the newly specified evolution style  $x$  is subject to a classification process to avoid redundant know-how and maximize reuse, and this, in both hierarchies. Therefore, the two sets  $MSS_x$  and  $MGS_x$  give the

right position of the new style into the given hierarchy. It simply remains to add the missing semantic relations by breaking direct connections between MSS and MGS, then by connecting  $x$  to its MSS and its MGS to itself. Notice that in the case of specialization the inheritance comes into play, implying that some headers have to be automatically restructured in order to eliminate (possible) redundancy.

## 6.3. Retrieval Feature

The shelf must provide more than a text-based search technique. Instead, the search is based on a classification-driven technique as mentioned in the beginning of this section. From a search engine perspective, the evaluation of a query gives rise to three classes of results: success, failure or candidate. Success or failure are the two basic classes, as provided by any repository techniques. Candidate is a particular class which contains alternative results, which is very useful whenever the problem stated by the architect is-a-kind-of an already known problem or is-composed-of some already known problems.

In our approach, a query is deliberately expressed as an abstract evolution style  $x$ . The query evaluation process is then rephrased as a classification process. Consequently, the two sets  $MSS_x$  and  $MGS_x$  give important information while the choice of the target hierarchy (and the subsumption test thereof) reflects the questioning of the architect. The Table 1 summarizes the various scenarios from a query  $x$ .

Class	Test	Description
Success	$\sqsubseteq_s$	$MSS_x = MGS_x$
Success	$\sqsubseteq_c$	$MSS_x = MGS_x$
Candidate	$\sqsubseteq_s$	$MSS_x$ solely
Candidate	$\sqsubseteq_c$	$MSS_x$ solely
Failure	$\sqsubseteq_s$	$MSS_x = MGS_x = \emptyset$
Failure	$\sqsubseteq_c$	$MSS_x = MGS_x = \emptyset$

**Table 1. Classes of results induced by the classification-driven reasoning performed on a query  $x$ .**

Let us detail the various scenarios below:

- Success means that there exists a mutual subsumption between an evolution style  $\delta$  and the query  $x$ , that is,  $x \equiv \delta$ . In other words,  $\delta$  has a competence part that strictly fulfills the architect's need.
- About candidate class, the difference between specialization and composition must be well understood:
  - With regard to specialization, the subsuming styles of  $x$  can substitute to the query. Indeed,

though more general, their goal is somewhat similar and therefore the architect can instantiate any of them on its architecture with confidence. Such a substitution is "type-safe" by dint of the style inheritance principle described in Section 4.2.

- With regard to composition, the subsuming styles of  $x$  are intended to be combined to satisfy the query. Putting this set of styles together can be done in an ad-hoc manner when instantiating each one on the architecture, or in a more clever way by specifying a new composite evolution style from them. It goes without saying that we advocate in favor of the latter one.
- Failure means that there does not exist any subsumption between  $x$  and the set of existing styles, regardless to the roots that are clearly not relevant results. An interesting point in case of failure is that the retrieval process switches automatically to the storage process. The purpose of that is to consider the query as an extension point<sup>3</sup> intended to be complemented later by skilled architects. Indeed, the missing competence could be provided via the competence overriding technique.

## 7. Conclusions

Our work is grounded on the assumption that software evolution could be and should be capitalized, like many other software artifacts to cope with the increasing complexity of systems and the time-to-market pressures. As a direct consequence of that, software evolution must be truly engineered in such a such that two items are verified:

1. Maximum reuse of existing capital.
2. Bringing any new achievement in this capital.

Following this vision, we presented in this paper an evolution model dedicated to the architecture level of systems where the key concept is the style. An evolution style is used to specify recurring architectural evolution operations applied onto an architectural element (e.g., component, connector, configuration). Relational elements provided by our evolution model allow architects to structure (mainly hierarchically) their evolution-related knowledge. Specifically, relying on the specialization hierarchy, a dynamic classification-driven reasoning is performed to select semi-automatically the suitable evolution styles. From this point of view, our model leverages a kind of expert system technique for software evolution activities and the current capital is merely viewed as a knowledge base. Here, the capital is supplied by the shelf. The shelf we presented supports

<sup>3</sup>i.e., the description of a problem without the description of the solution thereof

storage and retrieval in a uniform way, both features relying on a similar static classification-driven reasoning. First, acquisition of new know-how through their systematic confrontation with the exiting ones on the shelf is of particular interest with regard to item 2. Second, defining queries is necessary with respect to item 1. Besides, expressing them as full-fledged evolution styles could shorten the learning curve of architects because there is no need for a specific query language.

## References

- [1] R. Allen and D. Garlan. A formal basis for architectural connection. *ACM Trans. Softw. Eng. Methodol.*, 6(3):213–249, 1997.
- [2] O. Barais, A.-F. L. Meur, L. Duchien, and J. L. Lawall. Software architecture evolution. In T. Mens and S. Demeyer, editors, *Software Evolution*, pages 233–262. Springer, 2008.
- [3] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *Pattern-oriented software architecture: a system of patterns*. John Wiley & Sons, Inc., New York, NY, USA, 1996.
- [4] D. Garlan, J. M. Barnes, B. R. Schmerl, and O. Celiku. Evolution styles: Foundations and tool support for software architecture evolution. In *WICSA/ECSCA*, pages 131–140, 2009.
- [5] O. L. Goer, D. Tamzalit, M. C. Oussalah, and A.-D. Seriai. Evolution styles to the rescue of architectural evolution knowledge. In *SHARK*, pages 31–36, 2008.
- [6] M. M. Lehman and J. F. Ramil. Software evolution: background, theory, practice. *Inf. Process. Lett.*, 88(1-2):33–44, 2003.
- [7] J. Magee and J. Kramer. Dynamic structure in software architectures. *SIGSOFT Softw. Eng. Notes*, 21(6):3–14, 1996.
- [8] N. Medvidovic, D. S. Rosenblum, and R. N. Taylor. A language and environment for architecture-based software development and evolution. In *ICSE '99: Proceedings of the 21st international conference on Software engineering*, pages 44–53, New York, NY, USA, 1999. ACM.
- [9] N. Medvidovic and R. N. Taylor. A classification and comparison framework for software architecture description languages. *IEEE Trans. Softw. Eng.*, 26(1):70–93, 2000.
- [10] R. T. Monroe and D. Garlan. Style-based reuse for software architectures. In *ICSR '96: Proceedings of the 4th International Conference on Software Reuse*, page 84, Washington, DC, USA, 1996. IEEE Computer Society.
- [11] D. E. Perry and A. L. Wolf. Foundations for the study of software architecture. *SIGSOFT Software Engineering Notes*, 17(4):40–52, 1992.
- [12] M. Shaw. Some patterns for software architecture. In *Pattern Languages of Program Design*, volume 2, pages 255–269. Addison-Wesley, 1996.
- [13] M. Shaw and P. C. Clements. A field guide to boxology: Preliminary classification of architectural styles for software systems. In *COMPSAC '97: Proceedings of the 21st International Computer Software and Applications Conference*, pages 6–13, Washington, DC, USA, 1997. IEEE Computer Society.

# Reasoning about Attribute Architectures

Tacksoo Im, John D. McGregor  
Clemson University  
Clemson, USA  
{tim,johnmc}@cs.clemson.edu

## Abstract

*When software architects create an architecture they must balance the tradeoffs inherent among multiple quality attributes. Often this is accomplished by optimizing the architecture for each quality attribute separately. These separate, attribute-specific architectures are then combined into one cohesive architecture. Combining the architectures involves considering how to combine different types of component relationships. We define a composition operator that combines component relationships by using properties of the reasoning frameworks for the attributes represented by the relationships. We conclude that the composition operator is sufficient to compose the attribute-specific architectures into a comprehensive architecture.*

## 1. Introduction

Modern software architecture design techniques emphasize the quality attributes of the anticipated products. In order to satisfy the non-functional requirements that define acceptable levels for the quality attributes the architect applies specific tactics. Each tactic is intended to enhance a particular quality but also affects other qualities as well. Tools such as ArchE [1], intended to support the architect in this activity, focus on one quality at a time. The result is an architecture that optimizes one attribute, but is seldom optimal for other qualities.

The goal of the architect is to design one software architecture that satisfies all of the quality attribute goals of the system. The reality is that the architect can balance a couple of attributes at the same time but not the number usually associated with an industrial-strength program. One approach is to address one attribute at a time until a set of architectures are produced, each optimizing one of the qualities. The architect must then combine these multiple architectures. The contribution of this paper is to describe an investigation into techniques for combining multiple attribute-specific architectures into a single product architecture.

The initial goal of this research was to develop a reasoning framework for dependability by exploring tactics that enhance the dependability quality attribute. Dependability is a quality attribute that is defined in terms of six other quality attributes: reliability, availability, confidentiality, integrity, safety, and maintainability [4]. These qualities are measured on different scales, some of which are not even interval scales. This results in a large complex reasoning problem but one that is not unique to dependability.

In Section 2 we provide the background necessary to understand the work. In Section 3 we describe attribute-specific architectures. In Section 4.1 and Section 4.2 we give an algorithm for combining multiple attribute architectures. In Section 5 we apply the technique to an example. Finally we summarize the results so far and identify future work.

## 2. Background

Generally accepted definitions of “software architecture” include three aspects: structures, relationships, and qualities. The structures are groupings of behavior. The structures are connected by relationships. Different arrangements of these connections result in different levels of quality attributes. For the purposes of this study we reduced each structure to a single behavior we will call a responsibility. A responsibility is defined as an “action, knowledge to be maintained or a decision to be carried out [10].” We also hold the set of responsibilities constant, meaning that each of the attribute architectures will include the same set of responsibilities. We focus on the connections between responsibilities as embodied by connectors.

Connectors correspond to dependencies among code modules. A dependency may be a static relationship such as inheritance or a dynamic relationship such as control flow or data flow [5]. In software architectural languages such as AADL (Architecture Analysis and Design Language) [2], keywords such as data, event, event data, data access, bus access and port group are used to specify the different types of connectors found in the architecture [2]. A more detailed

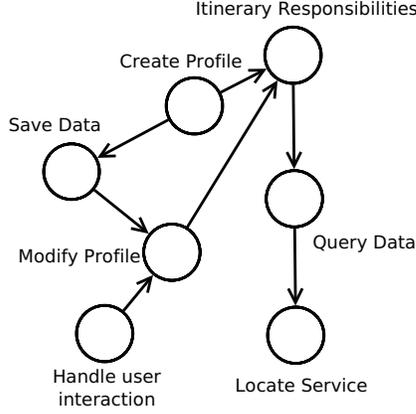


Figure 1. Dependency Graph of CTAS

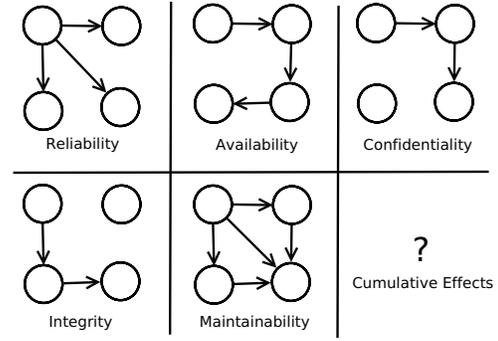


Figure 2. Attribute Architectures

analysis of the various types of connectors is found in [7].

Connectors are also found in the architectures that are derived to analyze various quality attribute requirements of a system. Figure 1, shows a fragment of a dependency graph found in the Clemson Travel Assistance System (CTAS) architecture[6]. The circles in the figure represent responsibilities and the lines represent connectors. In this example, the connectors represent dependencies and the dependency connectors are used to analyze modifiability scenarios of the architecture. Depending on the quality attribute that is being analyzed, the connectors can represent control flow, data flow, or a more general dependency.

Figure 2, represents five of the qualities that constitute dependability<sup>1</sup>. Each circle is a responsibility and the lines between circles are connectors. We will use these cartoons as a means of explaining our technique for combining architectures.

In the following sections we present the process of taking two or more attribute architectures and combining them into one product architecture.

### 3. Attribute Architectures

An attribute architecture consists of a finite set of responsibilities and its connections that determine the value of a particular quality attribute. The set of responsibilities of an attribute architecture is represented as:

$$R = \{R_1, R_2, \dots, R_n\} \quad (1)$$

and the connections among them are represented as:

$$C = \{C_{12}, C_{21}, \dots, C_{nm}\} \quad (2)$$

<sup>1</sup>We omit safety for simplicity because it is combination of multiple attributes itself.

A connection  $C_{12}$  is a connection starting from responsibility  $R_1$  and ending at  $R_2$ . Connections are created by an architect, or explicitly omitted, in an effort to optimize a specific quality attribute.

We use a common set of responsibilities across all of the attribute architectures but each attribute is characterized by distinct arrangements of connections to form a specific architecture. “Control flow” connections are used to form an attribute architecture that is being optimized for reliability because reliability is computed by tracing the program flow and combining the reliabilities of the various segments. The architecture can be expressed as (the  $r$  represents reliability and  $c$  represents a control flow):

$$A_r = \{C_{12}^c, C_{21}^c, \dots, C_{nm}^c\} \quad (3)$$

An attribute architecture consists of two elements: a set of responsibilities (which is a subset of the entire responsibilities found in an architecture) and a set of quality attribute connections.

$$A_r = \{R_r, C_r\} \text{ where } R_r \subseteq R \text{ and } C_r \subseteq C \quad (4)$$

The aggregation of attribute architectures results in a product architecture that can be expressed with  $R$  and  $C$  as in the following equation:

$$A = \{R, C\} \quad (5)$$

But deriving  $A$  becomes a composition problem when more than one quality attribute architecture needs to be “combined” to form a product architecture.

In order to derive  $A$ , a composition of the attribute architectures must be computed. Suppose we are trying to combine a reliability attribute  $A_r$  and an availability attribute architecture  $A_a$  into  $A$ .

Quality Attribute	Analytic Theory	Connector Type
Reliability	Reliability rate based calculation [4]	Control flow ( $C^c$ )
Availability	Availability rate based calculation [4]	Control flow ( $C^c$ )
Confidentiality	Authorization level based calculation [3]	Data flow ( $C^d$ )
Integrity	Authorization level based calculation [3]	Data flow ( $C^d$ )
Maintainability	Change impact based calculation [?]	Dependency ( $C^l$ )

**Table 1. Connector Types**

$$A = A_r \circ A_a = \{R_r \cup R_a, C_r \circ C_a\} \quad (6)$$

Deriving  $R_r \cup R_a$  is a straight-forward union of responsibilities but deriving  $C_r \circ C_a$  requires the composition of the connectors found in the attribute architectures. In the next section we explore the issue of combining the various connectors of different attribute architectures.

## 4. Combining Attribute Architectures

In this section, we present the basic theory on combining the attribute architectures and then present a realization of the theory based on AADL syntax.

### 4.1. Connection Types & Compositions

In order to combine the six sub-attributes architectures of dependability into one representative architecture, the connectors of each attribute architectures and their characteristics must be analyzed. Table 1, shows the connector type for each attribute architecture and shows how it relates to the analytic theory of the respective quality attribute.

The analytic theories for reliability and availability involve calculations using the control flow  $C^c$  between responsibilities. A control flow expresses the flow of execution from one responsibility to another. As such, the connector is expressed as  $C^c$  and it is used in the reliability and availability attribute architectures.

The analytic theories for confidentiality and integrity use the data flows between responsibilities. A data flow expresses the passage of data from one responsibility to an-

$\circ$	$C^c$	$C^d$	$C^l$	<i>null</i>
$C^c$	$C^c$	$C^{cd}$	<b>*See Table 3</b>	$C^c$
$C^d$	$C^{cd}$	$C^d$	<b>*See Table 3</b>	$C^d$
$C^l$	<b>*See Table 3</b>	<b>*See Table 3</b>	$C^l$	$C^l$
<i>null</i>	$C^c$	$C^d$	$C^l$	<i>null</i>

**Table 2. Combination of Connectors**

other and the connection is expressed as  $C^d$  and it is used in the confidentiality and integrity attribute architectures.

The analytic theory for maintainability uses the dependency between responsibilities. The dependencies show that a responsibility relies on some other responsibility for its operation. But as shown in [8], there are different types of connectors in a dependency connector, namely: abstraction, binding, realization, substitution and usage. With the exception of *usage*, which shows a dynamic (run time) relationship among responsibilities, the other types of dependency show a static (compile time) structural relationship. A dependency is expressed as  $C^l$  and it is used in the maintainability attribute architectures.

Connectors are combined when attribute architectures are combined to derive a product architecture. Connectors such as *control flow*, *data flow* and *dependency* are useful when analyzing quality attributes but only provide an incomplete view of the product architecture. The attribute architectures need to be combined for a complete product architecture.

One of three things can happen when we try to combine connectors. The two connectors can be of the same type and they are combined to a single connector of that type, two connectors of different but compatible types are combined and transformed into a new type of connector, or the two connectors cannot be combined and instead are left in place as a set of connectors. In the next section, we explore the specific details on how to combine the connectors found in the attribute architectures.

### 4.2. Composition Operator Defined

A composition operator is used to combine two connectors. As such, the composition operator needs to be defined for each possible pair of connector types. Depending on which type of connector is involved, a combination may be possible or not. The composition operator assumes that the connections are unidirectional and indicate a relationship in the same direction. But the composition operator can also be used to combine bidirectional connections with other bidirectional connections.

The three types of connectors found in the attribute architectures for quality attributes in Table 1 can be combined as shown in Table 2. Combining  $C^c$  and  $C^d$  with  $C^l$  is pre-

$\circ$	$C^c$	$C^d$
$C^l$ (abstraction)	×	×
$C^l$ (binding)	×	×
$C^l$ (realization)	×	×
$C^l$ (substitution)	×	×
$C^l$ (usage)	$C^l$	$C^l$

**Table 3. Dependency Connector Combination Specifics**

sented in detail in table 3 as a dependency connector can represent more than one type of dependency.

We observe the following from Table 2 when we try to combine the connectors:

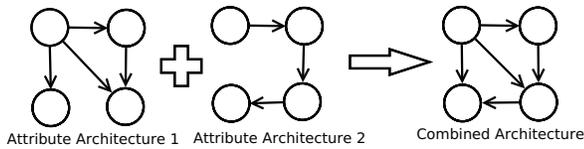
- When a control flow connector and a data flow connector are combined, a new connection called the control-data flow  $C^{cd}$  is created. For example, this would correspond to a method call with parameters at the implementation level.
- The dependency connection  $C^l$ , in some cases, subsumes the control flow connector and the data flow connector when they are combined. This is because control flow and data flow can be considered as a type of dependency. The specific implementation level meaning of the dependency combinations are shown in Table 3.
- Quality attribute values as calculated from the attribute architectures must be recalculated if new connections of the same type are introduced from another attribute architecture as part of the combination. This is further explained in section 4.3.
- The composition operator is commutative as the order in which the connectors are combined do not influence the outcome of the combination. Combining more than two connectors involves pair-wise combinations that result in a same connector regardless of the order in which the combinations were applied.

Table 3 shows that some types of dependency connectors cannot be combined with some other connectors (as indicated by ×). Instead of a combination, such pairs are left in place and show a dynamic interaction dependency [9]. The list below elaborates on the pairs found in Table 3:

- $C^l$  (abstraction)  $\circ C^c = \times$   
This can be seen as a relationship where the parent responsibility is acting as a delegate for the child responsibility. The child responsibility is an instance of the parent responsibility and the control flow shows a delegation of control to the parent.

- $C^l$  (abstraction)  $\circ C^d = \times$   
This is a case where the child responsibility has a role of a *subject* and the parent responsibility has to role of an *observer*. This relationship can be found in the observer pattern.
- $C^l$  (binding)  $\circ C^c = \times$   
This can be found in the template method pattern. The child responsibility is bound to the parent responsibility as it implements the details as specified in the parent template. Also, the child responsibility transfers control to the parent responsibility as it contains the template method.
- $C^l$  (binding)  $\circ C^d = \times$   
The same as dependency binding with control flow but instead of control, data is sent to the parent responsibility.
- $C^l$  (realization)  $\circ C^c = \times$   
This relationship closely resembles the factory method pattern. The parent responsibility defines a interface (a specification) for the child responsibility to implement and there is a control flow to the parent to access the methods found in the parent.
- $C^l$  (realization)  $\circ C^d = \times$   
The same as dependency realization with control flow but instead of control, data is sent to the parent responsibility.
- $C^l$  (substitution)  $\circ C^c = \times$   
This relationship closely resembles the protection proxy pattern where a child responsibility is substituted for the parent responsibility as a proxy. The control flow represents the proxy communicating with the parent responsibility.
- $C^l$  (substitution)  $\circ C^d = \times$   
Similar to dependency substitution and control flow but with data.
- $C^l$  (usage)  $\circ C^c = C^l$   
The dependency connector subsumes the control flow connector. This is because a dependency of type usage includes a control flow.
- $C^l$  (usage)  $\circ C^d = C^l$   
The dependency connector subsumes the data flow connector. Again, this is because a dependency of type usage includes a data flow.

The attribute architectures can be combined using the observations we have presented in this section. By combining the connectors found in each individual attribute architecture into one representative architecture, we derive an architecture that is complete with combined connectors. As



**Figure 3. Combining Attribute Architectures**

shown earlier, some connectors cannot be combined and must be left in place. In the next section, we explore how the quality attributes under consideration can change in the process of combining the attribute architectures.

### 4.3. Effects of Combining Attribute Architectures

Quality attribute values calculated from attribute architectures can change when new connections of the same type are introduced during the combination. This puts a constraint on combining attribute architectures because there are quality attribute goals that a software system has to satisfy.

Combination of attribute architectures can introduce new connections to one another as illustrated in Figure 3. Also, disparate connections from separate architectures can also be combined as shown in Section 4.2. The combination of disparate connectors can influence the quality attributes as well.

When introducing new connections where there was no connection (a null connection) in attribute architectures, there is a possibility that the quality attributes as calculated from the attribute architectures can change. Table 4 shows how replacing a null connection from one attribute architecture with another type of connector can influence the quality attributes under consideration.

The rows in the table represent attribute architectures that are introducing a new connection and the columns represent attribute architectures that are receiving the new connection. (The initials of the quality attributes are used to save space, A - availability, R - reliability, C - confidentiality, I - integrity, M - maintainability). A check mark indicates that the newly introduced connections do impact the quality attribute values of the receiving attribute architectures. A cross indicates that there is no impact.

Table 4 shows that a newly added control flow connection can possibly affect all other quality attributes values under consideration. Control flow connections can create “entry points” to confidentiality and integrity attribute architectures via new connections thereby influencing confidentiality and integrity, and making recalculation necessary. Maintainability is also influenced by newly introduced control flow connections as a control flow is a type of dependency.

	<i>A</i>	<i>R</i>	<i>C</i>	<i>I</i>	<i>M</i>
<i>A</i>	✓	✓	✓	✓	✓
<i>R</i>	✓	✓	✓	✓	✓
<i>C</i>	×	×	✓	✓	✓
<i>I</i>	×	×	✓	✓	✓
<i>M</i>	✓	✓	✓	✓	✓

**Table 4. Effects of Combination on Quality Attributes**

Newly introduced data flow connections do not influence availability and reliability, since data flow is not used in their computations. But maintainability is influenced by newly introduced data flow connections as a data flow is a type of dependency.

A newly introduced dependency connection can influence availability and reliability if the dependency type represents a control flow. It is difficult to determine if a dependency connection represents a control flow unless it is annotated. In cases where a dependency connection is not annotated, the architect would be required to determine the type of connection by examining the responsibilities involved in the connection. This is also true for dependency connections that represents data flow.

Quality attribute requirements put constraints on combining attribute architectures when null connections are converted to actual connections. But combined connections do not pose any quality attribute concern as the configuration of the attribute architectures does not change when a connection is combined.

### 4.4. Summary

The combination of attribute architectures results in a changed configuration of the architecture that often requires the recalculation of the quality attribute values under consideration. As such, the combination of attribute architectures is an iterative process that requires the architect to change the attribute architectures if a combination results in a negative effect on other attribute architectures. In the next section, we present an example of how attribute architectures represented using AADL are combined.

## 5. Example with AADL

There are six types of connectors in AADL namely: data, event, event-data, data-access, bus-access and port-group connectors. Previously we have defined that there are three types of relationships in the quality attributes under consideration, namely: control flow, data flow, and dependency.

	<i>Static</i>	<i>Dynamic</i>
<i>Implicit</i>	Property	Mode
<i>Explicit</i>	Dependency (implementation, extends, binding)	Control Flow (Event, Event-Data)  Data Flow (Data, Event-Data, Data-Access, Bus-Access)

**Table 5. Relationship Types in AADL**

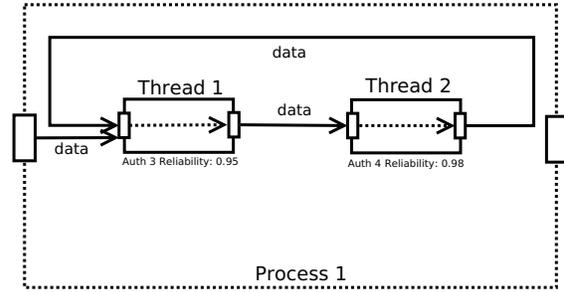
o	$C^c$	$C^d$	$C^l$	<i>null</i>
$C^c$	Event	Event-Data	Port-Group	Event
$C^d$	Event-Data	Data	Port-Group	Data
$C^l$	Port-Group	Port-Group	Port	Port
<i>null</i>	Event	Data	Port	<i>null</i>

**Table 6. AADL equivalent of Combined Connectors**

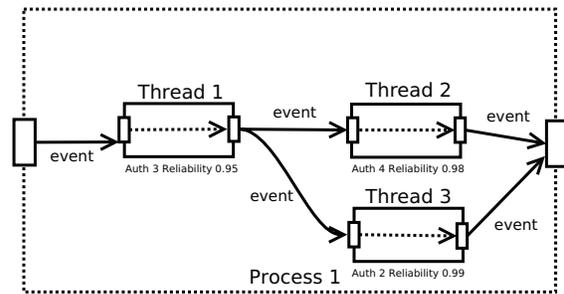
We can map the three types of relationships found in the attribute architectures to the connectors found in AADL. AADL connections are relationships that are found among AADL elements and can be broadly divided into implicit and explicit relationships. Implicit relationships are relationships that are present but not represented by specific visible constructs in the architecture. For example, having a “reliability value” property for an element is not a relationship that is explicitly shown by the architecture but nevertheless is implied by it. Explicit relationships are expressed using specific types of connectors or keywords. Table 5 shows a summary of the relationships found in AADL.

Control flow and data flow connectors have their equivalent connectors in AADL and the dependency connection can be expressed in AADL using syntactic constructs such as *implementation*, *extends*, *binding*. Table 6 illustrates the AADL equivalent connectors for the combined connectors found in the attribute architectures. A port-group connector is used to express any set of connections that cannot be resolved to a single connection. For example, a dependency combined with any other connection results in a port group connection. Previously, in Table 3, we showed that some dependency connections cannot be combined with other connections on a more abstract level.

Figure 4 shows a confidentiality attribute architecture with data connections between threads and Figure 5 shows a reliability attribute architecture. Each attribute architecture represents the responsibilities (which includes constructs such as threads) and the connections that are used to calculate quality attribute values. By applying an authorization



**Figure 4. Confidentiality Attribute Architecture**



**Figure 5. Reliability Attribute Architecture**

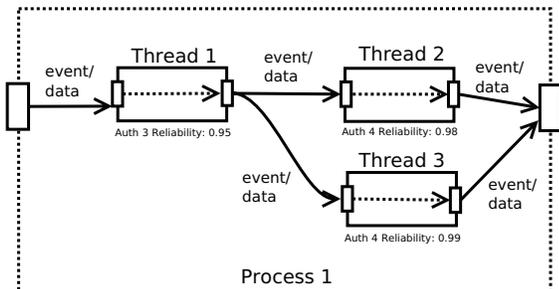
level based analytic theory, we can calculate if any confidentiality requirement violations occur in figure 4.

The authorization level based analysis for the confidentiality attribute architecture restricts access to certain responsibilities based on authorization level. Assume a scenario in which the authorization level 4 is required. That would mean that an authorization level of 4 is required for the data flow to completely take place. Confidentiality requirement is violated if the data flow is permitted with a lower authorization level or denied even if the required authorization level is met. From Figure 4, we can see that the data flow can take place with an authorization level of 4 and thus satisfies the confidentiality requirement.

The reliability attribute architecture is analyzed using standard reliability calculation methods treating the reliability values as probabilities. For the scenario that involves Figure 5, the reliability value that is required is 0.93. Equation 7 shows us the reliability values we calculate from the information from Figure 5 and we can observe that it meets the reliability requirement.

$$R = \frac{(0.95 \times 0.98) + (0.95 \times 0.99)}{2} = 0.936 \quad (7)$$

The two attribute architectures can be combined as



**Figure 6. Combined Architecture**

shown in Figure 6. Here are the specific changes made during the combination. (the list omits detailed descriptions due to space concerns)

- When a data flow (data connection) and a control flow (event connection) are found between two elements with the same direction, the two connections are combined into an event-data connection (as per Table 6). The combination was performed as it is more convenient to have one connection instead of two.
- The data connection from thread 2 to thread 1 is removed as it decreases the reliability value to less than 0.95. (the data feeds back in a loop) In order to preserve the reliability requirement the explicit connection has been removed and instead the connection is routed to thread 3 where it can exit the process and be operated on (to check for errors) before it comes back into the process.
- The authorization level for thread 3 is raised to 4 as required by the confidentiality scenario under consideration. This can be accomplished by adding additional safeguards to thread 3.

In this section, we have seen how an attribute architecture would be combined in AADL. Because a combination can potentially cause the quality attribute values calculated from the analytic theories to change (as indicated in Table 4), combining attribute architecture becomes an iterative process of considering the trade-offs that are involved.

## 6. Summary

We have defined the composition operators that are necessary in order to combine attribute architectures of certain quality attributes. We have also seen how the quality attribute values from attribute architectures can change due to combination with other attribute architectures. An example combination of attribute architectures was shown using AADL.

Combining the attribute architectures is an iterative process of combining the attribute architectures, recalculating the resulting quality attribute values, and adjusting the combined attribute architectures as needed. In the future work, we plan to extend the composition operators to include more quality attributes and their attribute architectures.

## References

- [1] A. Diaz-Pace, H. Kim, L. Bass, P. Bianco, and F. Bachmann. Integrating quality-attribute reasoning frameworks in the ArchE design assistant. *Proceedings QoSA*, pages 14–17, 2008.
- [2] P. Feiler, D. Gluch, and J. Hudak. The architecture analysis & design language (AADL): An introduction, 2006.
- [3] K. Im and J. D. McGregor. Debugging support for security properties of software architectures. In *Proceedings of CSIRW 2009*, 2009.
- [4] T. Im and J. D. McGregor. Toward a reasoning framework for dependability. In *DSN 2008 Workshop on Architecting Dependable Systems*, 2008.
- [5] B. Malloy, J. McGregor, A. Krishnaswamy, and M. Medikonda. An extensible program representation for object-oriented software. *ACM Sigplan Notices*, 29(12):38–47, 1994.
- [6] J. McGregor, F. Bachman, L. Bass, P. Bianco, and M. Klein. Using an Architecture Reasoning Tool to Teach Software Architecture. In *Proceedings 20th Conference on Software Engineering Education & Training (CSEE&T 2007)*, pages 275–282, 2007.
- [7] N. Mehta, N. Medvidovic, and S. Phadke. Towards a taxonomy of software connectors. In *Proceedings of the 22nd international conference on Software engineering*, pages 178–187. ACM New York, NY, USA, 2000.
- [8] J. Rumbaugh, I. Jacobson, and G. Booch. *Unified Modeling Language Reference Manual, The*. Pearson Higher Education, 2004.
- [9] J. Stafford, A. Wolf, et al. Architecture-level dependence analysis for software systems. *International Journal of Software Engineering and Knowledge Engineering*, 11(4):431–451, 2001.
- [10] R. Wirfs-Brock and A. McKean. *Object Design: Roles, Responsibilities, and Collaborations*. Boston, MA: Addison-Wesley, 2003.

# Formal Specification of Software Architecture Security Tactics

Andrew Wyeth and Cui Zhang  
California State University, Sacramento  
[awyeth@gmail.com](mailto:awyeth@gmail.com), [czhang@csus.edu](mailto:czhang@csus.edu)

## Abstract

*This paper provides a Z specification for the Software Architectural Tactics of Authentication and Authorization for the Security Quality Attribute. A model of a system is created and each tactic is defined with respect to the model. Each tactic is independent however, the system encompasses all the required functionality for all the tactics.*

## 1. Introduction

An important step in creating a system that meets the users functional and non-functional requirements is the design of the software architecture. To achieve system qualities (non-functional requirements), Bass, Clements and Kazman [2] recommend the use of software architecture design tactics. Software architecture design tactics are high level design decisions. Each design tactic will satisfy one or more quality attributes and may adversely affect others [2].

Security is one set of quality attributes which has three classes of tactics. The first class is Resisting Attacks which consists of: Authenticate Users, Authorize Users, Maintain Data Confidentiality, Maintain Data Integrity, Limit Exposure, and Limit Access. The first four tactics correspond to their associated intuitive definitions. The idea of limit exposure is to provide only a few services on each host. The idea of limit access is to restrict what external entities can access the system. Second, Detecting Attacks which consists of using Intrusion Detection Systems. Lastly, Recovering From Attacks which consists of Auditing and Restoration. However, when defining the security tactics of a system, there is currently no way to formally prove the implementation of the tactics. One way to establish confidence in the implementation of the tactics is to use formal methods to define the system [2].

This paper presents the definitions of the authentication and authorization security tactics with respect to a generic system. Formal specification of other security tactics can be found in a detailed document [9]. There is extensive use of the Z formal language, for a good introduction to Z see [8].

## 2. Background and Related Works

Since the work done by Shaw and Garlan [6] on formal specification of software architectural styles, significant advances have been made in Software Architectural Methodologies.

Security, as previously stated, has become a well-recognized system quality [2]. While formal methods have been associated with software architecture since the mid-nineties, formal methods have been associated with security for even longer. Since computer security became an issue, people have looked at formal models of the security of a system. This is in part because without a formal definition of security, there is no way to precisely determine if a system is secure or not [4].

Another avenue of research is specifying the properties of the security components of a system. See [1] and [5]. Song et al. [7] create a system model and prove that an intrusion detection system can detect unauthorized access to the password file. Using ACL2 (a formal language), a Unix like system model is defined and logging is explicitly added. Then the intrusion detection systems detection rules are defined and they are proved to be correct. The approach this paper uses is similar to the approach Song et al. took in proving the correctness of their system.

## 3. Formal specification of the design tactics

### 3.1. Construction of tactic specification

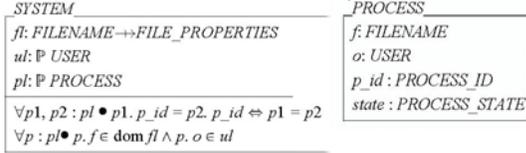
The specification for each design tactic is built upon a simple base system model which is explained in Section 3.2. The operations for each design tactic build upon the extended system model and modify the base system operations.

### 3.2. Base system model specification

The base system is a simple, high level definition of a software system. The system is described in terms of an operating system as this makes it easier to understand and provides a ready source of examples. The specification can be applied to applications or other systems by a simple mapping. The base system consists

of three elements: a userlist, a filelist, and a processlist as shown in Figure 1. USER, FILENAME, and FILE\_PROPERTIES are undefined sets, they depend on the system to be implemented. The filelist is a partial function from a filename to a file property. Thus, each filename is unique in the system and filenames do not have to map to a file property.

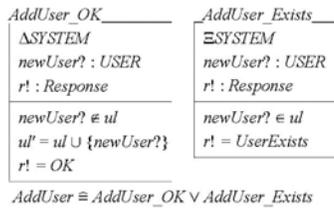
The basis of the process is that it is a file run by a user. Figure 1 shows the process specification, it consists of the user running the process, the file being run, the current state of the process and the processes id. PROCESS\_ID and PROCESS\_STATE are undefined sets. The PROCESS\_STATE represents the current state of the process. As specified in Figure 1, the system requires the process\_id is unique and that the file being executed and the user running the process both exist. This model does not consider the sharing of resources, each operation is considered atomic. If concurrency is required then the model would need to be extended to include a concurrency mechanism.



**Figure 1. Base system and process**

There are many operations on even this simple system: adding and deleting users; adding, deleting, reading, and changing files; starting, stopping and changing processes. A representative operation is discussed here: AddUser.

The AddUser operation, shown in Figure 2, consists of two possible cases: AddUser\_OK - a user is successfully added to the set of users and AddUser\_Exists - the user already exists so an error is returned.



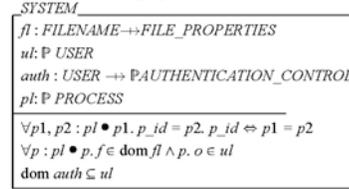
**Figure 2. Add user**

This specification describes the parts of a system that are monitored or extended to provide the various security tactics. The system as defined here has no security properties whatsoever, so any entity can manipulate the system.

### 3.3. Authentication tactic model specification

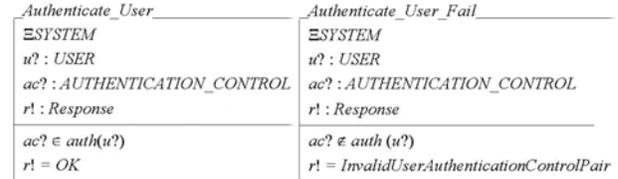
Authentication is the first security tactic to be added to the base system. Authentication is the act of verifying a users identity [3]. In software systems, authentication is achieved by having the user provide their name and a

piece of information that uniquely identifies them [3]. This unique information is provided through the AUTHENTICATION\_CONTROL set. Thus, the base system is extended to include a partial function from USER to an AUTHENTICATION\_CONTROL set as shown in Figure 3. Note that each user can have any number of authentication controls and thus there can be users in the userlist that are not authenticated. This means the creation of a user is a two step process, first the user is created, then the user is mapped to an authentication control. This allows the system to provide separation of concerns, which is one of the secure design principles [3].



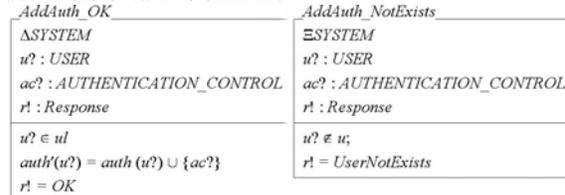
**Figure 3. System including authentication**

There are three new operations for the authenticated system. The first, consists of two parts: Authenticate\_User and Authenticate\_User\_Fail. Authenticate\_User, shown in Figure 4, provides the authentication tactic. It verifies that a user is allowed to access the system by making sure the username, authentication\_control pair is in the authenticated function. The second, Authenticate\_User\_Fail, also shown in Figure 4, returns the appropriate error message if the username, authentication\_control pair does not exist.



**Figure 4. Authenticate user**

The other two new operations added are AddAuth and DeleteAuth. They control how the authentication set is modified. Figure 5 shows AddAuth. AddAuth\_OK confirms the user exists before adding the user, authentication\_control pair to the authenticated set. If the user doesn't exist, AddAuth\_NotExists returns an error. AddAuth checks the user adding the authentication pair is authenticated, then attempts the add. DeleteAuth is similar.



$$AddAuth \equiv (\text{Authenticate\_User} \wedge (\text{AddAuth\_OK} \vee \text{AddAuth\_NotExists})) \vee \text{Authenticate\_User\_Fail}$$

**Figure 5. Add authentication**

AddAuth shows how all the operations on the system need to be modified to add authentication, the user performing the task must first be authorized, then the operation is performed or the authorization fails. Figure 6 shows how AddUser has been modified.

$$\text{AddUser} \equiv (\text{Authenticate\_User} \wedge (\text{AddUser\_OK} \vee \text{AddUser\_Exists})) \vee \text{Authenticate\_User\_Fail}$$

**Figure 6. Add user including authentication**

There are several important features that this specification emphasizes. First, the user is authenticated and then the action is performed, this prevents covert channels where different responses from the system can be used to determine properties of the system. Second, the system as specified performs complete mediation. Complete mediation, where every action is authenticated, is one of the secure design principles [3]. Third, the system, while requiring complete mediation, allows for a Kerberos style authentication scheme where the user is provided with a ticket or authentication\_control. This scheme allows the user a simpler access control, instead of having to provide their main authentication\_control for every action. Fourth, the authentication scheme is only as good as the authentication\_control, if the authentication\_control is easily compromised, then the system will not be secure. This is a feature of all authentication schemes and this specification make the association explicit. As this system model is refined, the designer will have to determine what constitutes a sufficiently strong authentication\_control for their system. For example, a system for the military would need stronger authentication than a system for a university. Finally, the system does not provide authorization. If an entity knows a correct user, authentication\_control pair then that entity can perform every operation in the system.

### 3.4. Authorization tactic model specification

Authorization is the determination of whether an entity is allowed to perform an action or not. The access control matrix, which describes the rights every user has with respect to every object in the system [3], is one way to implement the authorization tactic. For this system, the access control matrix (ACM) consists of four separate sets, one for each type of object in the base system: userlist, filelist and processlist and one set that controls access to the ACM itself.

Each access control set is a mapping from the user performing the operation and the object being operated on to a set of rights. The rights are defined to coincide with all possible actions on the system. Access to the ACM must also be controlled. Access control for the ACM is defined for each of the four sets, the user is either granted access to a set in the ACM or not. This level of granularity was chosen for the specification

because additional granularity, such as controlling whether a user can only add or delete entities from a given set in the ACM makes the specification more verbose without adding meaningfully to the specification. Figure 7 shows the new system and additional types.

$$\begin{aligned} \text{UserAuth} &::= \text{Add\_User} | \text{Delete\_User} \\ \text{FileAuth} &::= \text{Add\_File} | \text{Delete\_File} | \text{Read\_File} | \text{Change\_File} \\ \text{ProcessAuth} &::= \text{Start\_Process} | \text{Stop\_Process} | \text{Change\_Process} \\ \text{ACMAuth} &::= \text{User} | \text{File} | \text{Process} | \text{ACM} \end{aligned}$$

```

SYSTEM
-----
fl: FILENAME → FILE_PROPERTIES
ul: P USER
pl: P PROCESS
uACM: USER × USER → P UserAuth
fACM: USER × FILENAME → P FileAuth
pACM: (USER × (USER × FILENAME)) → P ProcessAuth
acmACM: USER → P ACMAuth
-----
∀p1, p2: p1 • p1. p_id = p2. p_id ⇔ p1 = p2
∀p: p1 • p. p ∈ dom fl ∧ p. o ∈ ul
dom (dom uACM) ⊆ ul
dom (dom fACM) ⊆ ul
dom (dom pACM) ⊆ ul
dom (ran (dom pACM)) ⊆ ul
ran (ran (dom pACM)) ⊆ dom fl
dom acmACM ⊆ ul

```

**Figure 7. System including authorization**

The domains of the four parts of the ACM are subsets of the userlist, a user that doesn't exist does not have rights within the system. However, a user that doesn't exist can be an object in the ACM, in fact, users and files that don't exist must be objects in the ACM, otherwise creation of objects would not be controlled by the ACM.

<pre> AddUACM_OK ----- ΔSYSTEM u?: USER nu?: USER au?: USER rights?: P UserAuthorization r!: Response  User ∈ acmACM (u?) nu? ∈ ul uACM' = uACM ∪ {(nu?, au?) → rights?} r! = OK </pre>	<pre> AddUACM_IR ----- ΞSYSTEM u?: USER nu?: USER au?: USER rights?: P UserAuthorization r!: Response  User ∉ acmACM (u?) r! = IncorrectRights </pre>
$\text{AddUACM} \equiv \text{AddUACM\_OK} \vee \text{AddUACM\_IR} \vee \text{AddUACM\_muNE}$	
<pre> AddUACM_muNE ----- ΞSYSTEM u?: USER nu?: USER au?: USER rights?: P UserAuthorization r!: Response  User ∈ acmACM (u?) nu? ∉ ul r! = UserNotExists </pre>	

**Figure 8. Add user ACM**

The system has four new elements, the sets in the ACM and each element needs to be modifiable, therefore there are eight new operations, an add and a delete for each set in the ACM. Access to these operations is authorized through the acmACM set of the ACM. Figure 8 shows AddUACM. AddUACM takes three users, the first user is the user modifying the uACM, the second user is the user being granted the right and the third user is the object that the right is

being granted for. For example, AddUACM\_OK(u1, u2, u3, {Add\_User}) would, assuming u1 has the right to modify the uACM, allow u2 to add u3 to the userlist. The other operations for adding and deleting entries in the uACM, fACM, pACM and acmACM are similar.

Most of the revised operations of the base system are modified in a similar way to authentication, except each operation requires a check that the user has the correct rights, not that the user has the correct authentication\_control. Figure 9 shows the modified AddUser.

AddUser_Authorization_OK	AddUser_Authorization_Fail
$\exists$ SYSTEM	$\exists$ SYSTEM
user? : USER	user? : USER
newUser? : USER	newUser? : USER
r! : Response	r! : Response
Add_User $\in$ userACM(user?,newUser?)	Add_User $\notin$ userACM(user?,newUser?)
r! = OK	r! = IncorrectRights

$AddUser \equiv (AddUser\_Authorization\_OK \wedge (AddUser\_OK \vee AddUser\_Exists)) \vee AddUser\_Authorization\_Fail$

**Figure 9. Add user including authorization**

The operations on processes have been modified to include a runAs user. The idea behind this is a user might want to start a process that has fewer rights than they do, for example, root on a Unix system starting the mailer daemon should not run the mailer daemon as itself.

The authorization of a system operates similarly to the authentication of a system, it wraps the existing system so a user must be authorized before they can try and perform any action. There are several features of authorization that this specification emphasizes. First, it does not perform authentication, which is a separate tactic. This does mean that if a system only implements authentication, if someone can imitate a user, they would immediately have that users rights. This is like having a Unix system where all the passwords are empty. In practice, authentication and authorization will almost always be used together. Second, like authentication, this specification performs complete mediation; every operation is checked to make sure the user attempting the operation is allowed to perform that action. Third, the ACM must be created in an appropriate manner, if a user is given rights that they should not have, the information within the system could be compromised. Finally, the ACM is capable of representing any style of access control: Roll Based Access Control, Discretionary Access Control or Mandatory Access Control by defining constraints on how the ACM is modified.

## 4. Discussion

This paper provides Z specifications of architectural tactics that can be used to achieve the security goals of a system. A base system model, with no security is specified and the authentication and authorization

tactics are defined as constraints on the extended base system model.

This paper also demonstrates the power of creating Z specifications of architectural tactics and formal languages capacity to capture high-level abstract concepts.

Formal specifications of software systems provide a myriads of benefits [6]. The formal specification of the design tactics for the security quality attribute is no exception. First, it provides a template or building block for creating systems that make use of the various tactics. Second, the formal specifications of the tactics clarify the requirements of each tactic. Third, a formally defined system can also be rigorously analyzed. Forth, the tactic specifications create a framework in which to analyze specific security mechanisms.

There are several avenues of additional research. First, it is necessary to formally specify the other quality attributes. Second, as many of the tactics require policies such as an intrusion detection or logging policy, there is an area of additional work to create templates for the aforementioned policies.

For a detailed discussion of the results and future work, see [9].

## 5. References

- [1] A. E. Abdallah and E. J. Khayat, "Formal Z Specifications of Several Flat Role-Based Access Control Models," in *Proc. of Software Engineering Workshop*, 2006, pp. 282-292.
- [2] L. Bass, P. Clements and R. Kazman, "Software Architecture in Practice, " 2<sup>nd</sup> ed., Addison-Wesley, 2003.
- [3] M. Bishop, "Computer Security Art and Science," Addison-Wesley, 2003.
- [4] Carl E. Landwehr, "Formal Models for Computer Security," *ACM Computing Surveys (CSUR)*, vol. 13, issue 3, pp. 247-278, Sep. 1981.
- [5] S. Morimoto, S. Shigematsu, Y. Goto and J. Cheng, "Formal Verification of Security Specifications with Common Criteria," in *Proceedings of SAC*, 2007, pp.1506-1512.
- [6] M. Shaw and D. Garlan, "Software Architecture Perspectives on an Emerging Discipline," New Jersey: Prentice Hall, 1996.
- [7] T. Song, J. Alves-Foss, C. Ko, C. Zhang, and K. Levitt, "Formal Reasoning about Intrusion Detection Systems", in *Proceedings of the International Symposium on Recent Advances in Intrusion Detection*, 2004, pp.278-295.
- [8] J. M. Spivey, (2009, January 10). *The Z Notation: A Reference Manual*. (2nd ed.) [Online]. Available: <http://spivey.orient.ox.ac.uk/~mike/zrm/>
- [9] A. Wyeth, "Formal Specification of Software Architecture Design Tactics for the Security Quality Attribute," M.Sc. thesis, California State University, Sacramento, Sacramento, Ca, USA, 2009.

# Enterprise Systems Development: Impact of Aspect Oriented Software Architecture

Pawan Kumar Verma

Jaypee University of Information Technology (JUIT)  
Waknaghat, Solan, H.P., India

Deepak Dahiya

Jaypee University of Information Technology (JUIT)  
Waknaghat, Solan, H.P., India

**Abstract** - Typical software comprises of several crosscutting concerns. Aspect-Oriented Software Development (AOSD) is becoming a new technique, which provides modularization of crosscutting concerns. There are number of web application frameworks available. Struts is one of the open source frameworks. This paper presents an Aspect Oriented Software Architecture which provides modularization of crosscutting concerns with minimum code tangling and scattering; and also implements an E-Governance portal using struts framework. By this architecture not only the design efficiency can be improved but also the model built is easier to comprehend and reuse.

**Keywords:** E-Governance, Struts, Frameworks, crosscutting

## I. INTRODUCTION

Managing complex software system is one of the most important problems for software engineering to handle. Aspect Oriented Programming [1, 2, 3, 4 and 5] is the most promising solution to the problem of creating modularized, well encapsulated objects. Aspect Oriented Design focuses on the explicit representation of crosscutting concerns. Our approach to Aspect Oriented Design [6] is to use AspectJ (an Aspect Oriented extension to Java) language support for Aspect Orientation in the software architecture that promises crystalline nature in distributed environment. E-Governance portal can be used to provide services and important information to citizens through internet, it changes the way of working of government which are operated from traditional approaches to an advanced and more efficient operations. The major advantages of adapting E-Governance / E-Business portal to any government are: removal of location and availability restrictions, reduction of time and money spent, heightening customer services and the provision of competitive advantages [7]-[8]. E-Governance offers a new way forward, helping improve government processes, connect citizens and build interactions with a civil society.

## II. RELATED WORK

The architecture of a program or computing system is the structure of the system, which comprises software components, the externally visible properties of those components and the relations among them [9]. At present, there is no software architecture design method which makes an unambiguous distinction between non-crosscutting and crosscutting features. Only few architecture design process can surmount this problem. PCS, DAOP-ADL, AOGA, Trans SAT, ASAAM etc are some architectural design approaches that explicitly treat crosscutting architectural concerns and are described in [10-12].

In contemporary architectural approaches, these cross-cutting concerns often cannot be cleanly decomposed from the rest of the system in both the design and implementation, and can result in either code scattering and tangling or both. This architectural design approaches offer explicit mechanism to identify and specify aspects at the architecture design level. This is different from traditional approaches where architectural aspects are implicit information in the specification of the architecture.

Several software architecture specification approaches have been proposed for modeling software architectures. We can classify these approaches in two categories: textual based [13, 14] and visual based [15]. Textual based architectural specifications are represented as so-called software architecture description languages. Visual models of software architecture improve the understanding and communication within a development team. These models are meant to be a natural reflection of an architecture designer's thoughts, and to support (at least, not impair) the traceability, composability, evolvability, and scalability within a software development process. The next section of this paper proposes an aspect oriented software architecture design approach.

## III. PROPOSED APPROACH: ASPECT ORIENTED SOFTWARE APPROACH

Aspect Oriented Software Development implements individual concerns in a loosely coupled fashion, and combine these implementations to form the final system. AOSD creates system using loosely coupled, modularized implementations of crosscutting concerns. In our proposed approach mainly three distinct development steps are performed:

1. **Aspectual decomposition:** Decompose the requirements to identify crosscutting and common concerns. We separate module-level concerns from crosscutting system-level concerns.
2. **Concern Implementation:** In this step, we implement each concern separately.
3. **Aspectual re-composition:** In this step, all module-level concerns and system-level concerns are integrated according to system requirement. The re-composition process known as "weaving" or "integrating".

Our research work focuses on developing efficient procedure and well defined set of activities to identify, represent and weave aspects or concerns in the software design. The proposed architecture provides a sequential flow of data between various layers described in the Aspect Oriented Software Architecture. The architecture can be employed to design enterprise application with a systematic and structured manner.

The complex or enterprise application can be broken into different layers that further communicate with each other to provide the complete implementation. Every layer described in the architecture has its fixed defined functionality which is required to be implemented by the application and these layers communicate between themselves to give the desired functionality.

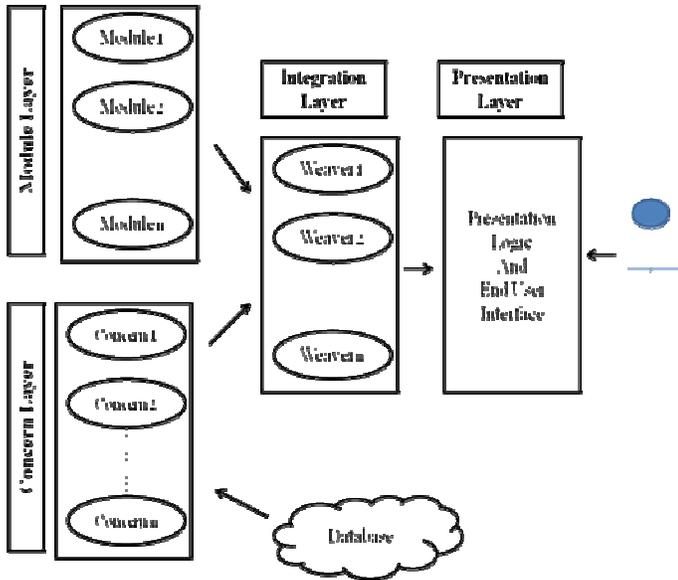


Figure 1: Proposed Aspect Oriented Software Architecture (AOSA)

Our proposed architecture contains four different layers:

- **Module Layer:** The main objective of this layer is the breaking of application in to major functionality and concern. It implements only major functionality of application to be developing.

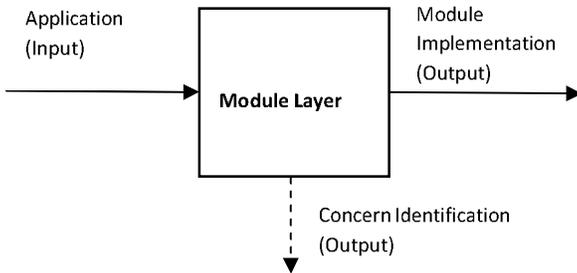


Figure 2: Module Layer

- **Concern Layer:** It comprises of the different concerns present in the concerned application. Concern of any web application can be security, authentication, encryption etc. This layer implements those concerns which are identified in previous layer.

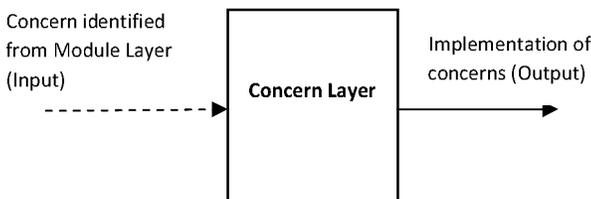


Figure 3: Concern Layer

- **Integration Layer:** After the implementation of the various modules and concerns of the system, the logic to link the concerns with module functionality is provided which is referred to as integration layer. It takes the input from the module layer and the aspect layer i.e. the modules that are required (module) are integrated with the concern in the order that define the basic work flow.

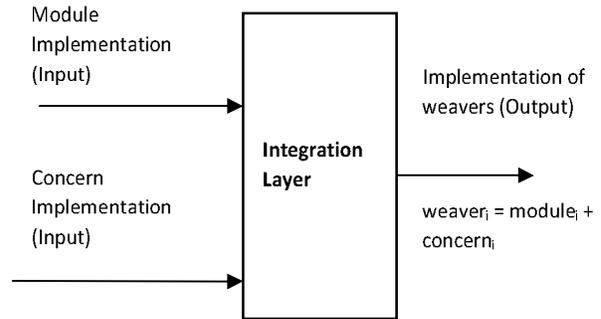


Figure 4: Integration Layer

- **Weaving Layer:** The layer comprises of the complete application logic that describes the connectivity between the various weavers. In addition to the application logic the layer also provide a user interface that provides the end user with interaction to the system.

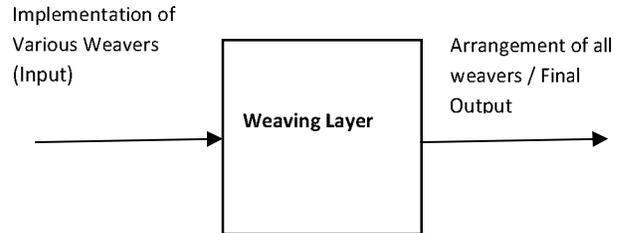


Figure 5: Weaving Layer

#### IV. IDENTIFICATION OF CROSSCUTTING CONCERNS

Concerns are introduced into the software development process with the beginning of requirements engineering. Modularization of concerns is important for software development. Identification of crosscutting concerns plays an important role in aspect mining, defect detection and software maintenance. Several research studies have resulted in number of techniques and tools for identification of crosscutting concerns. We are showing the comparison of eight different techniques applicable either on source code or on requirement document: ConcernMapper, Fan-in analysis, Theme/Doc, Prune dependency rule, Formal concept analysis, event traces, line co-change and clone detection [16]. The techniques are compared on five different parameters: Stage of Software development when the technique is applicable, whether the technique is supported by a tool, number of input parameters or

the quantum of information required by the technique, format of output generated and software on which the technique have been applied or tested.

	<b>Software Development Stage</b>	<b>Input Parameters</b>	<b>Format of output</b>	<b>Tool Support</b>	<b>Software on which the technique has been tested</b>
<b>Prune Dependency Rule</b>	After Implementation requires source code	Source Code/ Implementation Unit	Mapping of requirement with the related code	Manual	Medium sized projects 13-44 KLOC
<b>Concern Mapper</b>	After Implementation	Requirement	Concern graph	Tool support exists (concern mapper)	
<b>Fan-in Analysis</b>	After Implementation	Source Code		Semi Automatic tool: FINT	PET Store, JhotDraw and Tomcat
<b>Theme</b>	Analysis and Design	List of key actions identified from requirements and the requirement Document	Action View: is a graph like structure depicting the relation between actions and requirement	Theme/Doc tool	Crystal game having 85 requirement
<b>Line Co-change</b>	Maintenance	Source code and the bug report	Mapping of Lines of code changed together when a bug is corrected	Manual	JHotDraw
<b>Clone Detection</b>	After Implementation	Source code	The code clones are identified and the code clones are reflected with highlight	CC finder	ASML: a producer of lithography system CC with 16406 lines of C code
<b>Formal Concept analysis using execution traces</b>	After Implementation	Execution trace or use cases and source code	Concept Lattice which relates the use cases with the program elements	Dynamo for tracing method execution	Java Implementation of Dijkstra Algorithm
<b>Event Trace</b>	After Implementation	Execution trace of method calls	Recurring patterns of method following or followed by another method and nested calls	Dynamic Aspect mining tool: DynAMiT	Graffiti: editor for graphs and a toolkit for implementation of graph visualization developed in Java

Figure 6: Comparison of crosscutting concerns identification Techniques [16].

#### V. CONSTRUCTION OF E-GOVERNANCE PORTAL BASED ON STRUTS FRAMEWORK

This paper has taken an example of E-Governance [17] named “Vital Records Information for India” portal based on above framework. Vital Records Information for India (VRII) maintains and groups together information from various public services departments that come under the preview of collector’s office namely, schools, hospitals etc. Conventionally, the citizen has to go to collector’s office in person to get issued certificates like for community, birth, income and driving license etc. This results in wastage of time. Here we provide the online registration facility to apply for

various documentation certificates for the citizens so that the public in general has to visit the collector’s office only once at the time of submitting relevant documentary proofs or where physical presence is mandatory.

This E-Governance portal has various functionalities. In this section we discuss only login approach which is used by Struts Framework. For this approach, following we require some files namely

- Login.jsp: This page provide user interface for login page. In this page Struts HTML Tags are used for developing user interface. And form also contain one submit button, login action class is called when this submit button is

clicked. <html:errors /> tag is used to display the error message to user.

- Success.jsp
- Failure.jsp
- Web.xml: For any web application we need to define web.xml file. This file describe first page of that application.
- Struts-config.xml: After submitting the form of Login.jsp page, validate method of LoginForm class is called. If there is any error, like username is missing or password is missing, then control is returned back to the input page where errors are displayed to the user.
- LoginAction.java: Business logic of web application is written within execute method of LoginAction class. If the user name and password is correct then we forward the user to the success page else we forward to failure page.
- LoginForm.java: This java file contains validate method; this method is used for checking the entries of username and password of login.jsp page. If there is any error then corresponding error message is displayed to the user.
- ApplicationResource.properties: It contains all error messages which are used in our application. By separating error message we can make any change any time without making any changes to the java files or jsp pages.

## VI. CONCLUSION

This paper has provided an overview of the primary aspect-oriented architecture design approaches (like AOGA, TranSAT, and PCS Framework etc.) that had been published before. This paper has considered both the process, and the modeling technique. Basing on them our proposed Aspect Oriented Software Architecture (AOSA) was integrated. It utilized the complementary features of the source approaches. By proposed architecture not only the design efficiency can be improved but also the model built is easier to comprehend and reuse. And this paper also discusses implementation of web portal names “Vital Records Information of India” using Struts framework.

## VII. FUTURE WORK

Struts framework is a classical implementation of MVC architecture. Hibernate is a powerful technology for persisting data, and it enables Application to access data from any database in a platform-independent manner. Spring is a dependency injection framework that supports IOC. Future work of this paper is to develop an enterprise application which is based on SSH (Struts, Spring and Hibernate) and also uses Aspect Oriented approach.

## VIII. REFERENCES

- [1] Charles Zhang, Dapeng Gao and Hans-Arno Jacobsen, “Generic Middleware Substrate Through Modelware” , Paper published in book of Middleware 2005, LNCS 3790, ISBN: 978-3-540-30323-7, pp.314-333.
- [2] Siobhan Clarke and Robert J. Walker. “Towards a Standard Design Language for AOSD,” Paper published in ACM Proceedings on Aspect Oriented Software Development, (April 2002), pp. 113- 119.
- [3] Siobhan Clarke and Robert J. Walker. “Composition Patterns: An Approach to Designing Reusable Aspects”, Paper published in ACM Transactions on Software Engineering Journal, (Oct 2001), pp. 5-14.
- [4] Ruzanna Chitchyan, Awais Rashid, Pete Sawyer, Allesandro Garcia, Monica Pinto Alarcon, Jethro Bakker, Bedir Tekinerdogan, Siobhan Clarke and Andrew Jackson, “Survey of Aspect -oriented Analysis and Design Approaches”, Report of the EU Network of Excellence on AOSD, 2005.
- [5] Kiczales G, Lamping J, Mendhekar A, et al. “Aspect-Oriented Programming”, In Proceedings of the European Conference on Object-Oriented Programming, Springer-Verlag, Finland, 1997, pp. 220-242.
- [6] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, “Design Patterns: Elements of Reusable Object-Oriented Software”, Addison-Wesley, 1994.
- [7] Hendershot, R. (2007), “E-business Benefits: Learn about the advantages of adopting an eBusiness solution for your small business”, [www.enetsc.com/EBusinessArticles.htm](http://www.enetsc.com/EBusinessArticles.htm)
- [8] E-business and its Advantages (2006), Accessed from: <http://onlinebusiness.volusion.com/articles/e-businessadvantages/>
- [9] L.Bass, P.Clements and R.Kazman, “Software Architecture in Practice”, Addison Wesley, 1998.
- [10] B. Tekinerdogan, “ASAAM: Aspectual software architecture analysis method”, Paper published in Proceedings of the 4th Working IEEE/IFIP Conference on Software Architecture, 2004.
- [11] M. M. Kande. A concern-oriented approach to software architecture. PhD thesis, Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland, 2003.
- [12] O. Barais, E. Cariou, L. Duchien, N. Pessemier, and L. Seinturier, “TranSAT: A Framework for the Specification of Software Architecture Evolution”, Presented at Workshop on Coordination and Adaptation Techniques for Software Entities (held with ECOOP 2004), Oslo, Norway, 2004.
- [13] M. Pinto, L. Fuentes, and J. M. Troya, “DAOP-ADL: An Architecture Description Language for Dynamic Component and Aspect-Based Development”, presented at International Conference on GPCE, Erfurt, Germany, 2003.
- [14] D. Garlan, R. T. Monroe, and D. Wile, “Acme: Architectural Description of Component-Based Systems”, in Foundations of Component-Based Systems, M. S. Gary T. Leavens, Edition: Cambridge University Press, 2000, pp. 47-68.
- [15] U. Kulesza, A. Garcia, and C. Lucena, “Towards a Method for the Development of Aspect-Oriented Generative Approaches”, Workshop on Early Aspects, OOPSLA'04, Vancouver, Canada, November 2004.
- [16] Arvinder Kaur and Kalpana Johari, “Identification of Crosscutting Concerns: A Survey”, International Journal of Engineering Science and Technology, vol.1 (3), 2009, pp 166-172, ISSN: 0975-5462.
- [17] Web link: <http://www.gnn.in>, Last accessed Jan 10 2010.

# Reducing Black-box Test Suite Using Input Parameter Relationships

Lixin Wang<sup>1,2</sup>

<sup>1</sup> College of Information Science and Technology, Donghua University, Shanghai, 201620, China

<sup>2</sup> School of Electronics and Information Engineering, Anhui Institute of Architecture and Industry, Hefei, 230022, China

wlx@mail.dhu.edu.cn

## Abstract

*Under many circumstances, input parameters of object under test (OUT) have relationships with each other and testers often use input parameter combinations as test cases to test the OUT. Thus, there are many ineffective input parameter combinations due to the input parameters relationships. In this paper, we propose a method of reducing ineffective input combinations, which is based on a solution space tree and input parameters relationships. we firstly analyzes the dependent relationships among input parameters, and build a solution space tree corresponding to input parameters, then use the relationships to reduce branches of the solution space tree, and lastly traverse the tree and generate a small black-box test suite. The method solves the problem of reducing combinatorial test cases from a new perspective, and the experiments show that the method is feasible and effective.*

## 1. Introduction

The combinatorial test is one of main methods in black-box tests. The combinatorial test generates test suites by selecting values of input parameters and combining these values as test cases [1, 3]. In the definition of combinatorial test, a precondition is that the input parameters of object under test (OUT) are independent with each other.

However, there are often some relationships among these input parameters of OUT. For example, some tests for interfaces of systems often are integration tests of some modules, but in many cases, programmers have done some restrictions on input parameters in modules or glue codes so that some dependent relationships exist in the interface inputs of a system. There also are other cases, such as users store information into databases, etc. The restrictions of data fields in a relative database must be reflected on the interface of the store module of a system in

order to save these proper data into the database. This will also causes dependent relationships of input parameters on the interface of the module.

Due to relationships among input parameters, some combinations of input parameters cannot satisfy the user's demands, or do not exist, or cannot be obtained. We call these combinations as ineffective input combinations. When we generate combinatorial test suite such as  $k$ -wise ( $k \leq n$ ,  $n$  is the number of input parameters.) test suite, there may be many ineffective input combinations so that it bring some troubles for test work. So, it is necessary to find methods to remove these ineffective combinations of input parameters.

Many researchers have presented some methods to reduce combinatorial test suites. Most researchers focus on considering pair-wise or  $k$ -wise combinatorial test of input parameters, which are independent with each other [2, 3, and 4]. Some researchers used Input-Output relationships to reduce combinatorial test cases [5, 7, 8]. Nevertheless, they all do not consider the dependent relationships among input parameters.

In this paper, we present a method of reducing ineffective combinations of input parameters. The method is based on the solution space tree model, then reduces the tree branches and leaves by using the dependent relationships of input parameters, and at last, obtains those effective combinations of input parameters by traversing the tree.

## 2. A solution space tree-model for combinatorial test suite

### 2.1. The concepts of combinatorial test

Combinatorial test suite is composed of all combinations of input parameters values. For a OUT with multiple input parameters which are independent, suppose  $X$  is the set of all input parameters  $x_1, x_2, \dots, x_n$ , and we will have a set of test data values for each of the input parameters:  $D(x_1), D(x_2), \dots, D(x_n)$ . The  $n$ -

wise combinatorial test is to test every possible combinations of the selected test data values in  $D(x_i)$  ( $i=1,2,\dots,n$ ). The set  $T$  of  $n$ -wise combinatorial tests is the Cartesian product of the test data values of the individual input parameters [6]:

$$T = D(x_1) \times D(x_2) \times \dots \times D(x_n)$$

And  $T$  cardinality is  $|T|$ (i.e.  $|D(x_1)| * |D(x_2)| * \dots * |D(x_n)|$ ).

In the combinatorial test method, the smaller the  $i$  is, the smaller the number of the  $i$ -wise combinatorial coverage test cases in test suite is. Testers may satisfy different test demands using different “wise” combinatorial coverage test suite. In this paper, we will introduce how to generate and reduce the  $n$ -wise combinations of input parameters.

## 2.2. The tree model representation of combinatorial test cases

A  $n$ -wise combinatorial test suite can be expressed with a solution space tree model. Suppose a solution space tree has  $(n+1)$  levels nodes, then a test case  $t$  in  $T$  is a path from the root node to leaf ( $l_1, l_2, \dots, l_n$ ) in the solution space tree  $TR$ , where  $l_i(i=1,2,\dots,n)$  is a combination of branch values from the root to a  $(n+1)$ -th level node. See Figure. 1.

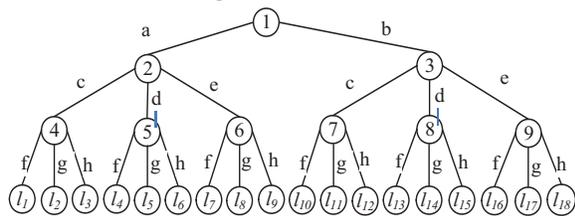


Figure 1. The solution space tree (where  $n=3$ ,  $D(x_1)=\{a, b\}$ ,  $D(x_2)=\{c, d, e\}$ ,  $D(x_3)=\{f, g, h\}$ )

A solution space tree has some properties as follows:

1. The number of leaves in the tree is  $|T|$ ;
2. There are  $n$  levels of non-leaf node in tree corresponding to  $n$  input parameters in OUT;
3. Every nodes of the same level in the tree have the same branches.

By observation, we associate the path set  $P$  with input parameter combination set  $T$ . If the number of input parameters  $n$  and the cardinalities of sets  $D(x_n)(i=1,2,\dots,n)$  are large, the cardinalities of set  $T$  will become a very large number. This will bring great difficulties or even it is impossible to execute every element in  $T$  while testers test the OUT.

Meanwhile, the set  $T$  is obtained at an assuming precondition which all input parameters are independent with each other. In factual, input parameter dependent relationships often exist in many OUTs and testers may use it to reduce the test suite.

## 3. Reducing solution space tree-model by relationships among input parameters

### 3.1. The analysis of dependent relationships among input parameters

In many cases, not all combinations of values of input parameters are meaningful for combinatorial testing. The dependent relationships should be systematically considered during modeling and generating combinatorial test suite.

For example, with respect to an OUT, we test an inquiry interface of the OUT, which has year, month, and day selections. After testers select February (not other months), there are 1, 2, ..., 28 or 29 in the day item list, but there are not 30 and 31. In other words, testers cannot obtain some combinations such as (year-number, February, 30) and (year-number, February, 31), since there have been some relative restrictions among this input parameters in module codes, and some relative tests may have been conducted during module test. Therefore, in system test, we can find that the values of input parameter days are depend on the values of input parameter month in the OUT.

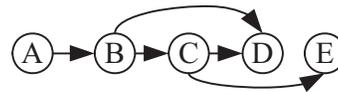


Figure 2. Dependent relationships.

If testers want to use the dependent relationships to reduce the combinatorial test suite, the first task is the analysis for these dependent relationships.

There are some dependent relationships between two parameters or among more than two parameters. The kinds of dependent relationships are classified as follows [9]:

**One-one:** The subset or all of values of one parameter depend on the subset or all of values of another parameter. B is dependent on A in Figure 2.

**One-many:** The subset or all of values of one parameter depend on the subsets or all of values of many other parameters. D is dependent on B and C in Figure 2.

**Many-one:** The subsets or all of values of many parameters depend on the subset or all of values of one parameter. D and E are dependent on C in Figure 2.

**Many-many:** The subsets or all of values of many parameters depend on the subsets or all of values of other more than one parameter.

For example, Date includes  $year(y)$ ,  $month(m)$  and  $day(d)$ . Suppose the values of  $year$  belong to  $\{1999,$

2000}; the values of *month* belong to {1, 2, ..., 12}; the values of *day* belong to {1, 2, ..., 31}. We divide the values set of *year* into two subset {{1999}, {2000}}, the values set of *month* into {{1, 3, 5, 7, 8, 10, 12}, {4, 6, 9, 11}, {2}}, and the values set of *day* into {{1-28}, {1-29}, {1-30}, {1-31}}. We use the form such as  $f(a \in A) \Rightarrow b \in B$  to denote a dependent relationship, which means *b* belongs to *B* if *a* belongs to *A*. The dependent relationships among *year*, *month* and *day* are as follows:

$f(y \in \{1999\}) \Rightarrow d \in \{\{1-28\}, \{1-30\}, \{1-31\}\}$ ; (Note: This set's form {{1-28}, {1-30}, {1-31}} indicates that the three subsets are possible to be chosen for the different *month*.)

$f(y \in \{2000\}) \Rightarrow d \in \{\{1-29\}, \{1-30\}, \{1-31\}\}$ ;

$f(m \in \{1,3,5,7,8,10,12\}) \Rightarrow d \in \{1-31\}$ ;

$f(m \in \{4,6,9,11\}) \Rightarrow d \in \{1-30\}$ ;

$f(m \in \{2\}) \Rightarrow d \in \{\{1-28\}, \{1-29\}\}$ ;

$f(y \in \{1999\} \text{ and } m \in \{2\}) \Rightarrow d \in \{1-28\}$ ;

$f(y \in \{2000\} \text{ and } m \in \{2\}) \Rightarrow d \in \{1-29\}$ .

For the above-mentioned cases, the programmers have often considered these dependent relationships in relative modules that constitute the OUT, and restrict these dependent relationships in program codes. Therefore, for such an OUT, removing these ineffective combinations of input parameters from combinatorial test suite is an important work for testers.

### 3.2. Reducing solution space tree by using the dependent relationships

From the analysis in section 3.1, we know that input parameter's combinations are classified into two kinds (i.e. the effective and the ineffective). The effective or the ineffective combinations are decided with whether or not input parameters existing dependent relationships. On solution space tree, an effective or ineffective combination of input parameter is a path from the root to a leaf. We may reserve those paths corresponding to combinations with dependent relationship or remove those paths corresponding to combinations without dependent relationship, and then we can obtain effective combinations by traversing the tree.

We may classify the ways of reducing the branches of solution space tree as follows.

(a) Reducing the adjacent next level branches in solution space tree. See Figure 1. If there is a dependent relationship  $f(x_1 \in \{a\}) \Rightarrow x_2 \in \{c\}$ , we will traverse down the tree along the branch with the state value *a* from the first level node. When we find the next level node adjacent to the branch with state value *a*, and then reduce all of the next level branches

adjacent to the found level node and whose values are not *c*. The left sub-tree is the reducing result in Figure 3.

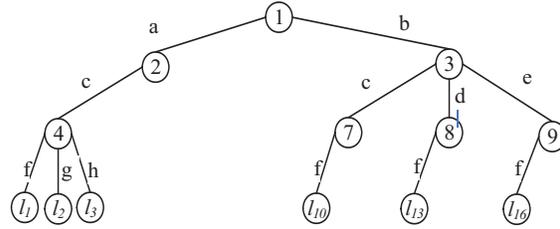


Figure 3. Reducing example 1

(b) Reducing the nonadjacent next level branches in the tree. See Figure 1. If there is a dependent relationship  $f(x_1 \in \{b\}) \Rightarrow x_3 \in \{f\}$ , we will traverse down the tree along the branch with the state value *b* from the first level node. When we find the third level nodes, which are in the branch with value *b*, and then reduce all of the third level branches adjacent to the third level nodes and whose values are not *f*. The right sub-tree is the reducing result in Figure 3.

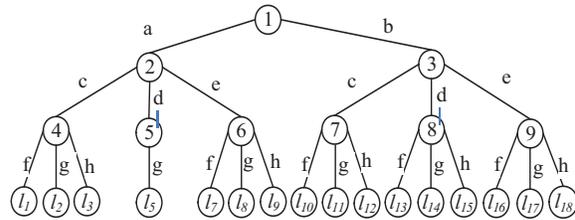


Figure 4. Reducing example 2.

(c) Reducing the nonadjacent next level branches when the left of the expression of a dependent relationship has more than one dependent condition. See Figure 1. If there is a dependent relationship  $f(x_1 \in \{a\} \text{ and } x_2 \in \{d\}) \Rightarrow x_3 \in \{g\}$ , we will traverse down the tree along the branch with the state value *a* adjacent to the first level node and go on down to the branch with the state value *d* adjacent to the second level node. When we find the third level node, which links to the branch with the state value *d* in the third level branches, and then reduce all of the third level branches adjacent to the found node and whose branch value are not *g*. The left sub-tree is the reducing result in the Figure 4.

By the above analysis, we know that after reducing solution space tree, we may traverse the reduced tree and obtain effective combinations.

## 4. The implementation of algorithm

There are several methods to implement the algorithm for obtaining the *n*-wise input parameter

effective combinations. A simple method to obtaining test cases is by using a program with  $n$  multi-loops. In  $n$  multi-loops codes, we inspect the every  $n$ -wise input parameter combinations and output the results of  $n$ -wise input parameter combination if the  $n$ -wise combination is satisfied with a dependent relationship, or discard it if not satisfied. The method is clear and understood easily, but if  $n$  or  $|D(x_i)|$  is large, the method will need more time to obtain the final results, and the running time is  $O(\prod_{i=1}^n |D(x_i)|)$  [10].

We also can build a solution space tree to obtain combinatorial test cases at a small cost of running time and space. There are two ways to utilize the solution space tree model for combinatorial test cases.

The process of the first way is that we build a solution space tree firstly; then reduce the branches by dependent relationships, and traverse the reduced tree and output the results at last. The method is simple and easily implemented, but space requirement is big and it is infeasible for many OUTs.

Another way is using back track algorithms to obtain the results [6]. The method reduces the tree by using the dependent relationships and outputs the combinations of paths from root to leaves while it traverses the tree. The method only needs a smaller space and time.

The basic process of generating combinatorial test cases based on the tree model is as follows:

(1) Find those dependent relationships of input parameters in OUT; obtain the set of dependent relationships; and derive the smallest dependent relationships from known relative theory.

(2) Draw the dependent graph according to dependent relationships; and obtain the dependent sequence of the input parameters by dependent graph.

(3) Decide every input parameter level in solution space tree by the dependent sequence of the input parameters; and build the solution space tree.

(4) Traverse the solution space tree using back track algorithms and dependent relationships to obtain the combinatorial test cases.

The pseudo-code of the back track algorithm is shown in Figure 5. In the algorithm, one important work is program initialization, which includes input parameters symbolization, building three stacks, etc. The initializing work is as follows:

(1) Symbolize input parameters;  
(2) Represent dependent relationships with character string;

(3) Build three stacks for the nodes, the input parameters combinatorial strings and the branch starting points;

(4) Build the root node, and then fill in branch domains with breach values and fill in link domains with marks which ①: its child is sub-tree; ②: its child is null; ③: its child has already been traversed ④: its child is a leaf;

(5) Fill in the first branch value of the root node to the input parameters combinatorial strings stack and fill in the mark ③ to the corresponding link domain;

(6) Fill in the root node to the node stack and fill in the starting branch to the starting branch stack.

```

BackTrackForTestcase()
{
  Initializing;
  iLevel=1; // iLevel represents level
  while (iLevel>=0)
  {
    if (traversing down)
    {
      if (the next level node is not leaf)
      {
        building a non-leaf node;
        traversing down according to the mark in the link;
      }
      else // the node of next level is leaf
      {
        building the leaf node;
        traversing the leaf node;
        output combinatorial strings if meeting the
                                dependent relationships;
        back track to up-level node;
      }
    }
    else // back track up to
    {
      obtaining the information of the back track node;
      traversing the next branch of the node;
      if the last branch in node, traversing back to up-level;
    }
  }
}

```

Figure 5. The pseudo-code of the back track algorithm.

For the convenience of program code, with regard to the two types of dependent relationships  $1:n$  and  $n:m$ , we transform them into the types of  $1:1$  or  $n:1$ . For example, for a dependent relationship  $f(x_1 \in \{a\}) \Rightarrow x_2 \in \{d\}$  and  $x_3 \in \{g\}$ , we can transform it into two dependent relationships  $f(x_1 \in \{a\}) \Rightarrow x_2 \in \{d\}$  and  $f(x_1 \in \{a\}) \Rightarrow x_3 \in \{g\}$ . In the same way, for  $f(x_1 \in \{a\}$  and  $x_2 \in \{d\}) \Rightarrow x_3 \in \{g\}$  and  $x_4 \in \{k\}$ , we transform it into  $f(x_1 \in \{a\}$  and  $x_2 \in \{d\}) \Rightarrow x_3 \in \{g\}$  and  $f(x_1 \in \{a\}$  and  $x_2 \in \{d\}) \Rightarrow x_4 \in \{k\}$ . For  $f(x_1 \in \{a\}) \Rightarrow x_2 \in \{d, e\}$ , we transform it into  $f(x_1 \in \{a\}) \Rightarrow x_2 \in \{d\}$  and  $f(x_1 \in \{a\}) \Rightarrow x_2 \in \{e\}$ .

While we use back track algorithm for test data, a distinct characteristic is dynamically generating the solution space. In any time, the back track algorithm only saves the path from root node to the current extended node. Set the length of the path from root to leaf node is  $h(n)$ , the necessary computing space of back track is usually  $O(h(n))$ . But if we save the whole solution space tree, the EMS memory needs  $O(2h(n))$  or  $O(h(n)!)$ .

Table 1. The main input parameters and values of parameters in interface of web

parameters	value number of parameter	parameter values
gender( $G$ )	2	male, female
birthday( $B$ )	59	1931,1932,1933,...,1988,1989
marital status( $M$ )	3	unmarried, divorced, widowed
education( $E$ )	6	junior college, undergraduate, double bachelors, master, Ph.D., post-doctoral
occupation( $O$ )	24	entrepreneur, manager, white-collar, blue-collar, student, engineer, civil servant, teacher, doctor...
pay( $P$ )	6	<3k, 3k-5k, 5k-8k, 8k-12k, 12k-20k, >20k

In the worst moment, the necessary time of back track algorithm is  $O(\prod_{i=1}^n |D(x_i)|)$ . But during the program executing, due to the reducing some branches according to the dependent relationships, the actual number of traversed nodes is less largely, and the time costs less accordingly.

## 5. Experiments

### 5.1. Case study

We firstly look at one example. There is an interface of friends-making website [11], which has eight text input boxes, one Radio button of gender, nine drop-down lists(province and city, date of birthday, marital status, education degree, occupation, pay), one checkbox, one input confirm button.

We analyze dependent relationships among several main input parameters on the interface of the example, and the six of the nine radio drop-down lists are the key test input parameters. There haven been some restricts among parameters in program module codes. One of the restrictions is for years, months, and days in birthday and another is for provinces and cities. But we can still find some dependent relationships among parameters. For example,

- The dependent relationship of genders and occupations, such as, a man cannot select the occupation item of stewardess.
- The dependence relationship of occupation and pay, such as, it is impossible that a blue-collar worker has a salary of over 20000 dollars a month.
- The dependence relationship of birthday and marital status, such as a 20-year-old man is illegal when he is married in china, etc.

We can find as more many dependence relationships as possible and restrictions on the parameters in codes. When testing the interface using combinations of input parameters as test cases, we often cannot obtain some parameter combinations in the interface. Table 1 lists the main input parameters

and its values in the interface of the example website [11]. Table 2 shows the parameter relationships in the interface. The letters  $G$ ,  $B$ ,  $M$ ,  $E$ ,  $O$ , and  $P$  in Table 2 represent those input parameters, seeing the Table 1.

Table 2. The relationships among the parameters in the interface

	$G$	$B$	$M$	$E$	$O$	$P$
$G$					√	
$B$			√	√	√	
$M$		√				
$E$		√				
$O$	√	√				√
$P$					√	

### 5.2. Experiment of back track algorithm

The experimental study in this paper gives the results of several theoretical and practical cases. All tests are performed on PC with a 2.2 GHz Intel CPU and 512 Mb of memory. Computer O.S. is Windows XP and the program code is edited, compiled, and run on C++ Builder 6. Before the program running, a text file must be built in advance. The text file includes the number of input parameters, the symbolic value sets for each input parameters and the set of input parameters dependent relationships.

In the example, in order to facilitate description of test, we reduced the scope *year* value as 1961-1989. By using the dependent relationships and after running the back track algorithm program, we obtain the reduced combinatorial test suite. See the last result item in Table 3.

In the experimental study, four are practical cases and three are theoretical cases. The experimental results of these cases can be seen in Table 3.

From the data of table 3, we can see the method can reduce the combinatorial test cases in varying degrees. The minimum and maximum of the input parameter combinations are 256 and 150336 and the ratios of the test case numbers in reduced suite to the numbers of  $n$ -wise input parameter combinations in different subjects are from 13.6% to 44.3%. The running time of the algorithm program is from 0.01s to 17s.

Table 3. The running results of the back track algorithm program

No.	Subject	Number of dependent relationships	Number of input parameter combinations (A)	Number of test cases in reduced suite (R)	R/A (%)	Running time (s)
1	4 <sup>4</sup>	6	256	88	34.4	0.01
2	4 <sup>4</sup>	7	256	40	15.6	0.01
3	5 <sup>5</sup>	10	3125	425	13.6	0.1
4	5 <sup>6</sup>	11	15625	2750	17.6	1
5	6 <sup>6</sup>	11	46656	7776	16.6	3
6	3 <sup>2</sup> *4 <sup>3</sup> *2 <sup>4</sup>	15	9216	1413	15.3	0.1
7	2 <sup>1</sup> *29 <sup>1</sup> *3 <sup>1</sup> *6 <sup>2</sup> *24 <sup>1</sup>	21	150336	66704	44.3	17

(Note the col. *Subject* in Table 3.  $m^n$ :  $n$ -the number of input parameters,  $m$ -the number of input parameter values;  $m^l * q^p$ :  $n+p$ -the number of input parameters; and the number of  $n$  input parameters is  $m$  and the number of  $p$  input parameters is  $q$ .)

## 6. Conclusions

This paper proposes a method to reduce the black-box test suite. In general, the reducing effect of our method is good for some OUTs. The computing space of algorithm in our method is usually small, and the algorithm running time is acceptable.

However, the number of input parameter combinations is often enormous, and sometimes the number of test cases is still unacceptable even after reducing. How to test such OUTs will be a big challenge. Therefore, it is undeniable that using our method to reduce  $n$ -wise combinatorial test suite has still had some limitations for some OUTs.

For improving our method, our future research work includes how to obtain the dependent relationships of input parameters from program dependence graph or documents and how to build a solution space tree to make the method in this paper more effective.

## 7. References

- [1] R. Mandl, "Orthogonal Latin squares: An application of experimental design to compiler testing", *Communications of the ACM*, 28(10), NY, USA, Oct. 1985, pp. 1054-1058.
- [2] D. M. Cohen, S. R. Dalal, M. L. Fredman, G. C. Patton, "The AETG system: an approach to testing based on combinatorial design", *IEEE Transactions on Software Engineering*. 23(7), IEEE Computer Society, Washington, DC, USA, 1997, pp. 437-444.
- [3] T. Shiba, T. Tsuchiya, T. Kikuno, "Using artificial life techniques to generate test cases for combinatorial testing", *Proceedings of the 28th International Computer Software and Applications Conference (COMPSAC'2004)*, Hong Kong, China, 2004, pp. 72-78.
- [4] R. Kuhn, Y. Lei, R. Kacker, "Practical combinatorial testing: beyond pair-wise", *IT Professional*, vol. 10, no. 3, IEEE Computer Society, Washington, DC, USA, May/June 2008, pp. 19-23.
- [5] P. J. Schroeder, B. Korel, "Black-Box Test Reduction Using Input-Output Analysis", *Proc. of the 2000 ACM SIGSOFT International Symposium on Software testing and Analysis*, Portland, Oregon, United States, 2000, pp. 173-177.
- [6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms*, 2nd edition, The MIT Press, Massachusetts, USA., September 1, 2001.
- [7] P. Saraph, M. Last, A. Kandel, "Test Case Generation and Reduction by Automated Input-Output Analysis", *Proceedings of 2003 IEEE International Conference on Systems, Man & Cybernetics (SMC 2003)*, Washington, D.C., USA, 2003, pp. 5-8.
- [8] C. Cheng, A. Dumitrescu, P. Schroeder, "Generating Small Combinatorial Test Suites to Cover Input-Output Relationships", *Proceedings of the Third International Conference on Quality Software (QSIC'03)*, IEEE Computer Society, Washington, DC, USA, 2003, pp. 76-82.
- [9] A. Silberschatz, H. F. Korth, S. Sudarshan, "Database System Concepts (Fifth Edition)", The McGraw-Hill press, May 17, 2005.
- [10] M. Main, "Data Structures and Other Objects Using Java (3rd ed)", The Pearson Addison-Wesley press, October 14, 2005.
- [11] [www.juedui100.com/register/register1.jsp](http://www.juedui100.com/register/register1.jsp).

# Introducing Automated Environment Configuration Testing in an Industrial Setting

Caryna Pinheiro  
MCIT Solutions Inc.  
Calgary, Canada  
[caryna.pinheiro@gmail.com](mailto:caryna.pinheiro@gmail.com)

Vahid Garousi\*, Frank Maurer+, Jonathan Sillito+  
\*: Department of Electrical and Computer Engineering  
+: Department of Computer Science  
University of Calgary  
[vgarousi,frank.maurer,sillito}@ucalgary.ca](mailto:{vgarousi,frank.maurer,sillito}@ucalgary.ca)

**Abstract**— This paper presents an automated environment configuration testing strategy developed as part of an action research project to deal with issues of staging environment instability in a large organization. We demonstrate how a suite of automated environment configuration tests provided an unobtrusive way to verify the hospitability of staging environments, decreased the time wasted on manual troubleshooting of environmental issues, and consolidated software configuration management information. The test strategy was also greatly welcomed by upper-level management and is now being expanded to other parts of the organization.

**Keywords** – *Environment configuration, Software configuration management, staging, software development lifecycle, test automation, test strategy, action research.*

## I. INTRODUCTION

Modern IT ecosystems have a complex combination of hardware, software, middleware, components, applications, organizational culture, practices, and application lifecycles [1]. Staging environments are part of the release lifecycle of many complex software systems under development. They are used to assemble, test and review new versions of a software system before it is deployed in the field. During staging, software applications are tested in “production-like” server environments that integrate code from all existing applications in one organization [2, 3].

During deployment to staging environments, developers start to find out about the configurations needed in order to successfully run the deployed application. Dependencies to other software applications and infrastructural components are also identified. In this paper, we define software environment configurations as:

- Settings that must be applied to servers
- Application-related settings (e.g., access permission) stored in files or databases
- Third party shared services and components.

Deployment to staging environments – and later to the production environment – can be either manual or automated. In complex IT ecosystems, it is hard to successfully automate deployments to encompass all of the other dependencies on external components, systems from other project teams, and infrastructural configurations [1].

In large organizations that have multiple concurrent software projects, it is common to have project teams with different release schedules sharing the same physical resources for their staging environments. It is also common

to have a combination of teams with their own automated deployment scripts and manual deployments of legacy applications [1]. Such diversity in deployment approaches and timelines often lead to instabilities and rework effort in staging environments’ configurations.

Strict vigilance of configuration changes may reduce the number and frequency of changes introduced into staging environment. But this often creates bottlenecks for the quick delivery of iterations and necessary fixes for project teams. There is also a problem of governance. The fine line between enterprise-wide configuration management of staging environments and the configuration needed by individual project teams is usually a grey one. In our experience based on several large-scale industrial projects, configuration management is often shrunken to a source control repository that includes documentation of how to set up a known baseline for an application system.

The challenges that we have faced in an industrial setting are the goals for the work reported in this paper: (1) How can we validate that the environment configurations, our software application depends on, have been correctly applied to staging environments? (2) When our application starts failing after other teams deploy to the same staging environment, how can we quickly validate that this is not due to missing or changed configurations? (3) How can we provide a proactive way to consolidate configuration management information?

In this paper, we present an action research and development project conducted to address the above issues with staging environment instability. We demonstrate how a carefully-designed automated suite of environment configuration tests can provide an unobtrusive way to verify the hospitability of staging environments, consolidate configuration information and external dependencies, and also decrease the cost associated with manual troubleshooting of environmental issues.

The remainder of this article is structured as follows. We introduce our guiding research methodology in Section II. The problem domain is discussed in Section III. The existing literature is discussed in Section IV. Our test strategy is presented in Section V. Lessons learned are shared in Section VI. We conclude this paper in Section VII.

## II. GUIDING RESEARCH METHODOLOGY

Action research has been increasingly used in the fields of information systems [4]. It looks at combining theory and practice to solve an existing issue. The work of Davison *et al.*

[5] provides a concise set of principles and guidelines for researchers looking for ways to combine theory and practice in an industrial setting using an action research method called Canonical Action Research (CAR). The CAR model has five encompassing principles: (1) the creation of a researcher-client agreement – signed in August of 2007; (2) the cyclical process model with five distinct phases: diagnosis (discussed in Section III), action planning, intervention (action taking), evaluation, and reflection (all discussed in Section V); (3) the adoption of theory; (4) the implementation of change through action; and (5) learning through reflection.

During action planning, we further investigated the issues related to environment instability. Informal question & answer sessions were set up with team members. The questions were structured to ask probing questions and concrete examples [6]. The testing techniques used during the intervention and evaluation steps are discussed on Section V.

### III. PROBLEM DOMAIN

This section describes the diagnosis phase of our action research project.

#### A. Context

The IT department of our industrial partner (based in Calgary, Alberta, Canada) has over 190 professionals. This paper studies projects from the largest IT Program in this agency with approximately 50 IT professionals. This agency’s IT focus is to develop, enhance and maintain contemporary systems to enable timely responses to requests from the oil & gas industry. In 2008, there were a total of 18 different concurrent software projects with budgets ranging from \$0.5 to \$1.5 million dollars.

The overall architecture for hosted applications is illustrated in Figure 1. Servers are clustered for redundancy. The organization’s projects are developed on a backbone of four staging environments: development (DEV), test (TST), acceptance (ACT), and production PRD. The environments are described in detail next.

Construction iterations lead to completed development activities built and delivered to the DEV environment and code ready and tested in an integrated environment (TST), which are constituted of many virtual servers. A production release includes many iterations. Once a group of iterations leads to a candidate release, the completed work must be deployed and tested in real staging (ACT), and if approved by all stakeholders, it then becomes a production release (PRD).

#### B. Staging Environment Instability

All in-house development teams share the staging servers. Software applications have to be deployed on several staging environments during different milestones of the release lifecycle. Changes to server configurations made by one team have the potential to impact other teams. This situation certainly happened very frequently in the organization under study.

On many occasions, successfully deployed applications started to fail and teams were not made aware of what actually changed in the environment. There was no formal

ownership of non-hardware environment configurations. As a result, teams had to constantly and manually verify server configurations to ensure that all different environments were set-up correctly for their applications.

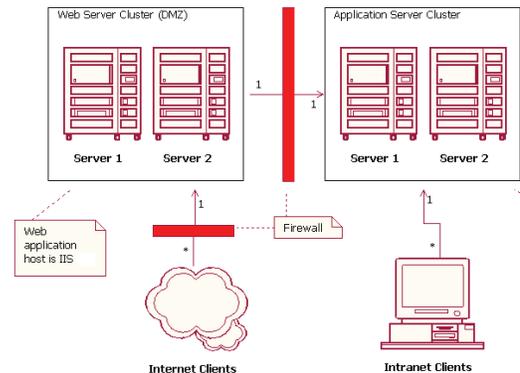


Figure 1. Overall architecture of hosted applications.

The entire situation was almost ad-hoc, caused last-minute surprises, and chaos in many occasions. “It worked for me yesterday, but I don’t know why it doesn’t work today!” This manual check of configurations was also very time consuming and frustrating. The company had looked for best practices out there to deal with the staging environment instability problem, however no good best practice was found. While using virtualized servers for each individual project would alleviate some of these instabilities, it would introduce an extra layer of complexity to test integrated systems. Existing problems with virtualization and n-tier applications using COM+ and other licensed third party tools [7] are also another barrier to many of the legacy applications in house. Virtualization also incurs a cost to performance, due to extra process boundaries to serialize data to and from.

In the presence of automated deployment scripts, one may suggest that re-deploying the application would be an easier way to solve these issues. However, automatic deployments still require core shared services (e.g., COM+, and Internet Information Services), and application pools to be shut down, introducing disruptions to other teams that also have tight schedules and cannot afford constant downtimes. While we highly believe and support automated deployments, in the context of multiple on-going software projects, they simply overwrite the existing configurations without revealing which ones were incorrectly configured or changed by other teams. This becomes an environment where “the last team to deploy wins.” And it is our experience that automated deployment scripts to not encompass all configurations needed in order to successfully host an enterprise n-tier software application. Automated deployment scripts also require expertise and knowledge to understand. The three projects under study had created semi-automated build scripts, consisting of Microsoft Installer files (msi) or simple Visual Build scripts. The word “semi-automated” is used to reflect that these scripts do not pull the source code from any source control repository. Instead these scripts copied existing compiled assemblies and files to desired locations. They also did not set server

configurations, which were left to be done manually. It is fair to say that deployment to the staging environments was mostly manual, done by a person following a set of written step-by-step instructions. In many occasions, especially after deployments, parts of the application would simply not function due to missed or changed configurations in the servers.

At a later stage, a fulltime build automation expert was hired. The build scripts were enhanced to include core environment configurations, but the support for such configuration is limited in many commercially available tools (such as Visual Build and NAnt). Although further automation of the deployment process was helpful, it could not completely address the situation as many project teams did not have the budget or expertise to build automated deployments. The bottom line is that servers were still being changed manually.

Almost all system were being developed using object-oriented design and analysis methodologies. Subsystems were created to foster reusability. But such reusability creates dependencies between external objects, services, and database objects, amongst others. Project teams become consumers of components that are not deployed and configured by them. The delivery timelines of reused components may or may not be in-line with the consumers. It becomes important to validate that dependencies are addressed in staging environments.

#### IV. LITERATURE REVIEW

Lents and Bleizeffer [1] describe an ethnographical study of eight enterprises to understand the sources of IT environment complexity in the design of middleware software. They focused on organizations, user roles, and technologies involved in the life-cycle of applications. The authors found that software deployments differ depending on the environment being targeted. *“Deployment into unit test environments is usually informal while deployment into production is a highly rigorous process.”*

These authors [1] found that although deployment was typically done in an automated fashion, most enterprises used ad-hoc methods for deploying small changes. Deployment responsibilities also shift to the hands of external groups (such as change control) for higher staging environments. A point of complexity identified was the supporting software that surrounds software applications. Dependencies on databases, transaction and messaging servers, clients, and source control must be integrated well with applications. The authors did not provide suggestions on how to eliminate this bottleneck; instead they focused on design considerations to minimize complexity.

Ambler talks about the use of “sandboxes” as being an Agile best practice [8]. Sandboxes are integration staging environments that become more controlled as they move up in the release lifecycle. But still it is easy to automatically detect the issues when they occur using sandboxes.

Beck, the original author of eXtreme Programming (XP), advocates that software needs to be integrated and tested early and often [9]. Continuous integration paired with a suite of

NUnit tests are used to achieve this goal. Eckstein [11] states that each Agile team must have someone capable and responsible for integration and configuration management. Suggestions to support configuration management include source control versioning and baselining with continuous integration tools. But as we mentioned in Section III, automation alone does not resolve the underlying issue in contexts of multiple n-tier integrated project teams with different schedules and deployment practices.

A taxonomy of existing SCM systems by Conradi and Westfechtel [5] focus on the inner processes of source control versioning models. To the best of our knowledge, we were unable to find any related literature that addresses the automated verification of staging environment configuration as a way to consolidate configurations.

#### V. TEST STRATEGY: ENVIRONMENT CONFIGURATION TESTING

##### A. Action Planning

To address our first goal - (1) how can we validate that the environment configurations our software application depends on have been correctly applied to staging environments - we designed a test strategy that ensures certain properties of the deployment environments. Our automated test strategy has the following aspects: (1) definition of a meaningful and effective test adequacy criterion, (2) test oracle, (3) choice of test tool, and (4) an error seeding (mutation) technique.

**Test adequacy criterion:** Well-known black-box or white-box test adequacy criteria (e.g., line or decision coverage) do not fit the non-functional nature of our problem domain (i.e., staging instabilities). Thus, a fault-based testing criterion proved to be the best fit. A set of most common faults were identified by the interviewed developers and the lead tester. As a result, the initial test suite needed to cover:

- Folder permissions
- Database (SQL Server) stored procedures permissions
- Availability of required services
- COM+ DLL registration and identification
- Internet Information Services (IIS) settings
- Network groups, machine users, and machine groups
- Database (SQL server) users and roles memberships.

**Test Oracle:** Before our involvement, all teams were required to maintain an up-to-date “Disaster Recovery Plan System Manual” (DRP). This document was one of the few documents kept relatively up-to-date in source control due to auditing requirements.

This manual provides a high-level explanation of how to configure each individual application. The DRP did not list some required configurations, e.g., required database permissions, and folder permissions. Outdated or missing configurations were often gathered from deployed server configurations. External dependencies, such as services, stored procedures, and database users were gathered through code inspections.

**Chosen test automation tool:** NUnit version 2.2.6 was chosen as the test tool. NUnit was already an approved tool

for testing in the company, which meant that a long bureaucratic approval process from the technical enterprise team could be avoided. It was important to follow the path of least resistance due to existing contextual antipathy of management towards test automation.

**Error seeding (mutation):** We planned to manually inject defects into the configurations (during mutation testing only) to assess the fault detection effectiveness of the test suite. For example, deleting a folder permission, or stopping a service. The approach and results are presented in Subsection C.

### B. Action Taking

In this paper, we refer to test fixtures as parts of the code needed in order to run the four phases of a test case in the NUnit framework (i.e., set up, exercise, verify, and tear down). Test cases were grouped into ten test fixtures based on the type of configuration they were trying to validate: (1) folder permissions, (2) database stored procedures permissions, (3) services availability, (4) COM+, (5) universal data links, (6) web configuration files, (7) IIS settings, (8) network and machine user membership, (9) global assembly cache, and (10) database security. These test fixtures were coded with generic *private* functions that could “exercise” a predefined behavior. For example, check if a folder has been granted a determined set of permissions. This generic function takes as input a folder path, and the permission set.

Test cases executing a call to these generic functions were passing in specific test inputs and performing assertions to “verify” that the return values match the expected values. Test inputs and expected return values were declaratively specified in XML files loaded during the test “set up.”

Configurations are environment specific. They refer to different server paths, different domain accounts, and different namespaces for components and services. Before running, a test case reads an environment ID. Test inputs and expected values (oracle) are grouped by environment ID in each test-fixture-specific XML file. A “helper” class was created to assist in reading these declarative test inputs (making extensive use of the XPath, XML Path Language). This provided an easy way to point the test suite to the desirable environment.

As an example, a folder used for downloading attachments in a web application is given a tag (File name=TAG). This folder has a *path* in the Development staging environment (DEV) and it needs *modify permissions* granted to a network user called *userDEV*. The folder path acts as input while the permission and user pair are the expected values (oracle) for the test case execution. A folder that serves the same purpose in the Production environment (PROD) is given the same tag (File name=TAG). In PROD, this folder has a different *path* and it needs *modify permissions* granted to a different network user called *userPROD*.

Without the declarative XML inputs and expected values, we would need to code one test cases per staging environment (in our context at least five) to test that such

download folder (TAG) has the correct permissions. Functionally speaking, the code executed to check the folder permission is the same for different test cases (a download folder, a temporary folder). We simply need a different set of test input/oracle for the different folders and different staging environments. This declarative XML approach also allowed us to create a living catalog of the most important environment configurations needed to run our applications – which addressed our third research goal – (3) a way to consolidate configuration management information.

Figure 2 shows an example of the XML input (sensitive data has been either replaced by “xxx”). The *expectedValue* node contains the expected result for the test case execution and is used for assertions in NUnit.

### C. Preliminary Evaluation

After the test suites were implemented, we manually changed (mutated) the environment and checked if our tests would find all changes. Table II illustrates a subset of the mutants introduced and the results of the mutation testing. Our test cases found (killed) all the mutants (seeded faults) for every execution.

```
<?xml version="1.0" encoding="utf-8"?>
<TestConfiguration>
  <Name>Verify File Permission</Name>
  <Description><![CDATA[Verifies that the file
permission matches the expected value.]]>
</Description>
  <Environment name="DEV" >
    <Server name="xxx">
      <BuildPackage name="xxx">
        <File name="TAG">
          <Path>\\xxx</Path>
          <expectedValue>
            <Type>FilePermission</Type>
            <Value>Modify</Value>
            <AccountName>userDEV</AccountName>
            <Check>True</Check>
          </expectedValue>
        </File>
      </BuildPackage>
    </Server>
  </Environment>
  <Environment name="PROD" >
    <Server name="xxx">
      <BuildPackage name="xxx">
        <File name="TAG">
          <Path>\\xxx</Path>
          <expectedValue>
            <Type>FilePermission</Type>
            <Value>Modify</Value>
            <AccountName>userPROD</AccountName>
            <Check>True</Check>
          </expectedValue>
        </File>
      </BuildPackage>
    </Server>
  </Environment>
```

Figure 2 - Example of the XML test case inputs and expected output.

Table II. Example of mutants

Category	Mutant Generation (Error seeding)
Folder Permissions	Using windows explorer, we removed or changed permissions.
SQL Server stored procedures permissions	Using Microsoft SQL Server Management Studio, we removed permissions.
Availability of required services	Using Microsoft Computer Management Console, we stopped, disabled, or paused the services.
COM+ DLL registration and identification	Using Microsoft Component Services, we deleted or disabled the components. Also we changed the identities being tested.

After our mutation evaluation, we executed our test suite against the development staging environment (the most instable of the staging environments). The test suite successfully found at least half dozen missing permissions for stored procedures and file system folders in the first day of use. Such discovery would have taken many man-hours to diagnosed if done manually, and caused many hours of down time to the systems that needed these permissions. In

summary, the test suite proved to be effective in detecting both seeded and real environment configuration issues.

#### D. Periodic Execution of the Automated Tests

It is important that tests are run frequently so that errors are caught in a timely fashion. It is also important that test execution be made visible to appropriate stakeholders. The lack of frequency and visibility seems as one of the main root causes for test automation failures [10, 11]. Beck suggests that someone in the team must be responsible for executing tests frequently and for publishing the results to all team members [9]. In order to address our second goal - (2) if our application starts failing after other teams deploy to the same staging environment, how can we quickly validate that this is not due to missing or changed configurations – and to address the above concerns, we decided to automate the execution of the test suites on a scheduled basis, to automatically publish the results in the intranet and to proactively send emails to interested parties after each test execution.

NUnit provides console commands to execute test suites and to export test run results into XML files. A batch file was created to call the environment configuration test suite. Microsoft Task Scheduler was used to execute this batch file at 6:00 AM.

To provide an easier, more readable presentation of the results, an XSL style sheet was generated to display the results in an “Environment Forecast” webpage.

A “sunny” forecast indicates that all tests are passing. A “mostly sunny” with occasional clouds forecast indicates that less than 25% are failing. A “cloudy” forecast indicates that 25-50% are failing. Finally, a “stormy” weather is displayed if more than 50% of the tests are failing. To provide further information, data logged during tests (such as the parameters being tested) is exported to a text file. A link to this detailed log file is displayed at the bottom of the “Environment Forecast” web page. Once the test run was completed, e-mails were sent with a link to the results webpage to interested individuals (project lead, testing lead, and developers).

At first, we only had access to run the tests against the development and testing staging environments. We executed these tests for a period of 6 weeks. The test results provided improvements to the following areas:

- Database security: many stored procedures had the incorrect level of permission. This was brought to the attention of the Enterprise Architecture team which was asked to revisit the security models and tighten up permissions.
- Network Folder security: many folders had more permissions than necessary in the development and test environments, which resulted in certain problems to only appear in higher staging environments, where security was more strictly enforced.
- Visibility into changes in database administrators: stored procedures permissions seemed to “go missing” every once in a while, and system roles get regrouped without teams being properly notified.

The long term affect of the configuration tests is discussed in the following section.

#### E. Technical Challenges And Impact

After seeing the benefits of the testing suites on the lower level staging environments, we inquired about scheduling them to run against higher staging environments. The team responsible for granting such permissions (enterprise testing) was at first resistant to give any special permissions to run these tests in higher staging environments.

Because NUnit was used as the test harness tool, the enterprise testing team first assumed the tests were developer unit tests, thus unfit to be on-going in more secured environments. To clarify this misconception, and to get their support going forward, we booked meetings with the enterprise team, inviting the management group as well to present the test suite and automation strategy. Members of that team were also included in the daily test-run e-mail notifications.

After the presentation, managers and the enterprise testing team became interested to see the source code. Access to the source code was granted to them. Soon after the demos, this enterprise team assigned a developer to validate the test suite in order to give the permission to run it in higher staging environments. During this review, the developer abstracted the XML declarative input approach and adapted our test suite to be able to run tests from multiple projects. We gained approval to run the tests in higher staging environments while the enterprise architectural team enforced that all other in-house projects specify their environment configurations in XML to also run in this test suite. These tests are now scheduled to run three times a day in all-staging environments, including Production. They can also be run on-demand by a group of users that have been granted special permissions, including the first author of this article.

A verbatim quote from an e-mail sent by the test lead states that **“before the configuration tests:** *One or more testing environments would be unstable each day causing delays in development and testing. During a period of 3 months, environment outages were tracked and over 50 hours of development and testing time were lost and business confidence in the environments/systems was very low. In addition, identifying and addressing the cause of the outage required 2 to 5 people to get involved to search for and address the issue. The primary root cause was very difficult to determine.* **After configuration test:** *The suite initially focused on monitoring and “flagging” environment configuration changes. The tests were executed on a daily basis and on command. The impact of these tests is significant. Problem root cause identification is immediately available and resolution activities are focused (1 - 2 people) and usually are corrected in minutes. It was identified that 97% of all instability problems were associated with environment configuration and database issues/changes (not system code/functionality). Test environment downtime has been reduced to 0 - 10 minutes per week. Most importantly, overall organization confidence in the test environments has*

significantly increased - to the point where development and test results are trusted and used for benchmarking purposes.”

## VI. LESSONS-LEARNED

NUnit proved to be an effective, flexible and easy to use testing harness for the execution of tests other than traditional unit tests.

By running environment configuration tests on an ongoing basis, the root causes of environment instability and configuration problem areas started to disappear. Conflicting configurations were found during test executions. Dependencies on external components, such as services and databases were validated without manual intervention. Teams had to work together to ensure suitable resolutions to conflicts. Manual deployments still caused issues, but these issues were found in a timely manner, making individuals involved in manual deployments more cautious of changes to shared configurations.

The environment configuration necessary to run applications is no longer hidden in long documents in source control, or in deployment scripts. They are visible and readily available in a suite of automated tests. To provide quantitative insights to the usefulness of the automated build verification testing (BVT), some before- and after-BVT measures are provided below.

Before BVT	After BVT
<ul style="list-style-type: none"> <li>• Over 50 hours of downtime in 3 months</li> <li>• Very low confidence</li> <li>• Outage required 2 to 5 people</li> <li>• Primary root cause was very difficult to determine.</li> </ul>	<ul style="list-style-type: none"> <li>• 0 - 10 minutes of downtime per week</li> <li>• 97% of all instability problems were associated with environment configuration</li> <li>• Focused (1 - 2 people) and corrected in minutes</li> <li>• Problem root cause identification is immediately available.</li> </ul>

Last but not least, based on our experience, we make the following recommendations for testing practitioners wanting to introduce configuration testing practices:

*Expect resistance.* The truth is that automated testing requires special permissions for higher level staging environments. So admit it, get it working on a local environment where you do have enough security privileges and plan to do some convincing for getting appropriate privileges on higher staging environments. Don't take this resistance personally. But instead, nicely present the benefits of your tests to those being skeptical. They will see that it will actually help them do their job better too.

*Keep maintainability in mind.* This is one of the causes of why automated tests get abandoned by many teams [9]. Make it easy to add new test cases to your suite without having to change or redeploy code.

*Keep it simple and use free tools.* Create a simple testing strategy that leans on existing free test tools, creating generic test drivers that use declarative descriptions as test case inputs for future re-use.

*Automate as much as possible.* In addition to test automation, automate the test execution as well preferably

using existing OS tools, such as Microsoft Task Scheduler and expose the test execution results in an easy to access location, such as a simple website.

## VII. CONCLUSION

The introduction of a suite of automated environment configuration tests that verify the environment on demand has helped us identify many deployment dependencies. It has also assisted several of our partner teams resolve configuration issues in staging environments in a timely manner. It also abstracted environment configuration management into a live and evolving test suite that shows failure and it is easily maintainable. The creation of a simple test strategy that leaned on existing free test tools, with generic test drivers that use declarative descriptions as test case inputs enabled future re-use. This reuse escalated the use of this test strategy to an enterprise testing framework, proving the need and usefulness of the techniques we implemented.

## ACKNOWLEDGEMENTS

The authors would like to thank Jim King and Bryan Schultz for their support.

## REFERENCES

- [1] J. L. Lentz and T. M. Bleizeffer, "IT ecosystems: evolved complexity and unintelligent design," *Proc. of 2007 Symposium on Computer Human Interaction for the Management of information Technology*, 2007.
- [2] Disruptive Library Technology Jester, "Traditional Development/Integration/Staging/Production Practice for Software Development," *On-line: <http://dltj.org/article/software-development-practice>*, Downloaded on Sept. 2009.
- [3] MSDN, "What is the Staging Environment?," *On-line: <http://msdn.microsoft.com/en-us/library/ms942990.aspx>*, Downloaded on Sept. 2009.
- [4] H. D. Frederiksen and L. Mathiassen, "A Contextual Approach to Improving Software Metric Practices," *IEEE Transactions on Engineering Management*, vol. 55, no. 4, pp. 602–616, 2008.
- [5] R. M. Davison, M. G. Martinsons, and N. Koch, "Principles of Canonical Action Research," *Information Systems Journal*, vol. 14, no. 1, pp. 65–89, 2004.
- [6] The question man, *On-line: [http://www.ajr.org/article\\_printable.asp?id=676](http://www.ajr.org/article_printable.asp?id=676)*, Downloaded: July 2009.
- [7] Microsoft App-V team blog, "On-line: <http://blogs.technet.com/softgrid/archive/2007/09/27/list-of-applications-that-can-be-virtualized.aspx>," Downloaded: March 21, 2010.
- [8] S. Ambler, "Development Sandboxes: An Agile "Best Practice" " *On-line <http://www.agiledata.org/essays/sandboxes.html>*, Downloaded on Sept. 2009.
- [9] K. Beck, *Extreme Programming*: Addison-Wesley 2004.
- [10] S. Berner, R. Weber, and R. K. Keller, "Observations and lessons learned from automated testing," *Proc of the Int Conference on Software Engineering*, pp. 571–579, 2005.
- [11] M. Fewster and D. Graham, *Software Test Automation*: ACM Press, 1999.

# An Ontology-based Software Test Generation Framework

Valeh H. Nasser<sup>1</sup> and Weichang Du<sup>2</sup> and Dawn MacIsaac<sup>2</sup>

Department of Computer Science, University of Calgary, Calgary, AB, Canada<sup>1</sup>  
Faculty of Computer Science, University of New Brunswick, Fredericton, NB, Canada<sup>2</sup>  
E-mail: v.hnasser@ucalgary.ca, {wdu, dmac}@unb.ca

## Abstract

*In automated test generation, granting control to test experts over test selection can enhance quality of generated test suites. However, in many cases test experts' control is limited and they can not define custom coverage criteria. This work proposes a general framework for application of knowledge engineering to software testing, which facilitates specification of custom coverage criteria and provides means for generating test cases from different artifacts in dissimilar domains.*

## 1. Introduction

In automated test generation a test suite is automatically generated based on a software artifact (e.g. code, design diagrams, or requirements). A crucial issue in automated test generation is test selection, which is about what tests should be included in the test suite for a desired quality to be achieved [11]. One approach to address this issue is allowing test experts select test cases manually. While this approach may be required in some cases, it is not efficient [6]. A second approach is automatically selecting test cases to satisfy some *pre-defined* coverage criteria [9]. With this method, test experts do not have much control over the test suite that is being generated. A third method which can lend more control to test experts over the generated test suite, is to leverage specification of *custom* coverage criteria. The test cases are then generated based on the coverage criteria specifications, which can be domain or system specific [7].

Benz [4] demonstrates that utilization of domain-specific coverage criteria which are based on error-prone aspects of software can enhance the quality of a generated test suite. According to Bach [2] error-prone aspects, also used in *Risk Based Testing* [2], are software elements that are likely to produce an error and can be: (1) domain specific (such as concurrent

methods, database replications, network connection), (2) based on general test guidelines and experience (such as boundary values), or (3) system specific and revealed in interactions of testers with developers and designers (such as use of an unreliable library). In addition to utilization of test experts mental model about error-prone aspect of the system, Paradkar [10] suggests that a test case generation method is required that supports specification of arbitrary test cases, implementation knowledge, invariants on model elements, and distinguished states.

However, in many cases test case selection algorithms are tightly coupled with coverage criteria and software artifacts (such as [9]). This inhibits definition of custom coverage criteria for a specific domain or system.

In order to increase control of test experts over test selection, in the past few years, we have been extensively involved in the design of testing techniques that benefit from knowledge engineering techniques. For instance, we have previously proposed a knowledge-based system architecture that generates unit tests from UML state machines [7]. This system externalizes knowledge about what needs to be tested in the form of ontologies and rules based on which reasoning algorithms are used for the generation of high-level description of tests. Also, an ontology based description of the generated test suite is maintained in order to enable test redundancy checking using standard reasoning mechanisms. In the current paper, we have tried to capture the experiences that we have gained in the design of knowledge-based testing techniques in a generalized ontology-based test case generation framework. Our proposed general framework provides means for generating test cases from different artifacts in dissimilar domains.

The rest of this paper is organized as follows. Section 2 provides an overview of the framework. Section 3 and 4, respectively describe specifications that are used in the method and phases of test generation from the

specifications. Section 5 further discusses the framework. Section 6 concludes the paper.

## 2. Framework Overview

The ontology based test case generation framework uses knowledge engineering techniques to decouple test selection algorithms from the knowledge that is used for test selection. The decoupling allows test experts to freely manipulate the ontology based representations of specification of what needs to be tested, without the need to modify hardcoded test selection algorithms. To do this, ontologies and rules are used to specify what needs to be tested, and reasoning algorithms are used to select high level description of test cases, which are called test objectives. The selected test objectives are then translated into abstract test cases. The abstract test cases are represented in an ontology, which allows the use of reasoning for redundancy checking. Finally, executable test cases are generated from the abstract test case ontology.

Hence, the executable test suite is generated in four phases, which are depicted in Figure 1 and described below:

**Phase 1- Test Objective Generation:** The ontology based representation of what needs to be tested which includes: behavioural model specifications, expert knowledge, and coverage criteria rules, is used to generate a set of test objectives. This phase is conducted using reasoning on the ontologies, which describe the specifications.

**Phase 2- Redundancy Checking:** For every test objective, a redundancy checking rule is used to check whether the test objective is satisfied by a test case, which is already included in an abstract test suite ontology. This phase is also conducted using reasoning on the ontologies.

**Phase 3- Abstract Test Suite Ontology Generation:** For each non-redundant test objective, an abstract test case is generated and added to the partially generated abstract test suite ontology. This phase is run using prevalent test generation methods from literature (such as graph traversal algorithms [9]).

**Phase 4- Executable Test Suite Generation:** For every abstract test case in the abstract test suite ontology, an executable test case is generated using the implementation knowledge ontology.

Phase 1 generates a set of test objectives. After phase 1 is completed, phase 2 and phase 3 are performed repeatedly to generate an abstract test suite. Then in phase 4, based on the abstract test suite, the executable test suite is generated.

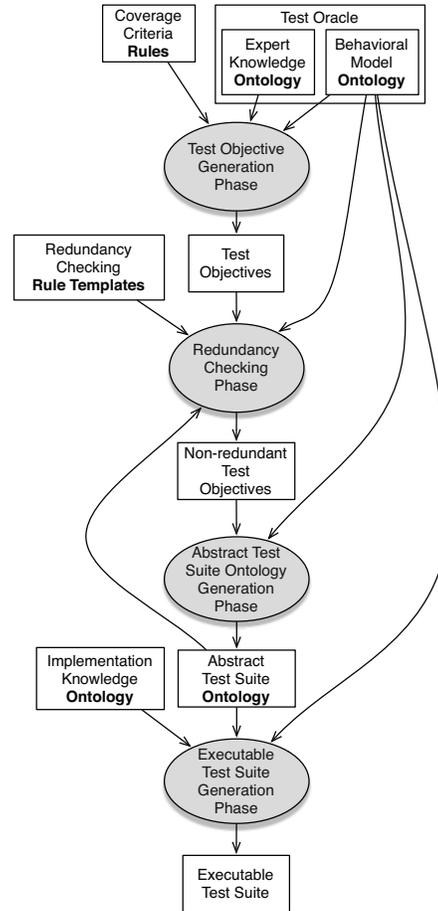


Figure 1. Framework Overview

## 3. Specifications

With this framework, in each phase some specifications are transformed into another form (Figure 1). This section describes the specifications which are used in the framework.

**Behavioural Model Ontology** The behavioural model ontology is an ontology-based representation of the software artifact based on which tests are generated. This ontology describes concepts corresponding to the software artifacts structural elements, the relationships between them, and their instances.

For example, for UML artifacts, this ontology can be created based on the the Ontology Definition Meta Model (ODM) [1] standard by the OMG, and be automatically populated from the XML Metadata Interchange (XMI) [8] representation of existing models. For code-based artifacts, this ontology can describe the concepts in a programming language (such as methods,

function call, etc.) and be automatically populated from the existing code. For GUI testing this ontology can describe the GUI elements and be automatically populated from the existing GUIs.

**Expert Knowledge Ontology** The expert knowledge ontology extends the behavioural model ontology with knowledge that is beyond the behavioural model ontology. This knowledge is exploited by the coverage criteria rules for identification of test objectives. This ontology describes test experts' mental model and is an extension point that facilitates support of various coverage criteria rules. The ontology has the advantage of retaining this knowledge, which is used by test experts.

This ontology can be designed based on error taxonomies, bug databases, and commonly used coverage criteria. Examples of pieces of knowledge that can be included in this ontology are: knowledge about use of an unreliable library, boundary values of state variables or inputs, expected exceptions, variable definition and use, concurrency relationships, and user interaction points.

**Test Objectives** A test objective delineates a test case and provides a language for test experts to define the test cases abstractly and decouples test case selection from test case generation. It consists of two parts: the predicateList and the parameterList, which are lists of predicates and parameters separated by a comma. A predicate in the predicateList has one or more parameters, which are listed respectively in the parameterList.

```
[The list of predicates separated by comma]
[The list of parameters separated by comma]
```

A test objective specifies a condition that must hold on some elements in a corresponding test case. The predicates specify the conditions. The parameters specify the elements which are defined in the behavioural model ontology or their values. A test objective specifies the structural properties of a test case. A test case can be described by combining test objectives.

Examples of test objectives for test case generation based on UML state machines are as listed below, where *Trans1*, *Trans2*, *Var1*, and *Val1* are respectively a transition, a transition, a state variable, and a value defined in the behavioural model ontology:

```
[CoverTransition] [Trans1],
[Immediate] [Trans1, Trans2],
[After] [Trans1, Trans2],
[AtTransitionStateVariableHasValue] [Trans1, Var1, Val1]
```

Similarly for code-based testing and GUI testing test objectives can be designed as listed below:

```
[CoverMethodWithInputs] [Method1, InputValue1]
```

**Coverage Criteria Rules** The coverage criteria rules are test case selection rules and can be expert-defined, system/domain-specific, or standard. In general, a rule follows the form below. The :- symbol denotes a deduction from the right side to the left side of the rule.

```
test objective :- test objective selection criteria.
```

The *test objective selection criteria* describe the conditions that should hold on some elements for them to be a part of the structure of a test case. The test objective specifies the structure of selected test cases. Coverage criteria rules refer to the vocabulary defined by the behavioural model and an expert knowledge ontology. The body of a rule specifies conditions on the parameters of the test objective, which is described in the head of a rule.

An example of a coverage criterion is shown below. In the examples, a variable name starts with a question mark.

Conditions:

- 1- A variable's domain has some boundaries,
- 2- A method uses the variable,
- 3- The variable's value is a user input.

Test Objective:

```
- Check if the method throws an OutOfBoundaryException.
```

```
coverage([ThrowException],
[?Method, ?Variable, ?Value, OutOfBoundaryException]) :-
StateVariableBoundaryValue (?Variable, ?Value),
Uses (?Method, ?Variable)
UserInput(?Variable).
```

In this example the behavioural model ontology, which can be reverse engineered from the source code, includes the knowledge that defines methods and variables and the *Uses* relationships between them. The boundaries of the variables and whether their value is a user input is described in the expert knowledge ontology. The expert knowledge ontology can be created manually or automatically from formal documents such as annotations made by programmers in a source code. Based on these knowledge bases and the coverage criteria, reasoning can be used to generate test objectives.

**Abstract Test Suite Ontology** The abstract test suite ontology, which is linked to the behavioural model ontology, describes the test suite. *Abstract* means that it can be implementation-neutral and programming-language-neutral, depending merely on the knowledge provided by a software artifact. The abstract test suite ontology consists of a set of abstract test cases. An abstract test case is specified by a list of steps, input

values at each step, and values of state variables after each step. A step corresponds to an event that changes the state of the system. The ontology-based representation of the test suite allows use of reasoning to identify redundant test cases.

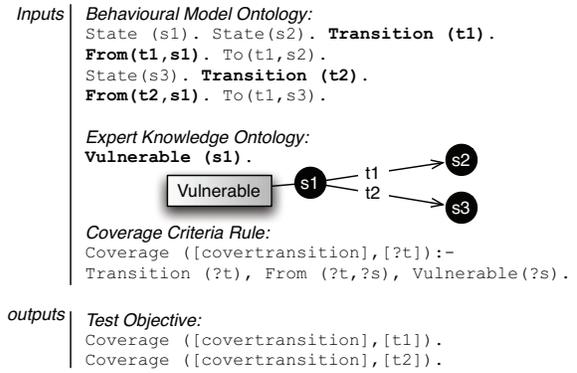
**Redundancy Checking Rule Templates** Redundancy checking rule *templates* are used to generate redundancy checking rules for test objectives. A redundancy checking rule facilitates checking whether a test objective is already satisfied by a test case, which is included in a test suite. The redundancy checking rules follow the form below:

```
The test objective is satisfied by the test suite :-
the structural characteristics of a test case that
satisfies the test objective.
```

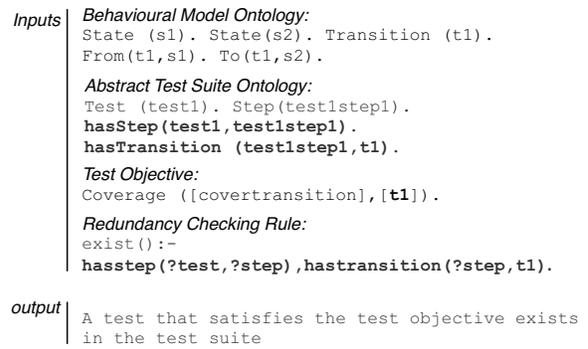
The *structural characteristics* of a test case that satisfies the test objective are described using the vocabulary defined by the abstract test suite ontology and the behavioural model ontology. For every test objective, a redundancy checking rule is generated based on redundancy checking rule template. For every test objective *predicate*, there exists a redundancy checking rule template.

**Implementation Knowledge Ontology** The implementation knowledge ontology is used for translating the abstract test suite ontology to an executable test suite. This ontology extends the behavioural model ontology with implementation information. The knowledge represented by this ontology, which is programming-language-dependent, can include: variable getters and setters, implementation names of methods, classes, namespaces, constructors, etc. This ontology can be automatically populated, if the source code is available. This ontology helps postponing the task of generation of actual executable test cases to after the implementation is done. Also it enables maintaining the same set of abstract test cases while the implementation details change.

**Executable Test Suite** An executable test suite is the main output of the system, and is the result of the abstract test suite being translated using the implementation knowledge. This test suite is written in a programming language for an implementation of the system under test. If the API of an automated testing framework such as JUnit [5] is used, the tests can be executed and verified automatically.



**Figure 2. Phase 1 Test Objective Generation**



**Figure 3. Phase 2 Redundancy Checking**

## 4. Phases by Example

This section describes the phases of test generation from the specifications using a simple partial example of unit testing based on UML state machines. Only relevant parts of the example ontologies are shown in the figures.

During the phase 1, which is test objective generation, initial test case selection is conducted using reasoning and the output, which is a very high level test suite is presented as a set of test objectives. Figure 2 illustrates the example which generates test cases based on a UML state machine. In this example test objectives that specify tests that pass transition *t1* and transition *t2* of the state machine are generated.

In phase 2, which is redundancy checking, for every test objective, reasoning is used to check a partially generated abstract test suite ontology for existence of a test case that satisfies the test objectives. Figure 3 illustrates an example, in which the test *test1* of the abstract test suite ontology satisfies the test objective, which specifies a tests that passes transition *t1*. Hence,

<i>Inputs</i>	<b>Behavioural Model Ontology:</b> State (s1). State(s3). Transition (t2). From(t2,s1). To(t2,s3). <b>Test Objective:</b> Coverage ([covertransition],[t2]).
<i>outputs</i>	<b>Abstract Test Suite Ontology:</b> Test (test2). Step(test2step1). hasStep(test2,test2step1). hasTransition (test2step1,t2).

**Figure 4. Phase 3 Abstract Test Suite Generation**

<i>Inputs</i>	<b>Behavioural Model Ontology:</b> State (s1). State(s3). Transition (t2). From(t2,s1). To(t2,s3). Event (m1). HasEvent (t2,m1). <b>Abstract Test Suite Ontology:</b> Test (test2). Step(test2step1). hasStep(test2,test2step1). hasTransition (test2step1,t2). <b>Implementation Knowledge Ontology:</b> HasImplementationMethod (m1, method1).
<i>outputs</i>	<i>//Test 2</i> ... method1(); ...

**Figure 5. Phase 4 Executable Test Suite Generation**

this test objective is discarded. Next, another redundancy checking rule is generated to examine the other test objective. The other test objective, which specifies a test that passes transition  $t2$  of the state machine, is not satisfied by the test suite, hence it is passed to Phase 3 of the method for test case generation.

Phase 3, which is abstract test suite generation uses a test case generation method from literature based on the software artifact (such as graph traversal algorithms [9] for UML state machines). The abstract test cases are generated for the given test objectives and are added to the abstract test suite ontology. Figure 4 illustrates part of an abstract test case that is generated for the given test objective and is added to the ontology.

Finally in phase 4, which is executable test suite generation, the implementation knowledge ontology is used to generate an executable test suite from the abstract test suite ontology. Figure 5 illustrates part of an executable test case which is generated for the abstract test case.

## 5. Discussion

The objective of this method is provision of a framework that can be extended by specification of custom coverage criteria rules. In order to extend the system, one can specify the corresponding coverage criteria rules in a logic programming language and populate the referred knowledge to the expert knowledge ontologies. Hence, this method is as powerful as the the rules and ontologies that are specified in it.

The method can be exploited to generate test cases based on various software artifacts such as code, design, requirements, etc. For this purpose, a behavioural model ontology representing the software artifact is used and the artifact is populated into the ontology.

Additionally, the method can be applied to different domains, such as GUI testing, concurrency testing, networking etc. To apply the method to different domains, the knowledge about error prone aspects of the domain is added to the domain-specific expert knowledge ontologies. New coverage criteria are defined based on the vocabulary defined in the behavioural model and expert knowledge ontologies. Also, the test cases can be manually specified by manually specifying test objectives.

Furthermore, rules can be defined to support standard coverage criteria from literature. Example rules for several standard state machine based coverage criteria [9, 3] are listed below:

```

- All Transition Coverage:
coverage([covertransition],[?tr]) :- transition (?tr).
- All Transition Pair Coverage:
coverage([immediate],[?t1, ?t2]) :- transition (?t1),
transition(?t2), notEqual(?t1,?t2), from(?t1,?state),
to(?t2,?state).
- Faulty Transition Pair:
coverage([immediate],[?t1, ?t2]) :- transition (?t1),
transition(?t2), faulty(?t2), notEqual(?t1,?t2),
from(?t1,?state), to(?t2,?state).

```

Redundancy checking rules for the corresponding test objectives are as follows:

```

- [covertranstions], [TR]:
exist():-
test(?t), hasStep(?t,?st1), hasTransition(?st1,TR).
- [immediate], [TR1,TR2]:
exist():-
test(?t), hasStep(?t,?st1), hasStep(?t,?st2),
hasTransition(?st1, TR1), hasTransition(?st2, TR2),
next(?st1, ?st2).

```

However, as described below, the method has some limitations:

**Non-Optimal Redundancy Checking** The redundancy checking process is not optimal. This is because the redundancy checking works backwards. That

is, it only checks whether a test case that covers the current test objective already exists in the test suite. It does not guarantee that a test objective will not be satisfied by a test case generated later. Thus the method does not necessarily generate a test suite of optimal size. The order of selection of test objectives affects the size of the generated suite.

**Time Efficiency** For larger units under test, test case generation can be slow because of the use of reasoners. The reasoning algorithms are not fast and this is the trade-off between efficiency and generality. For large systems the method may not operate interactively.

**Modelling Complexity** If the method is used with a model-based test generation methodology, the complexity of modelling software artifact is a challenge. However in presence of UML models, the behavioural model ontology can be automatically populated from the XMI representation of the existing model. Similarly, to reduce the complexity of creating expert knowledge, it may be possible to provide tools that populate it from existing documents that conveys the knowledge in another format, such as source code or custom annotations made by programmers within source code. Likewise, to deal with the complexity of creating implementation knowledge ontology, the ontology can be populated by reverse engineering existing source code.

**The Limitation of Logic Programming** Writing rules can be complicated and the expressiveness of the rule languages are limited due to the limitations of the reasoning algorithms. The constraints on the expressiveness of the rule languages limit the coverage criteria rules and redundancy checking rules that can be supported by the system. For instance, in Horn logic no universal identifier can be in the body of the rules. Hence, rules like the example below can not be expressed:

Predicate2(x) :- for all x, Predicate1(x)

This limitation can be overcome for finite domains by listing all possible values of x in a list and recursively iterating over the list to check the values of Predicate1 with them. But such an ‘extensional’ approach can not be very efficient for large domains.

## 6. Concluding Remarks

This work presents a general knowledge based test case generation framework that allows defining custom domain-specific and/or system-specific coverage criteria for various software artifacts and domains. By using custom coverage criteria, test experts can control what tests are included in generated test suites. For this

purpose, the framework uses reasoning on ontologies to address the test case selection issue.

However, further research should be done before the framework can be applied in practice. A test objective language, which is used for high level specification of test cases should be developed. Ontologies should be developed for the software artifacts and domains under test. Defining domain-specific coverage criteria and expert knowledge ontologies is not a trivial task and should be conducted in research labs. Additionally further research should be conducted on integration of the framework with existing development methodologies.

## References

- [1] Ontology definition metamodel. <http://www.omg.org/docs/ptc/07-09-09.pdf>, August 2009.
- [2] J. Bach. Risk-based Testing. *Software Testing and Quality Engineering Magazine*, 1(6), 1999.
- [3] F. Belli and A. Hollmann. Test generation and minimization with "basic" statecharts. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 718–723, 2008.
- [4] S. Benz. Combining test case generation for component and integration testing. In *Proceedings of the 3rd international workshop on Advances in model-based testing*, pages 23–33, 2007.
- [5] E. Gamma and K. Beck. JUnit. At <http://www.junit.org>.
- [6] A. Howe, A. Mayrhauser, and R. Mraz. Test Case Generation as an AI Planning Problem. *Automated Software Engineering*, 4(1):77–106, 1997.
- [7] V. H. Nasser, W. Du, and D. MacIsaac. Knowledge-based software test generation. In *Proceedings of the 21st International Conference on Software Engineering & Knowledge Engineering (SEKE'2009)*, pages 312–317, 2009.
- [8] Object Management Group. XML Metadata Interchange (XMI) specification. <http://www.omg.org/technology/documents/formal/xmi.htm>, 2007.
- [9] J. Offutt and A. Abdurazik. Generating tests from UML specifications. In *UML'99 - The Unified Modeling Language. Beyond the Standard*. Springer, 1999.
- [10] A. Paradkar. A quest for appropriate software fault models: Case studies on fault detection effectiveness of model-based test generation techniques. *Information and Software Technology*, 48(10):949–959, 2006.
- [11] H. Zhu, P. Hall, and J. May. Software unit test coverage and adequacy. *ACM Computing Surveys (CSUR)*, 29(4):366–427, 1997.

# Automated Integration Testing and Verification of a Secured SOA Infrastructure - an Experience Report in eHealth

Mario Bernhart, Thomas Artner, Andreas Mauczka and Thomas Grechenig  
Research Group for Industrial Software  
Vienna University of Technology  
Karlsplatz 13, 1040 Vienna, Austria  
{mario.bernhart, thomas.artner, andreas.mauczka, thomas.grechenig}@inso.tuwien.ac.at

**Abstract**—This work presents an experience report on the integration testing of a secured SOA with many independent service providers in the field of eHealth. The services itself are closed-source to the integrators and so the verification is message-based only. Due to strict security requirements, all services are limited to use SSL-encrypted connections. This makes it challenging to consume the messages even during testing and therefore to verify the correct behaviour of each service. Three strategies are presented, each showing different benefits and drawbacks to the integration testing process. In this report one most applicable approach was further implemented and discussed.

**Keywords**-Integration Test; Secured Infrastructure; eHealth

## I. INTRODUCTION

Service Oriented Architecture (SOA) is an established software integration paradigm for building systems out of independent components and to provide technical interoperability. Service providers develop their software components or subsystems and expose interfaces to them as web services. This new paradigm of service development, testing and integration imposes great challenges to the quality of the services. Since an application system usually involves services from different organizations, to guarantee that these services can collaborate correctly and seamlessly, the quality of services is extremely important. However, services are usually developed independently by service providers. Moreover, due to privacy concerns - e.g. in the domain of eHealth - the implementation of a service is usually invisible to service integrators. This may negatively affect the testability of such a system and, specifically for this work, constrain the verification methods to the message level.

This work presents an experience report on the integration testing of a secured SOA with many independent service providers in the field of eHealth. The services itself are closed-source to the integrators and so the verification is message-based only. Due to strict security requirements, all services are limited to use SSL-encrypted connections. This makes it challenging to consume the messages even during testing and therefore to verify the correct behaviour of each service. In the case of an failure it is essential to identify which service in the service-chain has is responsible (e.g.

send an incorrect response). A secured environment makes it difficult to verify the messages as - per definition - the messages should not be accessible by external entities. A typical test environment would then be able to test end-to-end, but not be able to verify intermediate messages. To enable the verification of messages sent between each service of the secured SOA, three approaches are presented and discussed in this paper: **man-in-the-middle proxy**: In this approach a proxy service is placed between two real services. **on-the-fly decryption**: In this approach the encrypted data will be recorded and decrypted for verification. **temporary testing interface**: For this approach each service has to implement a non-encrypted testing interface which then could be used during the integration tests. One of the presented methods (man-in-the-middle proxy) was then chosen to be implemented during the integration tests of a large European nationwide eHealth infrastructure. The rationale behind the selection of the method and the implementation are further described.

## II. RELATED WORK

Many authors such as Katt et al. in [7] and Haux et al. in [4] discuss the problems of integrating eHealth systems. Privacy and data protection in an heterogenous environment are among of the most discussed aspects. Additionally to the specific needs in the eHealth domain, software intensive systems in general still face security issues like incorrect data verification of the systems [5]. Efforts for end-to-end security under similar conditions have been undertaken in [16] by Wozak et al. Balasubramaniam et al. in [1] explain the challenges of SOA testing. Among others security is one of the major aspects that makes SOA testing a complex task. Rehman et al. in [6] points out that strong security mechanisms inhibit the feasibility of testing in general. Both strongly relate to this experience report, since the preferred method of verification (on-the-fly decryption) had to be discarded due to this reason. Ribarov in [13] discusses testing in the context of SOA and points out that integration testing should be done in an iterative manner. In expectation of faults emerging during integration testing this is essential to isolate errors. Vincent et al. concludes that in integration testing all components should be tested and integrated through their specified interface

[15]. Baresi et al. in [2] discuss many aspects of SOA testing including reliability, security, and trust. Tsai et al. [14] discuss the aspect of testability in a SOA environment and propose some evaluation criteria.

[12] and [11] support that end-to-end black-box tests are not sufficient in most integration testing scenarios. Especially this leads to the problem that multiple faults might cancel each other out. Such failures might never be found in a complex SOA without verifying intermediate messages.

Martin, Basu and Xie [9] present a framework for automated software tests in SOA environments. This work focuses on robustness tests, but the architecture of the testframework is typical for a SOA test.

### III. ENVIRONMENT AND TEST STRATEGY

This experience report is based on the work of the authors when setting up an incremental integration test of a secured SOA environment. This environment is an European nationwide eHealth infrastructure which is based on common webservice technologies. Figure 1 shows the main services and their communication paths. All communication is based on SOAP<sup>1</sup> messages. The Service Broker, the Log Service and the Service Registry are single instance components. The provided services on the right are about 30 instances, each instance provided by a different service provider. The Connector is a client-side proxy service that acts as a secure gateway to the backend infrastructure. In this integration scenario there were 5 to 10 different connector providers and up to 10.000 instances. Due to this complexity the main objectives in testing are interoperability, security and performance. The testing is done iteratively in three integrations steps. Step 1 and 2 are performed as an automated test whereas the final iteration - including all components - is performed manually. Figure 1 shows the included services for each integration step. In a preceding step (not shown in the figure) the single instance components are integrated and tested. The backend services are simulated and the test-driver takes the role of the Connector. In Step 1 all the backend services were integrated with the previously integrated single instance services. The simulation of is replaced by the actual services. In the following step 2 the Connector service is added to the test setup. The testdriver then changes its role from being a Connector to being a Client. In the final step 3 the testdriver is replaced by a real Client. The Client could be tested automatically e.g. through GUI-automation, but in this case the last integration step is tested manually.

Each client runs on a dedicated device with its applications and has its own connector. The connector is a trusted device on client side. It builds up a VPN tunnel to the secured SOA. All connections are SSL encrypted and all messages are digitally signed. The service broker receives the client requests and validates the signatures. Further, for some service calls the broker hides the clients identity from the backend services to avoid e.g. usage mining in health insurance agencies. The

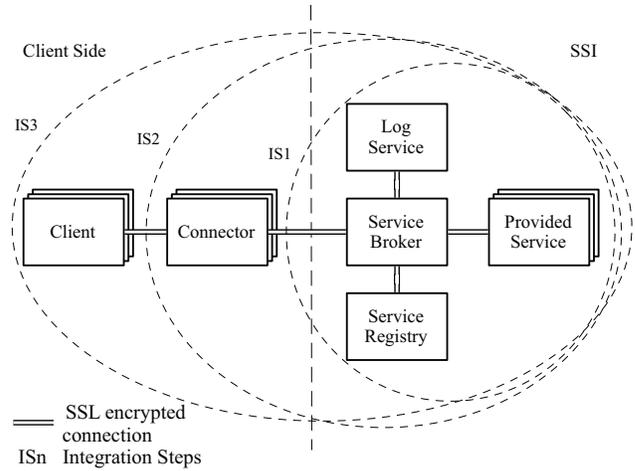


Fig. 1. Overview of the Infrastructure and Integration Steps

service broker is also responsible for the transaction log. All transactions are logged by the service broker to the log service. All of the provided services use the same service contract. Their interface to the secured SOA is identical. The messages logged by the log Service are not accessible. For security reasons access is only allowed in case of failures. The messages between the service broker and the provided services are also digitally signed and transported via SSL connections. The service registry is a customized UDDI service for looking up provided services by the service broker. The connection between the service broker and the service registry is SSL encrypted. These request and response messages are not digitally signed. Every SSL connection is established with client authentication. The SSL cipher suites used for the SSL connection between connector and service broker use the Diffie Hellman key exchange algorithm, the cipher suites used inside the secured SOA use the RSA algorithm for key exchange.

### IV. VERIFICATION METHODS

In this section three approaches for integration testing the secured SOA are discussed. The most straightforward approach On-the-fly Data Decryption was preferred, but due to technical limitations was not applicable. The Temporary Testing Interface in this case with many independent service providers would lead to an organizational problem. Further, this option would change the secured services and therefore would be risky from a security perspective. The Man-In-The-Middle Proxy was the most applicable method for this case. The services are unaltered and the in between proxies not only fetch all message but further enabled to edit the messages e.g. to test failure cases.

#### A. On-the-fly Data Decryption

This approach decrypts SSL encrypted data connections under some specific circumstances on-the-fly. The main advantage of this method is that it uses an existing verification

<sup>1</sup><http://www.w3.org/TR/soap/>

point and is completely passive. Hence, it will not influence the infrastructure in any way. The major difficulty is to access a capture point CP where the SSL network traffic, between a client C and a server S, can be recorded.

1) *SSL Decryption*: To decrypt SSL traffic the private key PK from the server S is needed to obtain the SSL session key. With this session key, the whole SSL conversation between the client C and the server S can be decrypted [8]. This is only possible when the used SSL cipher suite uses the RSA key exchange algorithm. Using a cipher suite, which uses the Diffie Hellman key exchange algorithm, SSL decryption with only the private key of the server is not possible.

2) *Diffie Hellman Key Exchange*: The SSL cipher suites used between the connector and the service broker must use the Diffie Hellman Ephemeral (DHE) algorithm for key exchange during the SSL handshake. Unlike the RSA key exchange, the DHE key exchange method does not use the private key from the server to encrypt the process of generating the symmetric encryption key for the SSL connection. This prevents already captured SSL traffic from being decrypted, if the private key of the server should be compromised in the future (see Bresson et al. in [3]). Decrypting the SSL connection while only having the private key of the server is not possible when using the DHE key exchange algorithm. The decryption of SSL connections when using the DHE key exchange method requires access to internal Diffie Hellman parameters. But these are in almost all cases not accessible. Acquiring these parameters would require massive changes to the SSL stack.

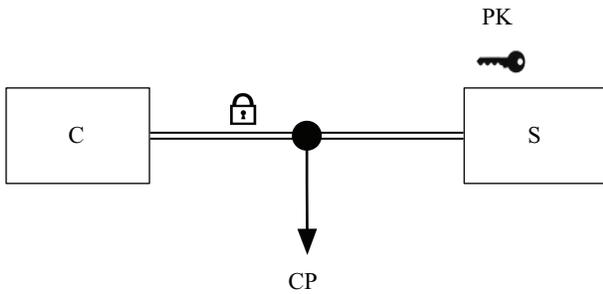


Fig. 2. On-the-fly Data Decryption Method

### B. Man-In-The-Middle Proxy

Man-in-the-middle proxies are additional active components in the infrastructure. They use the existing and unmodified interfaces, but they can influence the system in an unexpected way. As shown in Figure 3 the Proxy P has to act as the server S on one hand, and as the client C on the other hand. Since SSL is used with client authentication, the proxy must use an additional client certificate C3 as well as another server certificate C2. Disabling the validation of the common name (CN) of the SSL server certificate on the client allows the use of the servers server certificate C4 as the server certificate C2 of the proxy. The same is the case for the client certificates.

The clients client certificate C1 can be used as the proxies client certificate C3. (C1 == C3 and C2 == C4). The use of the man-in-the-middle proxies poses several advantages over a completely passive approach. Their use is independent of the transport protocol. They also allow altering the captured messages on-the-fly which makes them usable for other testing purposes as in (see [6] and [10]).

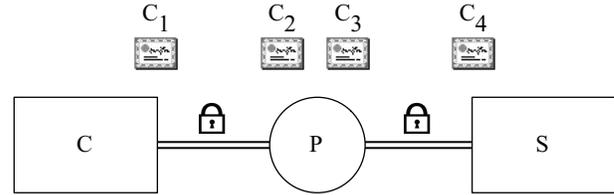


Fig. 3. Data Proxy Approach

### C. Temporary Testing Interface

The third method considered is the implementation of a Temporary Testing Interface. Every component like a client C or a server S must implement the same interface TTI. The application data AD is provided by the Temporary Testing Interface and can be collected for automated verification purposes. Temporary Testing Interfaces are an active approach to provide artificial testing interfaces for accessing the application data. The main advantage of implementing debug APIs is convenient access to the application data. This approach also provides the possibility to modify the application data in any way, even if it is digitally signed and encrypted. It is also independent from the used transport protocol. Disadvantages of this approach are security issues. The API must be built into every component of the secured SOA and it must be available during the integration process. Afterwards the API must be securely deactivated for production use. Removing the Temporary Testing Interface from the production components is complicated because almost every software layer of those components might be affected. These massive changes on the source code might affect the components in an unexpected way [6]. Another issue is the extended trust in the Temporary Testing Interface. It cannot be assured that the output of the interface is exactly the data which is processed.

The risk of abusing the Temporary Testing Interface is unsustainable for the secured SOA.

## V. IMPLEMENTATION

In this specific case we concluded that the method of the man-in-the-middle proxy provides the flexibility we need for this complex situation without seriously harming the security of each component and the overall system. Using the on-the-fly decryption method was not applicable because of the Diffie Hellman key exchange during the SSL handshake and for security reasons the implementation of a Temporary Testing Interfaces into every component was unsustainable due to possible security related side effects. Consequently, for the

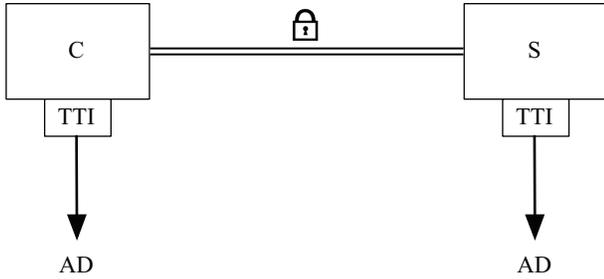


Fig. 4. Temporary Testing Interface Approach

integration tests of the secured SOA the man-in-the middle proxy method has been implemented.

#### A. Integration Test with the Man-In-The-Middle Proxy

1) *Test Case Definition*: Each test case description is split up into the definition of the control data and the definition of the verification data. The control data contains all information needed by the test engine TE to compose a request for the secured SOA. It consists of static parameter values for creating requests based on static data and dynamic parameter values for creating requests based on actual date/time, on responses from previous requests and many more. The verification data contains all information needed to verify the responses from the secured SOA and to verify the intermediate data. The definition of the verification data also consists of static parameter values and on dynamic parameter values. The test results are stored in a common repository that can be accessed by the test professionals in a convenient way.

2) *Proxy Distribution Scheme*: As shown in Figure 5 between every component a proxy instance P has been installed. The test cases are initiated by the test engine TE. PC\_SB captures the messages MC\_SB between the connector C and the system broker SB. PSB\_SR captures the messages MSB\_SR between the system broker SB and the service registry SR; PSB\_LS captures the messages MSB\_LS between the system broker SB and the log service LS. The messages MSB\_PSn between the system broker SB and the provided services PSn are captured by the proxy instances PSB\_PS1 to PSB\_PSn. All of the connections are SSL encrypted. Each proxy instance is configured as a server with a SSL server certificate on one hand, and as a client with its SSL client certificate on the other hand. The proxy instances are distributed over the entire secured SOA.

#### B. Automated Test Result Verification

The test engine TE initiates a request to the secured SOA. The request is composed out of control data. The secured SOA processes the request and as a result the test engine TE receives a response. This process also generates requests and responses between the intermediate components. This intermediate data is captured by the proxies and forwarded to the test engine TE.

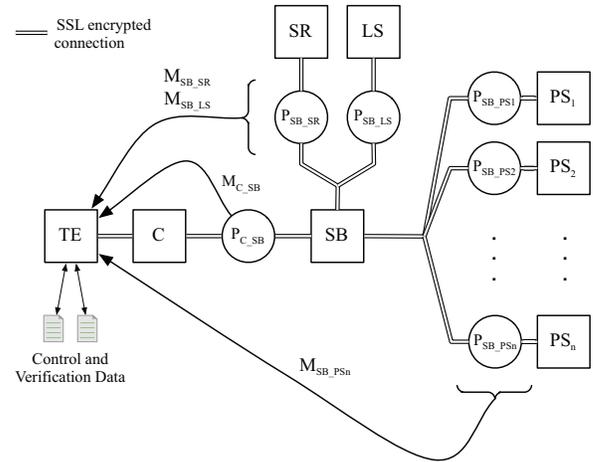


Fig. 5. Integration Test Scheme

TABLE I  
CONF. PARAMETERS OF A PROXY INSTANCE

Parameter	Description
name	Identifies the proxy instance for fetching the captured messages via JMX
remotehost	Specifies the host to forward the messages
remoteport	Specifies the port on the remotehost
localport	Specifies the port to listen on for incoming messages
sslservercert	Specifies the file that contains the SSL server certificate for the proxies' server side
sslclientcert	Specifies the file that contains the SSL client certificate for the proxies' client side

1) *Reassembling the Request/Response Sequences*: After a test case has finished, the verification starts. Since parallel testing is supported in most cases, the captured sequence of the requests and responses has to be associated to the appropriate test case by the test engine. Parallel running tests are not possible for some use cases. In this case the broker service stores session data that is not accessible. Such tests are run exclusively.

The design of the man-in-the-middle proxy allows the execution of a large number of proxy instances. Various proxy instances can be executed on one machine and a lot of machines can be distributed in the secured SOA. The configuration is done per machine. Table I shows the configuration parameters of a proxy instance. The man-in-the-middle-proxy and the test engine are implemented in Java. The captured messages are sent to the test engine via JMX<sup>2</sup>.

## VI. DISCUSSION

In secure software systems, access to application data is restricted or in some cases still not possible because of e.g. data encryption. All of the components are hardened and production ready. Integration tests in highly secure software systems may need special approaches. It is important what effects the testing method has on the production ready components. Using active

<sup>2</sup><http://java.sun.com/javase/technologies/core/mntr-mgmt/javamanagement/>

test components might influence the production system in an unexpected way. This report has shown how security restricts the testability. The goal is to apply verification methods, which allow accessing the data needed without serious impacts on security and on the system itself.

Every verification method has its own different strengths and weaknesses. There is no general rule that one verification method is more applicable than others. In the specific case of our study we have concluded that the method of the man-in-the-middle proxy provides the flexibility we need for this complex situation without seriously harming the security of each component and the overall system. From a security perspective passive verification methods - such as on-the-fly decryption - should always be preferred if applicable.

## VII. CONCLUSION AND FUTURE WORK

SOA testing in general is still a rapidly developing field. This work discusses one specific scenario, but it is assumable that similar scenarios face the same challenges. While this single scenario may not serve as a general pattern, the reflection of different strategies could be used as a baseline for other testing scenarios. Our future work aims to analyze similar cases and to generalize the results to provide guidelines for SOA testing in secure environments. Further we are investigating the use for the provided results for monitoring, surveillance and auditing the infrastructure during production.

## REFERENCES

- [1] S. Balasubramaniam, G. A. Lewis, E. Morris, S. Simanta, and D. B. Smith. Challenges for assuring quality of service in a service-oriented environment. In *PESOS '09: Proceedings of the 2009 ICSE Workshop on Principles of Engineering Service Oriented Systems*, pages 103–106, Washington, DC, USA, 2009. IEEE Computer Society.
- [2] L. Baresi and E. D. Nitto. *Test and Analysis of Web Services*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.
- [3] E. Bresson, O. Chevassut, and D. Pointcheval. Provably secure authenticated group diffie-hellman key exchange. *ACM Trans. Inf. Syst. Secur.*, 10(3):10, 2007.
- [4] R. Haux, E. Ammenwerth, W. Herzog, and P. Knaup. Health care in the information society. a prognosis for the year 2013. *International Journal of Medical Informatics*, 66(1-3):3–21, 11 2002/11/20/.
- [5] M. P. E. Heimdahl. Safety and software intensive systems: Challenges old and new. In *FOSE '07: 2007 Future of Software Engineering*, pages 137–152, Washington, DC, USA, 2007. IEEE Computer Society.
- [6] M. Jaffar-ur Rehman, F. Jabeen, A. Bertolino, and A. Polini. Testing software components for integration: a survey of issues and techniques: Research articles. *Softw. Test. Verif. Reliab.*, 17(2):95–133, 2007.
- [7] B. Katt, R. Breu, M. Hafner, T. Schabetsberger, R. Mair, and F. Wozak. Privacy and access control for ihe-based systems. *Electronic Healthcare*, pages 145–153, 2009/III.
- [8] H. K. Lee, T. Malkin, and E. Nahum. Cryptographic strength of ssl/tls servers: current and recent practices. In *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 83–92, New York, NY, USA, 2007. ACM.
- [9] E. Martin, S. Basu, and T. Xie. Automated testing and response analysis of web services. *Web Services, IEEE International Conference on*, 0:647–654, 2007.
- [10] R. Oppliger, R. Hauser, and D. Basin. Ssl/tls session-aware user authentication - or how to effectively thwart the man-in-the-middle. *Comput. Commun.*, 29(12):2238–2246, 2006.
- [11] R. Paul. End-to-end integration testing. In *Quality Software, 2001. Proceedings. Second Asia-Pacific Conference on*, pages 211–220, 2001.
- [12] L. Peyton, B. Stepien, and P. Seguin. Integration testing of composite applications. In *HICSS '08: Proceedings of the Proceedings of the 41st Annual Hawaii International Conference on System Sciences*, page 96, Washington, DC, USA, 2008. IEEE Computer Society.
- [13] L. Ribarov, I. Manova, and S. Ilieva. Testing in a service-oriented world. *InfoTech, Proceedings of the International Conference on Information Technologies*, 2007.
- [14] W. T. Tsai, J. Gao, X. Wei, and Y. Chen. Testability of software in service-oriented architecture. *Computer Software and Applications Conference, Annual International*, 2:163–170, 2006.
- [15] J. Vincent, G. King, P. Lay, and J. Kinghorn. Principles of built-in-test for run-time-testability in component-based software systems. *Software Quality Control*, 10(2):115–133, 2002.
- [16] F. Wozak, T. Schabetsberger, and E. Ammenwerth. End-to-end security in telemedical networks - a practical guideline. *International Journal of Medical Informatics*, 76(5-6):484–490, 6 2007.

# A Novel Software Metric Selection Technique Using the Area Under ROC Curves

Taghi M. Khoshgoftaar  
Florida Atlantic University  
Boca Raton, Florida 33431  
taghi@cse.fau.edu

Kehan Gao  
Eastern Connecticut State University  
Willimantic, Connecticut 06226  
gaok@easternct.edu

**Abstract**—Software quality can be monitored and estimated during the software development process. One common approach is to utilize classification models to predict quality of software modules prior to system deployment and then allocate software enhancement resources based on the predicted results. The predictive accuracy of classification models is often affected by the quality of input data. For example, redundant or irrelevant features harm prediction, and high dimensionality of input data may make classification very difficult. Feature (software metric) selection becomes a necessary step in the modeling process for such types of data sets. In this paper, we present our newly proposed threshold-based feature selection technique using the area under ROC (Receiver Operating Characteristic) curves and compare its performance to that of six standard filter-based feature selection methods. Our experiments are conducted on three groups of software data sets. The results demonstrate that our proposed threshold-based feature selection method has better or similar performance compared to the six standard filter-based feature selection approaches.

## I. INTRODUCTION

Software quality modeling is an important tool to achieve high quality software. The output of the modeling process can direct practitioners to allocate project resources strategically, e.g., assigning more resources to the program modules that have been predicted as high-risk or low-quality. The standard approach for software quality modeling is to apply data mining and machine learning algorithms to the software measurement data collected during the software development process and predict whether a program module is fault-prone (*fp*) or not fault-prone (*nfp*). In fact, data mining has been successfully applied in a variety of domains, such as protein structure prediction [1], network intrusion detection [2], text classification [3], and software defect prediction [4]. The software quality modeling process becomes ineffective and sometimes impractical if input data is of high dimensionality or contains a number of irrelevant or redundant features. Feature selection is required for such types of data sets. The objective of feature selection is to select a subset of features that minimize the prediction errors of the learner.

Generally, feature selection is divided into two groups, *wrapper*-based feature selection and *filter*-based feature selection. The main characteristic of wrapper-based techniques is that which attributes are selected is learner-dependent. The same classifier or inductive algorithm is used to both select the relevant features and execute the mining process [5]. Therefore, for a given data set, a wrapper-based technique may produce different feature subsets when using different learners. The potential problems of a wrapper-based technique lie in its high computational cost and a risk of overfitting to the model. In contrast, a filter-based technique is learner-independent. Once the data set is given, the filter-based technique will produce a feature subset (or feature subsets) that is (or are) correlated to the dependent attributes irrespective of whatever learner will be used after.

For this study, we propose a new threshold-based feature selection (TBFS) technique, which is in the filter-based feature selection

category. This method is developed from a classification performance metric. Each independent attribute is individually paired with the class attribute and the two-attribute data set is evaluated using the performance metric. In this study, we select AUC (area under a ROC curve) as the performance metric, since AUC has been widely used for evaluating the predictive capability of classifiers in many application domains, including software engineering [6], [7], [8]. AUC provides a single numerical metric that represents the predictive power of each attribute, and attributes with higher AUC values are considered better able to predict the class attribute. This is the first time AUC has been used for feature selection in this manner. Of course, we notice that Chen and Wasikowski proposed the FAST algorithm [9], which is based on the area under a ROC curve (AUC). However, in their work AUC was generated by moving the decision boundary of a single feature classifier with thresholds placed using an even-bin distribution. This means that they calculated a ROC curve by discretizing the distribution, while our technique is much more general compared to their work. Our technique does not require discretization, making it more precise and avoiding the determination of how wide the bins should be. Furthermore, our proposed TBFS can be readily extended by changing what classifier performance metric is used for feature ranking, although we only present the technique using AUC.

In order to validate the effectiveness of our proposed method, we compare TBFS to six commonly used filter-based feature selection techniques on three groups of software data sets, each group including three separate releases. The three groups of data sets exhibit different class distributions (i.e., the percentage of positive (*fp*) examples). Five different classifiers are applied to the training data sets with the attributes selected by the seven feature selection techniques. The experimental results demonstrate that our proposed TBFS method has better or similar performance compared to the best of the six standard filter-based feature selection approaches. Moreover, the comparison results show that TBFS is more appropriate when a training data set is highly imbalanced (i.e., the percentage of positive (*fp*) examples is very low).

The remainder of the paper is organized as follows. Section II describes related work. The six standard filter-based feature selection techniques and our newly proposed TBFS method, as well as the five classifiers and the associated classification performance metric used in study are presented in Section III. A case study is provided in Section IV. Finally, conclusions are drawn in Section V.

## II. RELATED WORK

Feature selection, as an important activity in data preprocessing, has been extensively studied for a few years in data mining and machine learning. Liu and Yu, in their research [10], provided a comprehensive survey of feature selection algorithms and presented

an integrated approach to intelligent feature selection. Another good review on various aspects of the attribute selection problem was done by Guyon and Elisseeff [11]. They outlined the main techniques and approaches used in attribute selection, including feature construction, feature ranking, multivariate feature selection, efficient search methods, and feature validity assessment methods.

Numerous variations of feature selection have been employed in a range of fields [12], [3], [13], [14], [15]. Jong et al. [15] introduced methods for feature selection based on support vector machines (SVM). Ilczuk et al. [14] investigated the importance of attribute selection in judging the qualification of patients for cardiac pacemaker implantation. In the context of text mining, Forman [3] investigated multiple filter-based feature ranking techniques.

Although feature selection has been widely applied in various data mining problems, its application in software quality and reliability engineering has been limited. Rodríguez et al. [16] applied feature subset selection with three filter-based models and two wrapper-based models to five software engineering data sets. The results showed that the reduced data sets maintained their prediction capability with fewer attributes than the original data sets. Moreover, while it was stated that the wrapper-based model was better than the filter-based model, it came at a high computational cost. Chen et al. [17] have studied feature selection using wrappers in the context of software cost/effort estimation. In their study, several COCOMO-I and COCOMO-II data sets were used. They also stated that the wrapper is theoretically quite slow, but since all the data sets in their study were very small, the experiments required only approximately 20 minutes per data set. They concluded that the reduced data set could improve the estimation and recommended feature selection in cost modeling, particularly when dealing with very small data sets.

Our research group recently studied various feature selection techniques including filter-based and wrapper-based methods [18], [19], [20], [21], [22] and applied them to a variety of software data sets. The results demonstrate that the performances of the classification models were maintained or even improved when over 85 percent of the features were eliminated from the original data sets.

### III. METHODOLOGY

#### A. Standard Filter-Based Feature Selection Techniques

The procedure of feature ranking is to score each feature according to a particular method, allowing the selection of the best set of features. The six standard filter-based feature ranking techniques used in this work include: chi-square (CS), information gain (IG), gain ratio (GR), two types of ReliefF (RF and RFW), and symmetrical uncertainty (SU). The chi-square -  $\chi^2$  (CS) test is used to examine the distribution of the class as it relates to the values of the feature in question. The null hypothesis is that there is no correlation; each value is as likely to have instances in any one class as any other class. Given the null hypothesis, the  $\chi^2$  statistic measures how far away the actual value is from the expected value:

$$\chi^2 = \sum_{i=1}^r \sum_{j=1}^{n_c} \frac{(O_{i,j} - E_{i,j})^2}{E_{i,j}}$$

where  $r$  is the number of different values of the feature,  $n_c$  is the number of classes (in this work,  $n_c = 2$ ),  $O_{i,j}$  is the number of instances with value  $i$  which are in class  $j$ , and  $E_{i,j}$  is the expected number of instances with value  $i$  and class  $j$ . The larger this  $\chi^2$  statistic, the more likely it is that the distribution of values and classes are dependent; that is, the feature is relevant to the class.

Information gain, gain ratio, and symmetrical uncertainty are measures based on the concept of entropy from information theory

---

#### Algorithm 1: Threshold-Based Feature Selection using AUC

---

**input :**

1. Data set  $D$  with features  $F^j, j = 1, \dots, m$ ;
2. Each instance  $\mathbf{x} \in D$  is assigned to one of two classes  $c(\mathbf{x}) \in \{fp, nfp\}$ ;
3. The value of attribute  $F^j$  for instance  $\mathbf{x}$  is denoted  $F^j(\mathbf{x})$ ;
4. Performance metric: AUC.

**output:** Ranking  $\mathbb{R} = \{r^1, \dots, r^m\}$  where  $r^j$  represents the rank for attribute  $F^j$ , i.e., the  $r^j$ -th most significant attribute as determined by metric AUC.

**for**  $F^j, j = 1, \dots, m$  **do**

Normalize $F^j \mapsto \hat{F}^j = \frac{F^j - \min(F^j)}{\max(F^j) - \min(F^j)}$ ; Calculate metric AUC using attribute $\hat{F}^j$ and class attribute $\{c(\mathbf{x})   \mathbf{x} \in D\}$ , $AUC(\hat{F}^j)$ ;
---

Create attribute ranking  $\mathbb{R}$  using  $AUC(\hat{F}^j) \forall j$

---

[23]. Information gain (IG) is the information provided about the target class attribute  $Y$ , given the value of another attribute  $X$ . IG measures the decrease of the weighted average impurity of the partitions compared to the impurity of the complete set of data. A drawback of IG is that it tends to prefer attributes with a larger number of possible values, i.e., if one attribute has a larger number of values, it will appear to gain more information than those with fewer values, even if it is actually no more informative. One strategy to solve this problem is to use the gain ratio (GR), which penalizes multiple-valued attributes. Symmetrical uncertainty (SU) is another way to overcome the problem of IG's bias toward attributes with more values, doing so by dividing by the sum of the entropies of  $X$  and  $Y$ .

Relief is an instance-based feature ranking technique introduced by Kira and Rendell [24]. ReliefF is an extension of the Relief algorithm that can handle noise and multiclass data sets, and is implemented in the WEKA tool [23]. When the WeightByDistance (weight nearest neighbors by their distance) parameter is set as default (false), the algorithm is referred to as RF; when the parameter is set to 'true', the algorithm is referred to as RFW.

#### B. Threshold-Based Feature Selection Technique

The threshold-based feature selection (TBFS) technique was developed and implemented by our research group within WEKA [23]. The procedure is shown in Algorithm 1. Each independent attribute is paired individually with the class attribute, and that two-attribute data set is evaluated using the AUC performance metric. The feature ranking framework we propose includes normalizing the attribute values (so that they fall between 0 and 1) and treating those values as the posterior probabilities from which to calculate AUC. Note that no classifiers were built during the feature selection process.

More specifically, feature  $F^j$  is normalized to  $\hat{F}^j$  by

$$\hat{F}^j = \frac{F^j - \min(F^j)}{\max(F^j) - \min(F^j)}$$

$\hat{F}^j$  can now be thought of as a posterior probability. Unlike the standard binary classification methods that use the default decision threshold of 0.5 to assign the predicted class, we propose the use of performance measures that can be calculated at various points in the distribution of  $\hat{F}^j$ . One example is the area under a ROC curve (AUC). The benefit of the proposed method is that it exhibits the characteristics of prediction models at every decision threshold value ( $t \in [0, 1]$ ) instead of at a single point ( $t = 0.5$ ), therefore providing a better and more general picture of the classification performance.

Similar to the procedure for calculating rates in a classification setting with a posterior probability, the true positive rate ( $TPR$ ), true negative rate ( $TNR$ ), false positive rate ( $FPR$ ), and false negative rate ( $FNR$ ) can be calculated at each threshold  $t \in [0, 1]$  relative to the normalized attribute  $\hat{F}^j$ . For example,

$$TPR(t) = \frac{|\{\mathbf{x} \in D \mid (\hat{F}^j(\mathbf{x}) > t) \cap (c(\mathbf{x}) = fp)\}|}{|\{\mathbf{x} \in D \mid c(\mathbf{x}) = fp\}|}. \quad (1)$$

$FNR(t)$ ,  $TNR(t)$ , and  $FPR(t)$  can be defined analogously. AUC utilizes  $TPR(t)$  and  $FPR(t)$  as described below.

Receiver Operating Characteristic [25], or ROC, curves graph true positive rate on the  $y$ -axis versus the false positive rate on the  $x$ -axis at each threshold  $t \in [0, 1]$ . The resulting curve illustrates the trade-off between true positive rate and false positive rate. In this study, ROC curves are generated by varying the decision threshold  $t$  used to transform the normalized attribute values into a predicted class. AUC is used to provide a single numerical metric for comparing the predictive power of each attribute, and attributes with higher AUC values are determined to better predict the class attribute. In this manner, the attributes can be ranked from most to least predictive based on the AUC metric. This definition is different than the one used by Chen and Wasikowski [9], which consider only a small subset of the possible threshold values when calculating the true positive and false positive rates.

### C. Classification Performance Metric

The effectiveness of each feature selection is assessed by evaluating the classification performance of the models subsequently trained and tested with that particular feature selection. In our experiments, we use AUC as the classification performance metric. Our selection of AUC is based on one of its characteristics, namely its invariance to a priori class probability distributions. Moreover, it has been proven to be statistically consistent [26]. AUC does not emphasize one class over the other as may be the case in some other performance metrics, thus it is not biased against the positive ( $fp$ ) class. Given the imbalanced nature of our data sets (see in later section), AUC is an appropriate measure for comparing the classification performance of the learners. In fact, AUC serves as an aid to both feature ranking and final classification evaluation in this study.

### D. Classifiers

In this study, the software quality prediction models are built with five different learners, including naïve Bayes [27], multilayer perceptron [28], K-nearest neighbors [29], support vector machine [30], and logistic regression [31]. The five learners were selected because of their common use in the software engineering domain and data mining, and also because they do not have a built-in feature selection capability. Unless stated otherwise, we use default parameter settings for the different learners as specified in WEKA [23]. Parameter settings are changed only when a significant improvement in performance is obtained.

Naïve Bayes classifier (NB) [27] utilizes Bayes's rule of conditional probability and is termed 'naïve' because it assumes conditional independence of the features.

MultiLayer Perceptron (MLP) [28] is a neural network of simple neurons called perceptrons. Some related parameters of MLP were set as follows. The `hidden-Layers` parameter was set to '3' to define a network with one hidden layer containing three nodes, and the `validationSetSize` parameter was set to '10' to cause the classifier to leave 10% of the training data aside to be used as a validation set to determine when to stop the iterative training process.

TABLE I  
DATA CHARACTERISTICS

Data set	Rel.	thd	#Attri.	#Obj.	#fp	%fp	#nfp	%nfp
Eclipse-1	2.0	10	209	377	23	6.1	354	93.9
	2.1	5	209	434	34	7.8	400	92.2
	3.0	10	209	661	41	6.2	620	93.8
Eclipse-2	2.0	5	209	377	52	13.8	325	86.2
	2.1	4	209	434	50	11.5	384	88.5
	3.0	5	209	661	98	14.8	563	85.2
Eclipse-3	2.0	3	209	377	101	26.8	276	73.2
	2.1	2	209	434	125	28.8	309	71.2
	3.0	3	209	661	157	23.8	504	76.2

K-Nearest Neighbors (KNN) [29] classifier, also called instance-based learning, uses distance-based comparisons. The choice of distance metric is critical. KNN was built with changes to three parameters. The `distanceWeighting` parameter was set to 'Weight by 1/distance', the `kNN` parameter was set to '5', and the `crossValidate` parameter was turned on (set to 'true').

Support VectorMachine (SVM) [30] also called SMO in WEKA had two changes to the default parameters: the `complexity constant c` was set to '5.0' and `build Logistic Models` was set to 'true'. By default, a linear kernel was used.

Logistic Regression (LR) [31] is a statistical regression model for categorical prediction by fitting data to a logistic curve.

## IV. A CASE STUDY

### A. Data Sets

In our experiments, we use publicly available data, namely the Eclipse defect counts and complexity metrics data set obtained from the PROMISE data repository [32]. In particular, we use the metrics and defects data at the software packages level. The original data for Eclipse packages consists of three releases denoted 2.0, 2.1, and 3.0 respectively. Each release as reported by [33] contains the following information: the name of the package for which the metrics are collected (name), the number of defects reported six months prior to release (pre-release defects), the number of defects reported six months after release (post-release defects), a set of complexity metrics computed for classes or methods and aggregated in terms of average, maximum, and total (complexity metrics), and the abstract syntax tree of the package consisting of the node size, type, and frequency (structure of abstract syntax tree(s)). For our study we transform the original data by: (1) removing all non-numeric attributes, including the package names, and (2) converting the post-release defects attribute to a binary class attribute with fault-prone ( $fp$ ) being the minority class and not-fault-prone ( $nfp$ ), the majority class. Membership in each class is determined by a post-release defects threshold  $thd$ , which separates  $fp$  from  $nfp$  packages by classifying packages with  $thd$  or more post-release defects as  $fp$  and the remaining as  $nfp$ . In our study, we use  $thd = \{10, 5, 3\}$  for releases 2.0 and 3.0 while we use  $thd = \{5, 4, 2\}$  for release 2.1. This results in three groups. Each group contains three data sets, one for each release. The reason why a different set of thresholds is chosen for release 2.1 is that we would like to keep similar class distributions for the data sets in the same group. All data sets contain 209 attributes (208 independent attributes and 1 dependent attribute). Table I shows the characteristics of the data sets after transformation for each group. These data sets exhibit different distribution of class skew (i.e., the percentage of  $fp$  examples).

### B. Results & Analysis

The experiments were performed on the three groups of Eclipse data sets. First, we used the six standard feature ranking techniques

TABLE II  
CLASSIFICATION PERFORMANCE

(a) Eclipse-1						
	NB	MLP	KNN	SVM	LR	Avg
CS	0.8355	0.8539	0.8829	0.8911	0.8628	0.8652
GR	0.8152	0.8094	0.8337	0.8604	0.8458	0.8329
IG	0.8527	0.8750	0.9003	0.9029	0.8716	0.8805
RF	0.8174	0.8228	0.8275	0.8519	0.8631	0.8365
RFW	0.8132	0.8356	0.7752	0.8523	0.8538	0.8260
SU	0.8323	0.8307	0.8689	0.8820	0.8573	0.8542
<b>AUC</b>	<b>0.8753</b>	<b>0.8850</b>	<b>0.9104</b>	<b>0.9156</b>	<b>0.8758</b>	<b>0.8924</b>
<b>Avg</b>	0.8345	0.8446	0.8570	0.8795	0.8614	

(b) Eclipse-2						
	NB	MLP	KNN	SVM	LR	Avg
CS	0.8504	0.8823	0.8912	0.9179	0.8999	0.8883
GR	0.8260	0.8576	0.8536	0.8913	0.8872	0.8631
IG	<b>0.8565</b>	0.8881	0.8927	0.9197	0.9024	0.8919
RF	0.8262	0.8456	0.7854	0.8730	0.8711	0.8402
RFW	0.8214	0.8276	0.7684	0.8364	0.8529	0.8213
SU	0.8484	0.8793	0.8867	0.9145	0.8987	0.8855
<b>AUC</b>	<b>0.8533</b>	<b>0.8930</b>	<b>0.8941</b>	<b>0.9204</b>	<b>0.9094</b>	<b>0.8940</b>
<b>Avg</b>	0.8403	0.8676	0.8531	0.8962	0.8888	

(c) Eclipse-3						
	NB	MLP	KNN	SVM	LR	Avg
CS	0.7877	0.8561	<b>0.8618</b>	0.8852	0.8871	0.8556
GR	0.7682	0.8257	0.8419	0.8669	0.8645	0.8334
IG	0.7889	0.8568	0.8600	0.8861	0.8865	<b>0.8557</b>
RF	0.7879	0.8141	0.7530	0.8424	0.8406	0.8076
RFW	<b>0.8032</b>	0.8301	0.7605	0.8559	0.8536	0.8207
SU	0.7854	0.8528	0.8562	0.8851	0.8831	0.8525
<b>AUC</b>	<b>0.7859</b>	<b>0.8590</b>	0.8583	<b>0.8866</b>	<b>0.8880</b>	0.8556
<b>Avg</b>	0.7867	0.8421	0.8274	0.8726	0.8719	

and TBFS with AUC to rank the attributes according to their respective scores. Then, we selected  $\lceil \log_2 n \rceil$  attributes that had the highest scores, where  $n$  is the number of the independent attributes in the original data set. In this study,  $n = 208$ , so  $\lceil \log_2 n \rceil = 8$ . We choose  $\lceil \log_2 n \rceil$  attributes, because 1) related literature does not provide guidance on the appropriate number of features to select; and 2) one of our recent empirical studies [34] showed that it was appropriate to use  $\lceil \log_2 n \rceil$  features when using WEKA to build random forests learners for binary classification in general and imbalanced data sets in particular. Although we used different learners here, a preliminary study showed that  $\lceil \log_2 n \rceil$  is still appropriate for various learners.

After feature selection, we applied five different classifiers to the training data sets with the selected attributes, and we used AUC to evaluate the performance of the classifications. All the results are reported in Table II, with three subtables, each representing the results for each group of the data sets. In the experiments, ten runs of five-fold cross-validation were performed. The values presented in the tables represent the average AUC for every classification model constructed over the ten runs of five-fold cross-validation. The best feature selection technique in terms of their classification performance (AUC) for each classifier is highlighted with **bold**. We also present the average performance (last column of the tables) for each feature selection method across five classifiers as well as the average of each learner (last row of the tables) across seven feature selection techniques. The outputs demonstrate that the AUC method outperformed six standard feature selection techniques over all five learners for Eclipse-1, over four learners for Eclipse-2, and three learners for Eclipse-3.

We also conducted a two-way ANalysis Of VAriance (ANOVA) F test [35] on the classification performance for each group of the data sets (Eclipse-1,-2 and-3) separately to examine if the performance difference (better/worse) is statistically significant or not. Two factors

are considered in the test: Factor A represents seven feature selection methods (six standard feature ranking techniques and the newly proposed TBFS method), while Factor B represents five classifiers (NB, MLP, KNN, SVM and LR). The null hypothesis for the ANOVA test is that all the group population means are the same and the alternate hypothesis is that at least one pair of means is different. In addition, the interaction between Factor A and Factor B is also taken into account in the test. Table III presents the ANOVA results for each group of data sets. The  $p$ -value is zero for each main factor and interaction of each group of data sets. This means that at least two filter-based feature selection techniques are significantly different from each other and also at least two classifiers performed significantly distinct from each other. In addition, the interaction significantly affects classification performance in each group of data sets. In other words, changing value of Factor A will significantly influence the value of Factor B, and vice versa.

We further carried out a multiple comparison test [35] on each main factor and their interaction with Tukey's honestly significant difference criterion. Note that for all the ANOVA and multiple comparison tests, the significance level  $\alpha$  was set to 0.05. Figures 1 through 3 show the multiple comparisons on the three groups of data sets, each with three subfigures representing Factor A, Factor B, and interaction  $A \times B$ , respectively. The figures display graphs with each group mean represented by a symbol ( $\circ$ ) and 95% confidence interval as a line around the symbol. Two means are significantly different if their intervals are disjoint, and are not significantly different if their intervals overlap. Matlab was used to construct the ANOVA models and perform the multiple comparisons presented in this work, and the assumptions for constructing ANOVA models were validated. For Factor A and interaction  $A \times B$ , the proposed TBFS method (AUC) is highlighted with thick lines for clear illustration purpose. From these figures we can see the following points:

- Among the six standard filter-based feature selection methods, IG performed best, followed by CS and SU; the other three techniques, GR, RF, and RFW, performed significantly worse than the first three. The AUC method performed better than all these six standard feature selection techniques. This pattern is reflected by all groups of data sets. However, the significance levels are different with respect to the different class distribution of each group of data sets. For example, the distinction between each pair of AUC, IG and CS can be clearly seen in Fig. 1(a) but not in Fig. 3(a).
- Of the five classifiers, SVM and LR performed best, followed by MLP and KNN; NB performed worst. This pattern is also true for all groups of data sets. However, a slight difference exists between groups of data. For instance, KNN performed better than MLP for Eclipse-1, but the reverse finding exhibits for Eclipse-2 and Eclipse-3.
- There are 35 groups for interaction  $A \times B$ ; these are formed by each of seven feature selection techniques being combined with five classifiers. The group means demonstrate that in addition to the two main factors, the classification performance is heavily influenced by their interactions. For example, for Eclipse-1, AUC significantly outperformed five standard feature selection techniques on average (see Fig. 1(a)) but this pattern was not observed when LR was used (see Fig. 1(c)).
- One point that is clearly observed in the multiple comparisons is that the performance advantage of AUC over other methods becomes increasingly dramatic as the class imbalance becomes more serious (i.e., the percentage of  $fp$  examples becomes

TABLE III  
TWO-WAY ANOVA TABLES

(a) Eclipse-1

Source	Sum Sq.	d.f.	Mean Sq.	F	<i>p</i> -value
A	0.5733	6	0.0955	72.45	0
B	0.2460	4	0.0615	46.63	0
A × B	0.2030	24	0.0085	6.41	0
Error	1.3384	1015	0.0013		
Total	2.3606	1049			

(b) Eclipse-2

Source	Sum Sq.	d.f.	Mean Sq.	F	<i>p</i> -value
A	0.7392	6	0.1232	63.38	0
B	0.4633	4	0.1158	59.58	0
A × B	0.1928	24	0.0080	4.13	0
Error	1.9731	1015	0.0019		
Total	3.3684	1049			

(c) Eclipse-3

Source	Sum Sq.	d.f.	Mean Sq.	F	<i>p</i> -value
A	0.3531	6	0.0589	80.82	0
B	1.0675	4	0.2669	366.47	0
A × B	0.2770	24	0.0115	15.85	0
Error	0.7391	1015	0.0007		
Total	2.4367	1049			

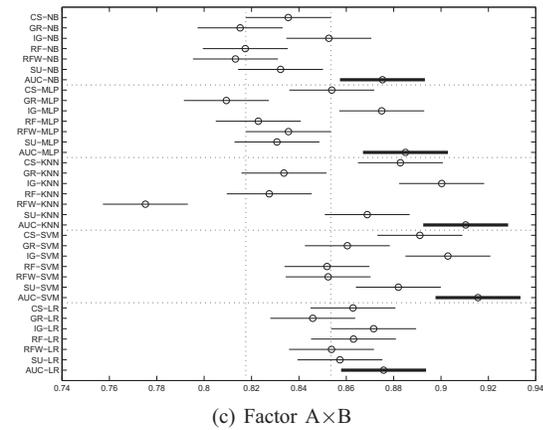
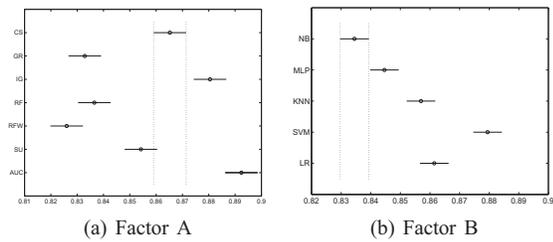


Fig. 1. Eclipse-1: Multiple Comparison

increasing low). This may implies that the AUC method is especially appropriate when classification occurs for highly imbalanced data (e.g., the percentage of positive (*fp*) examples is less than 10%).

Our recent work [36] has shown that classification models built on smaller subsets of attributes via the six commonly used filter-based feature selection techniques had similar or better performances than those built with a complete set of attributes. Thus, we did not present the results for full data sets in this paper.

V. CONCLUSION

Data mining has been applied in a wide variety of domains, including software defect prediction. However, one of the problems often encountered in the data mining process is that sometimes a

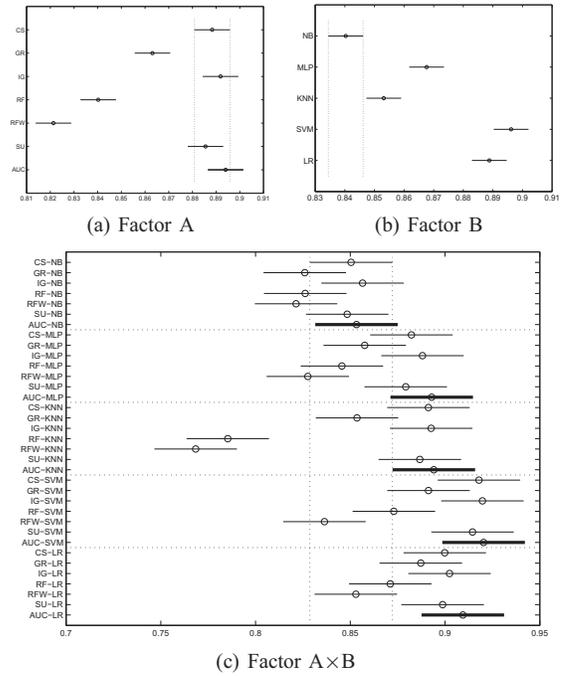


Fig. 2. Eclipse-2: Multiple Comparison

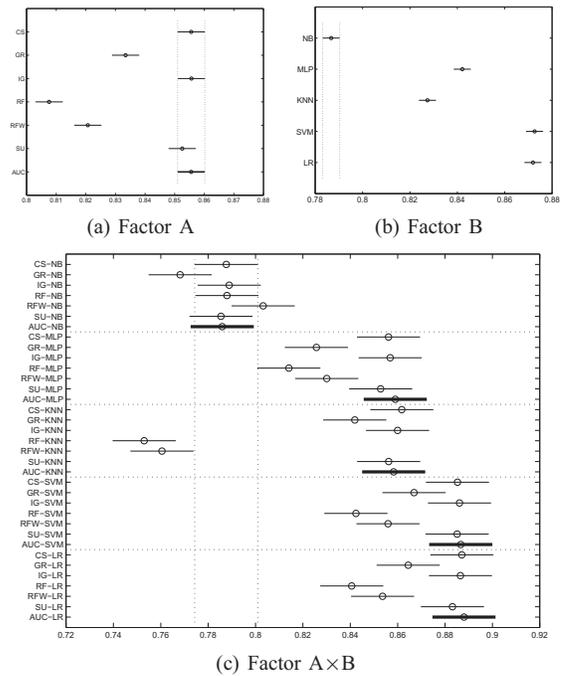


Fig. 3. Eclipse-3: Multiple Comparison

data set has an overabundance of features or attributes. Selecting a subset of features is needed in such situations. In this paper, we present a new threshold-based feature selection technique, in which classification algorithms are not involved. In order to evaluate the effectiveness of the proposed method, we applied this feature selection technique to three groups of software data sets, each group having three separate releases. We built a number of classification models using the selected attributes and five different classifiers. We also compared the new method to six commonly used standard feature selection techniques. The experimental results demonstrate that our proposed method performed better than or similar to the best of six commonly used feature selection techniques. Furthermore, the performance strength of TBFS over other methods becomes increasingly evident as the class imbalance becomes more serious. Of course, the conclusions drawn in this paper regarding the effectiveness of the proposed feature selection approach are based on the experiments conducted on the data sets from this specific software system. The extension of such conclusions to other types of data sets requires further study. We leave this as part of our future research. Future work also includes an extension of the approach by replacing AUC with different classifier performance metrics.

#### REFERENCES

- [1] Y. Zhang, "Progress and challenges in protein structure prediction," *Current Opinion in Structural Biology*, vol. 18, no. 3, pp. 342 – 348, 2008.
- [2] T. M. Khoshgoftaar, K. Gao, and N. H. Ibrahim, "Evaluating indirect and direct classification techniques for network intrusion detection," *Intelligent Data Analysis*, vol. 9, no. 3, pp. 309–326, 2005.
- [3] G. Forman, "An extensive empirical study of feature selection metrics for text classification," *Journal of Machine Learning Research*, vol. 3, pp. 1289–1305, March 2003.
- [4] C. Catal and B. Diri, "Software defect prediction using artificial immune recognition system," in *SE'07: Proceedings of the 25th conference on IASTED International Multi-Conference*. Anaheim, CA, USA: ACTA Press, 2007, pp. 285–290.
- [5] G. H. John, R. Kohavi, and K. Pfleger, "Irrelevant features and the subset selection problem," in *Proceedings of the 11th International Conference on Machine Learning*, 1994, pp. 121–129.
- [6] Y. Jiang, J. Lin, B. Kukic, and T. Menzies., "Variance analysis in software fault prediction models," in *Proceedings of the 20th IEEE International Symposium on Software Reliability Engineering*, Bangalore-Mysore, India, Nov. 16-19 2009, pp. 99–108.
- [7] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking classification models for software defect prediction: A proposed framework and novel findings," *IEEE Transactions on Software Engineering*, vol. 34, no. 4, pp. 485–496, July-August 2008.
- [8] T. Menzies, J. Greenwald, , and A. Frank, "Data mining static code attributes to learn defect predictors," *IEEE Transactions on Software Engineering*, vol. 33, no. 1, pp. 2–13, Jan. 2007.
- [9] X.-w. Chen and M. Wasikowski, "Fast: a roc-based feature selection metric for small samples and imbalanced data classification problems," in *KDD '08: Proc. 14th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2008, pp. 124–132.
- [10] H. Liu and L. Yu, "Toward integrating feature selection algorithms for classification and clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 4, pp. 491–502, 2005.
- [11] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, March 2003.
- [12] S. Doraisamy, S. Golzari, N. M. Norowi, N. Sulaiman, and N. I. Udzir, "A study on feature selection and classification techniques for automatic genre classification of traditional malay music," in *Ninth International Conference on Music Information Retrieval*, Philadelphia, PA, USA, Sept. 14-18 2008, pp. 331–336.
- [13] M. A. Hall and G. Holmes, "Benchmarking attribute selection techniques for discrete class data mining," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 6, pp. 1437 – 1447, Nov/Dec 2003.
- [14] G. Ilczuk, R. Mlynarski, W. Kargul, and A. Wakulicz-Deja, "New feature selection methods for qualification of the patients for cardiac pacemaker implantation," *Computers in Cardiology*, vol. 34, no. 2-3, pp. 423–426, 2007.
- [15] K. Jong, E. Marchiori, M. Sebag, and A. van der Vaart, "Feature selection in proteomic pattern data with support vector machines," in *Proceedings of the 2004 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, Oct 7-8 2004.
- [16] D. Rodriguez, R. Ruiz, J. Cuadrado-Gallego, and J. Aguilar-Ruiz, "Detecting fault modules applying feature selection to classifiers," in *Proceedings of 8th IEEE International Conference on Information Reuse and Integration*, Las Vegas, Nevada, August 13-15 2007, pp. 667–672.
- [17] Z. Chen, T. Menzies, D. Port, and B. Boehm, "Finding the right data for software cost modeling," *IEEE Software*, vol. 22, no. 6, pp. 38–46, November 2005.
- [18] K. Gao, T. M. Khoshgoftaar, and A. Napolitano, "Exploring software quality classification with a wrapper-based feature ranking technique," in *Proceedings of 21st IEEE International Conference on Tools with Artificial Intelligence*, Newark, New Jersey, Nov. 2-4 2009, pp. 67–74.
- [19] K. Gao, T. M. Khoshgoftaar, and H. Wang, "An empirical investigation of filter attribute selection techniques for software quality classification," in the *2009 IEEE International Conference on Information Reuse and Integration*, Las Vegas, NV, August 10-12 2009, pp. 272–277.
- [20] T. M. Khoshgoftaar, L. A. Bullard, and K. Gao, "Attribute selection using rough sets in software quality classification," *International Journal of Reliability, Quality and Safety Engineering*, vol. 16, no. 1, pp. 73–89, 2009.
- [21] H. Wang, T. M. Khoshgoftaar, K. Gao, and N. Seliya, "Mining data from multiple software development projects," in *Proceedings of the 3rd IEEE International Workshop Mining Multiple Information Sources*, Miami, FL, Dec. 6 2009, pp. 551–557.
- [22] T. M. Khoshgoftaar and K. Gao, "Feature selection with imbalanced data for software defect prediction," in *Proceedings of the 8th International Conference on Machine Learning and Applications*, Miami, FL, Dec 13-15 2009, pp. 235–240.
- [23] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. Morgan Kaufmann, 2005.
- [24] K. Kira and L. A. Rendell, "A practical approach to feature selection," in *Proceedings of 9th International Workshop on Machine Learning*, 1992, pp. 249–256.
- [25] F. Provost and T. Fawcett, "Robust classification for imprecise environments," *Machine Learning*, vol. 42, pp. 203–231, 2001.
- [26] C. X. Ling, J. Huang, and H. Zhang, "Auc: A better measure than accuracy in comparing learning algorithms," in *Canadian Conference on AI*, 2003, pp. 329–341.
- [27] G. H. John and P. Langley, "Estimating continuous distributions in bayesian classifiers," in *Proceedings of Eleventh Conference on Uncertainty in Artificial Intelligence*, vol. 2, San Mateo, 1995, pp. 338–345.
- [28] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Prentice-Hall, 1998.
- [29] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Machine Learning*, vol. 6, no. 1, pp. 1573–0565, 1991.
- [30] J. Shawe-Taylor and N. Cristianini, *Support Vector Machines*, 2nd ed. Cambridge University Press, 2000.
- [31] S. Le Cessie and J. C. Van Houwelingen, "Ridge estimators in logistic regression," *Applied Statistics*, vol. 41, no. 1, pp. 191–201, 1992.
- [32] G. Boetticher, T. Menzies, and T. Ostrand. (2007) Promise repository of empirical software engineering data. [Online]. Available: <http://promisedata.org/>
- [33] T. Zimmermann, R. Premraj, and A. Zeller, "Predicting defects for eclipse," in *Proceedings of the 29th International Conference on Software Engineering Workshops*. Washington, DC, USA: IEEE Computer Society, 2007, p. 76.
- [34] T. M. Khoshgoftaar, M. Golawala, and J. Van Hulse, "An empirical study of learning from imbalanced data using random forest," in *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence*, vol. 2, Washington, DC, USA, 2007, pp. 310–317.
- [35] M. L. Berenson, M. Goldstein, and D. Levine, *Intermediate Statistical Methods and Applications: A Computer Package Approach*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1983.
- [36] K. Gao, T. M. Khoshgoftaar, H. Wang, and N. Seliya, "Choosing software metrics for defect prediction: An investigation on feature selection techniques," Florida Atlantic University, Tech. Rep. FAU-CSE-TR-2009-11-2, November 2009.

# Automatic Bug Triage using Semi-Supervised Text Classification

Jifeng Xuan<sup>1</sup>

He Jiang<sup>2,3</sup>

Zhilei Ren<sup>1</sup>

Jun Yan<sup>4</sup>

Zhongxuan Luo<sup>1,2</sup>

<sup>1</sup>School of Mathematical Sciences, Dalian University of Technology, Dalian, 116024 China

<sup>2</sup>School of Software, Dalian University of Technology, Dalian, 116621 China

<sup>3</sup>State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, 100190 China

<sup>4</sup>Technology Center of Software Engineering, Institute of Software, Chinese Academy of Sciences, Beijing, 100190 China

<sup>1</sup>{xuan, ren}@mail.dlut.edu.cn

<sup>2</sup>{jianghe, zxluo}@dlut.edu.cn

<sup>4</sup>junyan@acm.org

**Abstract**—In this paper, we propose a semi-supervised text classification approach for bug triage to avoid the deficiency of labeled bug reports in existing supervised approaches. This new approach combines naive Bayes classifier and expectation-maximization to take advantage of both labeled and unlabeled bug reports. This approach trains a classifier with a fraction of labeled bug reports. Then the approach iteratively labels numerous unlabeled bug reports and trains a new classifier with labels of all the bug reports. We also employ a weighted recommendation list to boost the performance by imposing the weights of multiple developers in training the classifier. Experimental results on bug reports of Eclipse show that our new approach outperforms existing supervised approaches in terms of classification accuracy.

**Keywords**- automatic bug triage; expectation-maximization; semi-supervised text classification; weighted recommendation list

## I. INTRODUCTION

Most of large software projects employ a bug tracking system (bug repository) to manage bugs and developers. In software development and maintenance, a bug repository is a significant software repository for storing the bugs submitted by users. Those users, including developers, testers and end users, submit the content of bugs as *bug reports* to identify software defects or user suggestions. In the bug repositories, Bugzilla [6] is the most popular one in open source softwares.

Before verifying and modifying a bug, each bug report must be assigned to a relevant developer who could fix it [7]. This process of assignment is called *bug triage*. In traditional bug repositories, all the bugs are manually triaged by some specialized developers (triggers) [7]. Such manual work is expensive in labor costs. Taken Eclipse (an open source integrated development environment [8]) as an example, 37 bugs are submitted to the bug repository and 3 person-hours are spent on assigning the bug reports per day on average [1].

Aiming to reduce the human labor costs, some supervised text classification approaches have been proposed for automatic bug triage, including naive Bayes (NB) classifier [7] and support vector machine [3]. These approaches treat bug triage as the classification of text content of bug reports and treat relevant developers as class labels. The supervised approaches train learnable classifiers with existing bug reports

and then predict relevant developers for the incoming bug reports with these classifiers. In addition, the classification accuracy of these approaches is not high enough, so most of the bug triage approaches employ a recommendation list to provide the candidate developers to be selected by human triagers.

Before training a supervised classifier for bug triage, a necessary step is to collect numerous *labeled bug reports*, which are bug reports marked with their relevant developers. However, labeled bug reports are insufficient. Bettenburg, et al. investigate the existing quality problems of the bug reports in actual bug repositories [4]. Since some of bug reports are not well-formed, it is difficult to provide correct labels for all the bug reports. In practice, even a human triager usually mistakenly labels bug reports with developers who cannot fix the bugs. Jeong, et al. report that 44% of bugs are assigned to mistaken developers by triagers during the first assignment [11]. In other words, nearly half of bug reports may consist of wrong labels after they are assigned. Motivated by the deficiency of labeled bug reports with good quality, we take advantage of *unlabeled bug reports*, which are original bug reports without developer information.

In this paper, we propose a semi-supervised text classification approach to improve the classification accuracy of bug triage. This semi-supervised approach enhances a NB classifier by applying expectation-maximization (EM) based on the combination of unlabeled and labeled bug reports. First, this semi-supervised approach trains a classifier with labeled bug reports. Then, the approach iteratively labels the unlabeled bug reports and trains a new classifier with labels of all the bug reports. To adjust bug triage, we update such a semi-supervised approach with a weighted recommendation list (WRL) to augment the effectiveness of unlabeled bug reports. This WRL is employed to probabilistically label an unlabeled bug report with multiple relevant developers instead of a single relevant developer. The experimental results on Eclipse indicate that the semi-supervised approach increases the classification accuracy by up to 6%, compared to the original accuracy of 11% to 43% using the supervised NB classifier.

This paper makes the following main contributions:

1) *A semi-supervised text classification approach for bug triage*: We add the unlabeled bug reports to the existing

supervised approach to avoid the deficiency of labeled bug reports.

2) *A weighted recommendation list for the semi-supervised approach:* We provide a weighted recommendation list for augmenting the semi-supervised approach using probabilistic labels of unlabeled bug reports. Based on this weighted recommendation list, we improve the classification accuracy for the semi-supervised approach.

The remainder of this paper is organized as follows. Section II shows the previous related work of bug triage. Section III presents the semi-supervised bug triage approach and its augmentation of a weighted recommendation list. Section IV shows the experiments on bug reports of Eclipse. In Section V, we discuss the potential problems in the semi-supervised bug triage. Section VI concludes this paper and presents the future work.

## II. RELATED WORK

As to our knowledge, there is no semi-supervised bug triage approach in the literature. All the existing approaches on bug triage and its relevant problems are based on supervised or unsupervised learning. Čubranić & Murphy propose the first work on automatic bug triage [7]. They innovatively apply a supervised learning approach (NB classifier) using text classification to predict relevant developers. They also report the basic steps in the preprocess and provide a set of heuristics for labeling bug reports [2]. Anvik, et al. extend the above machine learning approach [3]. They call their bug triage approach as a semi-automatic approach since they firstly employ a recommendation list to provide the candidate developers for human triagers. Compared with the text classification approaches, Matter, et al. investigate the expertise model of developers on bug triage [12]. Jeong, et al. propose a bug tossing graph approach based on Markov chains from the knowledge of reassigning [11].

The most relevant work of bug triage is detecting duplicate bug reports. In a bug repository, some bug reports are marked as duplicates since such bug reports are just similar as some other handled ones. Runeson, et al. [15] and Wang, et al. [16] remove duplicate bug reports based on supervised natural language processing approaches. Jalbert & Weimer tackle the problem of duplicate bug reports by clustering, an unsupervised learning approach [10]. In contrast to removing the duplicate bug reports, Bettenburg, et al. merge duplicate ones by adding extra information to diagnose actual problems in bug repositories [5].

Before the researches on bug triage, Podgurski, et al. propose a clustering approach to gather the bug reports with the similar errors [14], which can be viewed as the first learning approach for bug reports. They focus on the stack traces in bug reports and analyze the causes of program failures by applying the unsupervised learning approach.

## III. SEMI-SUPERVISED BUG TRIAGE

A reasonable solution for the deficiency of labeled bug reports is to use the unlabeled ones by semi-supervised learning

approaches. In semi-supervised classification, we utilize the knowledge from the unlabeled bug reports to assist the existing supervised classifier.

### A. Semi-supervised framework of bug triage

In this paper, we address bug triage by a semi-supervised text classification approach with EM according to the classic text classification approach in [13]. EM is an iterative method, which is used for finding maximum likelihood estimates of parameters in probabilistic models. In our semi-supervised bug triage approach, the classifier with EM fills the “missing values” (labels) of unlabeled bug reports and then trains a new classifier with all the labels of bug reports.

For the application of automatic bug triage, a bug triage approach tests an incoming unlabeled bug report by a trained classifier. To evaluate the effect of a classifier, the data set is divided into two sets: training set for building the classifier and test set for measuring the classification accuracy. In semi-supervised approaches, such a training set is further divided into two subsets: labeled subset with labeled bug reports and unlabeled subset with unlabeled ones.

Algorithm 1 presents the framework of training this semi-supervised approach with EM. There are two basic phases in the framework. One phase is to train a classifier with labeled bug reports; the other phase is EM with both labeled and unlabeled bug reports. The phase of EM iterates two kernel steps. In expectation-step (E-step), the approach evaluates and labels the bug reports in unlabeled subset; in maximization-step (M-step), the approach rebuilds a new classifier with the labels of all the bug reports. The iterations of building classifiers repeat until the performance of classifiers does not improve.

<b>Algorithm 1.</b> Framework of training a semi-supervised classifier with EM
<b>Input:</b> labeled subset $R_l$ and unlabeled subset $R_u$ of bug reports, set of developers $D$
<b>Output:</b> classifier $\theta$ for semi-supervised bug triage
Build a basic classifier $\theta$ supervisedly from bug reports in $R_l$ .
Loop while classifier $\theta$ improves
E-step. Use classifier $\theta$ to evaluate each bug reports in $R_u$ . Label bug reports in $R_u$ .
M-step. Rebuild classifier $\theta$ with bug reports in $R_l$ and $R_u$ .

Most of the supervised approaches can be employed as the basic classifier in the first phase. In this paper, we apply NB classifier due to the following reasons. First, NB is a probability classification framework which can perform well on the text form of bug reports; second, a common method for providing the recommendation is to sort them with probability; third, it is easy to extend EM with a probability weight based on NB.

To enhance the supervised classifier for bug triage, a recommendation list can be employed to enlarge the set of relevant developers. According to a recommendation list of size  $n$ , the top- $n$  developers can be recommended as the

relevant ones instead of only one best developer. This technology is a common strategy on bug triage [1][3][11]. In this paper, we incorporate a WRL into EM to add the weights for multiple relevant developers while training a classifier. The mechanism of these weights provides probabilistic labels for unlabeled bug reports.

### B. NB classifier

NB classifier is the first approach when treating bug triage as text classification [7]. For further discussion on the semi-supervised approach, we briefly restate the classification framework of NB on bug triage as follows.

Given a set of bug reports  $R = \{r_1, r_2, \dots, r_{|R|}\}$  and a set of developers  $D = \{d_1, d_2, \dots, d_{|D|}\}$ , the task of bug triage is to assign a relevant developer for an incoming bug report  $r$ . For a given bug report  $r_i$ , a classifier  $\Phi$  (parameterized on  $\theta$ ) provides a developer  $d_j$  which can maximize  $P(d_j | r_i; \theta)$  for all  $d_j \in D$ . Thus, the task of building a NB classifier is to calculate the posterior probability  $P(d_j | r_i; \theta)$  and to choose the developer with maximum  $P(d_j | r_i; \theta)$ . With an application of Bayes' theorem, the posterior probability for a given report  $r_i$  is

$$P(d_j | r_i; \theta) = \frac{P(d_j | \theta)P(r_i | d_j; \theta)}{P(r_i | \theta)} \propto P(d_j | \theta)P(r_i | d_j; \theta) \quad (1)$$

The prior probability can be estimated from the training set,

$$P(d_j | \theta) = \frac{\sum_{i=1}^{|R|} P(d_j | r_i)}{|R|} \quad (2)$$

where  $P(d_j | r_i) = \{1, 0\}$  from the training set, i.e., if the label of  $r_i$  is  $d_j$ ,  $P(d_j | r_i) = 1$ ; else  $P(d_j | r_i) = 0$ . The list of words is  $W = \{w_1, w_2, \dots, w_{|W|}\}$  for all the text of bug reports. A NB classifier simplifies the calculation of  $P(r_i | d_j; \theta)$  under the assumption that the words are independently and identically distributed (i.i.d.). Thus, the likelihood probability can be estimated as,

$$P(r_i | d_j; \theta) = \prod_{k=1}^{|r_i|} P(w_k | d_j; \theta)^{N_{ki}} \quad (3)$$

where  $|r_i|$  is the number of the words in the bug report  $r_i$ ,  $N_{ki}$  is the occurrences of word  $w_k$  in  $r_i$ , and

$$P(w_k | d_j; \theta) = \frac{\sum_{i=1}^{|R|} N_{ki} P(d_j | r_i)}{\sum_{m=1}^{|W|} \sum_{i=1}^{|R|} N_{mi} P(d_j | r_i)} \quad (4)$$

To train a classifier, NB estimates (2) and (4) with the training set; to predict a relevant developer for an incoming bug report  $r$ , all the  $P(d_j | r; \theta)$  are calculated with (1) and (3). In practice, a Laplace smoothing is applied to (2) and (4) to avoid zero probabilities. Due to the limitation of paper length, we do not present the formulae after smoothing.

### C. EM in semi-supervised bug triage

The semi-supervised approach with EM depends on the assumption that the data are generated by a mixture model, and there is a correspondence between mixture components and classes [13]. In semi-supervised triage with EM, some steps of

the NB classifier are modified to adapt the unlabeled bug reports.

In E-step, (1) is still used to give the probabilities of labels of bug reports; meanwhile in M-step, the (2) and (4) can be modified to be

$$P(d_j | \theta) = \frac{\sum_{i=1}^{|R|} \Lambda(i) P(d_j | r_i)}{|R_l| + \lambda |R_u|} \quad (5)$$

$$P(w_k | d_j; \theta) = \frac{\sum_{i=1}^{|R|} \Lambda(i) N_{ki} P(d_j | r_i)}{\sum_{m=1}^{|W|} \sum_{i=1}^{|R|} \Lambda(i) N_{mi} P(d_j | r_i)} \quad (6)$$

where  $R_l$  and  $R_u$  are the labeled subset and the unlabeled subset, respectively. The weight factor of labels is

$$\Lambda(i) = \begin{cases} 1 & \text{if } r_i \in R_l \\ \lambda & \text{if } r_i \in R_u \end{cases} \quad (7)$$

where  $\lambda$  is a constant value and  $0 \leq \lambda \leq 1$ . Obviously, if  $\lambda = 0$ , the classifier degenerates into a NB classifier. The classifier is a basic EM for  $\lambda = 1$ , which treats the weights of bug reports in unlabeled subset as those in labeled subset. The classifier changes into a weighted EM for  $0 < \lambda < 1$ , which treats the bug reports in unlabeled subset with fewer weights than those in labeled subset. The mechanism of weight factor augments the basic EM with the distinction of the labeled and unlabeled bug reports [13]. In practice, the value of  $\lambda$  can be estimated by cross-validation method for parameter selection [18]. When the classifier parameters do not improve any more, EM terminates its iterations.

### D. Weighted recommendation list

Similar as NB classifier with a recommendation list, we design a WRL to guide the M-step of EM. The usage of a recommendation list is to provide a developer list for the decision by a triager; instead, WRL is employed to provide weights for promoting the iterations of EM. With this WRL, we add the weights of multiple developers without maximum posterior probability for bug reports in unlabeled subset.

Algorithm 2 presents the process of training the semi-supervised classifier with a WRL. To implement this extension of EM, E-step recommends the developers with first top- $n$  posterior probability for the bug reports in unlabeled subset. In other words, the top- $n$  developers are probabilistically labeled for each bug report in unlabeled subset in E-step. The sum of probabilities of developers in the list is one for a bug report. Then in M-step,  $P(d_j | r_i)$  is extended as the function  $\Gamma(d_j, r_i, q_{ji})$ ,

$$\Gamma(d_j, r_i, q_{ji}) = \begin{cases} \frac{2^{n-q_{ji}}}{\sum_{d_j \in L_i} 2^{n-q_{ji}}} = \frac{2^{n-q_{ji}}}{2^n - 1} & \text{if } d_j \in L_i \\ 0 & \text{if } d_j \notin L_i \end{cases} \quad (8)$$

where  $L_i$  is a developer list with top- $n$  posterior probability of  $r_i$  and  $q_{ji}$  is the ranking position of the developer  $d_j$  for  $r_i$  in  $L_i$  with  $1 \leq q_{ji} \leq n$  (this ranking is based on the posterior probability). For  $r_i \in R_l$ , the size of recommendation list can be denoted as  $n = 1$ , i.e., the label of  $r_i$  in the labeled subset is still

marked as its original relevant developer. Thus,  $\Gamma(d_j, r_i, q_{ji}) = 1$ , if and only if  $d_j$  is the relevant developer of  $r_i$  in the labeled subset. With the extension of (8), (5) and (6) can be formed as

$$P(d_j | \theta) = \frac{\sum_{i=1}^{|R_l|} \Lambda(i) \Gamma(d_j, r_i, q_{ji})}{|R_l| + \lambda |R_u|} \quad (9)$$

$$P(w_k | d_j; \theta) = \frac{\sum_{i=1}^{|R_l|} \Lambda(i) N_{ki} \Gamma(d_j, r_i, q_{ji})}{\sum_{m=1}^{|W|} \sum_{i=1}^{|R_l|} \Lambda(i) N_{mi} \Gamma(d_j, r_i, q_{ji})} \quad (10)$$

Thus, in every M-step, the weights of multiple developers are calculated for rebuilding the classifier. In a developer list for an unlabeled bug report, the developer with larger posterior probability can add larger weights for training the classifier. This extension of M-step provides the feasibility for making EM adapt for the semi-supervised bug triage.

<b>Algorithm 2.</b> Training NB classifier with EM and a WRL
<b>Input:</b> labeled subset $R_l$ and unlabeled subset $R_u$ , set of developers $D$ , size $n$ of recommendation list
<b>Output:</b> classifier $\theta$ for semi-supervised bug triage
Select $\lambda$ for weight factor of $R_u$ by cross-validation.
Build the NB classifier $\theta$ from the bug reports in $R_l$ , while calculating $P(d_j   \theta)$ with (2) and $P(w_k   d_j; \theta)$ with (4).
Loop while classifier $\theta$ does not improve.
E-step. Use the classifier $\theta$ to estimate the posterior probability of bug reports in $R_u$ , $P(d_j   r; \theta)$ with (1). Record each list $L_i$ of $r_i$ , and $q_{ji}$ for developer $d_j$ in $L_i$ .
M-step. Rebuild classifier $\theta$ , with bug reports in $R_l$ and $R_u$ , by calculating $P(d_j   \theta)$ with (9) and $P(w_k   d_j; \theta)$ with (10).

## IV. EXPERIMENTS

### A. Data preparation

We evaluate the semi-supervised bug triage approach with Bugzilla on Eclipse in the experiments. We take the bug reports with the id from 150001 to 170000 as the data set (The XML form of bug reports can be found in MSR Mining Challenge 2008 [17]). To apply the algorithm described in Section III, it is necessary to preprocess the bug reports as numerical values.

To label the relevant developer for bug reports, a direct method is to identify the developer in assigned-to field. The assigned-to field is a part of a bug report for marking the first relevant developer to fix the bug. This field is filled by triagers when a user of the bug repository submits a bug. However, the developer, who really solves the bug, is not always in agreement with the value of assigned-to field. To solve the problem of this confused field, we follow some previous works [3][7] to label the bug reports with a set of heuristics [2]. Moreover, during the step of labeling bug reports, we remove the bug reports without resolved status, verified status, fixed resolution or duplicate resolution. The status or resolution of a bug report indicates the current status in the life cycle. The removing is not necessary for any bug triage approach

(including ours), but we tend to keep the really resolved bug reports. We have 10747 bug reports left after removing. The capability of data set is close to some classic literature [3][7][12].

To extract the numerical values for the bug reports, we select the text of the short description and the first long description to describe a bug report. The reason for this selection is that the triager will face these to describe the details of an incoming bug report [12]. The text form of both short and long description can be tokenized as a list of words and converted into a vector based on words. Thus, every bug report employs such a word vector for recording the counts of words.

Before converting the bug reports, we remove the words in the stoplists and the non-alphabetic tokens to reduce the word vector space [9]. The stoplists store the words with high frequency, which can express little meanings in text. And the non-alphabetic tokens always stand for the specialized words appeared in few bug reports. In addition, we do not use the stemming technology to identify the words with various grammatical suffixes. Many approaches on bug triage employ no stemming since it is not effective in distinguishing bug reports [3][5][7].

After the above steps, numerous words are still left in the word vectors. We remove the words with low frequency to reduce the vector space according to [7]. The words with low frequency can only influence few bug reports and do not provide sufficient information for training a classifier. Similarly, we remove the developers within low frequency to avoid the retired developers according to [3][5][7]. We generated three data sets (in Table I) for comparison after removing the developers who fix less than 10, 30, and 50 bug reports, respectively.

### B. Experimental results

We implement all the approaches with Java (JDK 1.6) in our experiments.

To train and test the classifier effectively, we make the developers in training set, labeled subset and unlabeled subset follow the same probability distribution. For every experiment, the first 5% bug reports of each developer are selected as labeled subset in training set; the following 20% are test set; and the other 75% are unlabeled subset of training set. The unlabeled subset is only used when training a semi-supervised classifier.

In the experimental results, we report the performance of the approaches with the classification accuracy. For a bug triage with a recommendation list, we calculate the accuracy as

$$accuracy_n = \frac{\# \text{ of correct relevant developers}}{\# \text{ of bug reports in test set}}, \quad n \geq 1 \quad (11)$$

where  $n$  is the size of the recommendation list. The accuracy is a standard method to measure the performance of bug triage in [5][7][11]. Thus, we do not use other information retrieval metric methods, e.g., precision and recall rates. In our experiments, we abandon the classic evaluation method,  $k$ -fold cross-validation according to the reasons in [3]. In

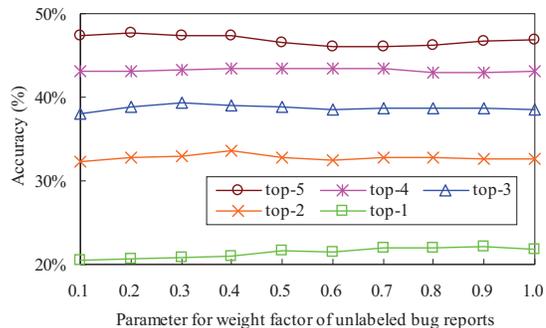
addition, we select the constant parameter  $\lambda$  for the weight factor by cross-validation, which is a standard method of parameter selection in machine learning [18].

We show the classification accuracy while varying the size of the recommendation list on the data sets with different scales in Table I. To simplify the following expression, NBEM is short for NB classifier with EM and NBEM+WRL is short for NBEM with a WRL. We set the size of WRL as the maximum size of recommendation list. From Table I, both NBEM and NBEM+WRL obtain better classification accuracy than NB. For the recommendation list with size 3, this accuracy improves 2% to 5%; for the list with size 5, this accuracy improves 3% to 6%. Considered the original accuracy of NB from 11% to 43%, this improvement is valuable for automatic bug triage. NBEM+WRL also obtains better accuracy than NBEM, from 1% to 3%. It is necessary to note that WRL may hurt performance when the size of the recommendation list is small (e.g., size 2 in the third data set). The reason is that WRL covers the ignored developers in EM, but adds too many weights to the relevant developers in a small recommendation list.

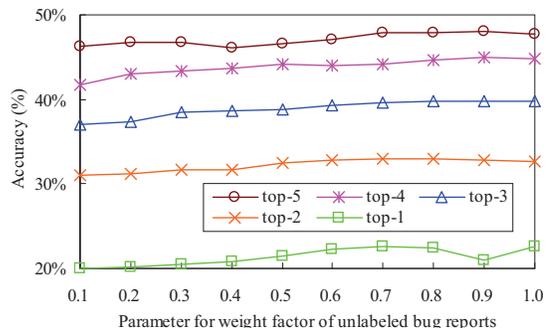
In our semi-supervised approach, EM relies on a constant parameter, which is for the weight factor of unlabeled bug reports. A common method to select such a kind of parameter is cross-validation. It is noted that cross-validation cannot provide the best weight factor for the optimal classification accuracy. In Figure 1, we present the classification accuracy on the third data set in Table I while varying the constant parameter  $\lambda$  for the weight factor of the recommendation lists with different sizes. To obtain high classification accuracy, NBEM (Figure 1a) tends to choose the small values near 0.1 for  $\lambda$ , but NBEM+WRL (Figure 1b) tends to the large values near 1.0. In addition, for a recommendation list with a certain size, the classification accuracy can change up to 3%.

TABLE I. CLASSIFICATION ACCURACY WHILE VARYING THE SIZE OF THE RECOMMENDATION LIST ON THREE DATA SETS WITH DIFFERENT SCALES

Data set	List size	Accuracy (%)		
		NB	NBEM	NBEM+WRL
9324 bug reports and 238 developers	1	11.26	11.82	11.82
	2	17.12	17.91	18.69
	3	20.78	22.07	23.48
	4	23.48	25.62	26.63
	5	26.52	28.10	30.12
6965 bug reports and 110 developers	1	15.47	16.28	16.28
	2	22.65	23.32	24.50
	3	27.61	28.42	29.68
	4	31.46	32.57	33.38
	5	35.09	35.97	37.75
5050 bug reports and 60 developers	1	19.92	21.04	21.04
	2	29.37	33.54	32.83
	3	34.96	39.02	39.74
	4	38.52	43.50	44.92
	5	43.29	47.36	48.07



(a) Varying the parameter for weight factor of NBEM



(b) Varying the parameter for weight factor of NBEM+WRL

Figure 1. Classification accuracy while varying the parameter for the weight factor

## V. DISCUSSION

We discuss three topics about our semi-supervised bug triage approach in this section.

The experimental results show that the classification accuracy can be improved by up to 6%. Actually, although the original accuracy of NB is only 11% to 43%, this improvement is not satisfactory for bug triage. The experimental results are insufficient for the real-world applications. As to our knowledge, three reasons are presented as follows: the quality of bug reports is not good enough; a relevant developer for a bug report is hard to label correctly even for human triager; and the mixture model assumption of EM (in Part C of Section III) is not always satisfied in real-world data. In the literature, Čubranić & Murphy mention the basic semi-supervised approach using EM in the discussion of [7], but Anvik, et al. abandon this approach because the results are worse than NB classifier in the discussion of [3]. The reason is that the EM algorithm may hurt classification accuracy when the unlabeled data are in small scale [13]. An extension of the basic EM is a weight factor or a weighted recommendation list, which can partly reduce the dependency of the above mixture model assumption of EM. Moreover, some other extensions may be used to improve the classification accuracy for the semi-supervised approach.

Many bug triage approaches draw an analogy between bug triage and text classification by processing the bug reports based on the existing text process approaches [3][7][15][16]. It is straightforward to take advantage of the text classification

approaches in bug triage since most of the content of bug reports consist of free text. However, there are two significant differences between bug triage and text classification. First, the scale of data sets on bug triage is too small in comparison to that on text classification; second, bug triage contains more specialized vocabularies than the common text classification. Thus, not all the text classification approaches can be helpful for the bug triage.

As the basic step of bug triage, we label bug reports under the guideline of the heuristics in [2]. Up till now, these heuristics are the most effective method for labeling. These heuristics can be viewed as a decision tree for automatic labeling bug reports according to the knowledge of the life cycle for bug reports. But these heuristics are hard to implement by programs in traditional bug repositories. A direct solution of this problem is to add a new field to the bug repository in the future to mark a developer who really handles the bug.

## VI. CONCLUSION AND FUTURE WORK

Bug triage is a significant step in software development and maintenance. In this paper, we propose a semi-supervised bug triage approach based on NB classifier with EM. This approach improves the classification accuracy with both the labeled and unlabeled bug reports. A WRL is employed to augment the performance via adding the weights of multiple developers when training a classifier. The experimental results demonstrate that our semi-supervised approach improves the classification accuracy of bug triage by up to 6%. During the discussion, we concentrate on three uncertain problems of this approach.

Our future work consists of the following three parts:

1) *Enhancing the classification accuracy via a many-to-one correspondence:* We plan to extend our semi-supervised bug triage approach with EM via the correspondence of many mixture components with one class (many-to-one) proposed in [13]. This correspondence can be considered as the modification of the mixture model of EM for the real-world data. We also plan to combine this many-to-one correspondence with our weighted recommendation list. This combination may improve the accuracy of our semi-supervised bug approach.

2) *Co-training for bug triage:* Apart from naive Bayes with EM, co-training also performs well in semi-supervised learning. Co-training, requiring data with two views, maintains disjoint feature spaces with multiple classifiers on both labeled and unlabeled data. We expect co-training can be applicable for bug triage to avoid the lack of labeled bug reports.

3) *Integration automatic bug triage with a bug repository:* To date, there is no bug triage plug-in component combining with the bug repository in real-world applications. We plan to implement a plug-in component for bug repositories to evaluate automatic bug triage and collect extra information for further researches on bug repositories.

## ACKNOWLEDGMENT

Many thanks to Dr. Thomas Zimmermann with Microsoft Research for sharing bug reports of Eclipse in MSR Mining Challenge 2008. We thank Dr. John Anvik with Department of Computer Science, University of Victoria for sharing the heuristics in labeling bug reports.

## REFERENCES

- [1] J. Anvik, "Automating bug report assignment," Proc. Intl. Conf. Software Engineering (ICSE 06), ACM, 2006, pp. 937-940.
- [2] J. Anvik, L. Hiew, and G. C. Murphy, Heuristics for Labeling Bug Reports. <http://www.cs.ubc.ca/labs/spl/projects/bugTriage/assignment/heuristics.html>.
- [3] J. Anvik, L. Hiew, and G. C. Murphy, "Who should fix this bug?," Proc. Intl. Conf. Software Engineering (ICSE 06), ACM, 2006, pp. 361-370.
- [4] N. Bettenburg, S. Just, A. Schröter, C. Weiss, R. Premraj, and T. Zimmermann, "What makes a good bug report?," Proc. ACM SIGSOFT Symp. Foundations of Software Engineering (FSE 08), ACM, 2008, pp. 308-318.
- [5] N. Bettenburg, R. Premraj, T. Zimmermann, and S. Kim, "Duplicate bug reports considered harmful... really?," Proc. IEEE Conf. Software Maintenance (ICSM 08), IEEE Computer Society, Sep. 2008, pp. 337-345.
- [6] Bugzilla, <http://www.bugzilla.org/>.
- [7] D. Čubranić and G. C. Murphy, "Automatic bug triage using text categorization," Proc. Intl. Conf. Software Engineering & Knowledge Engineering (SEKE 04), 2004, pp. 92-97.
- [8] Eclipse, <http://www.eclipse.org/>.
- [9] W. B. Frakes and R. Baeza-Yates, Information Retrieval: Data Structures and Algorithms. Englewood Cliffs, NJ: Prentice-Hall, 1992.
- [10] N. Jalbert and W. Weimer, "Automated duplicate detection for bug tracking systems," Proc. IEEE Intl. Conf. Dependable Systems & Networks (DNS 08), IEEE Computer Society, 2008, pp. 52-61.
- [11] G. Jeong, S. Kim, and T. Zimmermann, "Improving bug triage with bug tossing graphs," Proc. Joint Meeting European Software Engineering Conf. & ACM SIGSOFT Symp. Foundations of Software Engineering (ESEC-FSE 09), ACM, 2009, pp. 111-120.
- [12] D. Matter, A. Kuhn, and O. Nierstrasz, "Assigning bug reports using a vocabulary-based expertise model of developers," Proc. IEEE Working Conf. Mining Software Repositories (MSR 09), IEEE Computer Society, pp. 131-140.
- [13] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using EM," Machine Learning. Hingham, MA, vol. 39, pp. 103-134, May 2000.
- [14] A. Podgurski, D. Leon, P. Francis, W. Masri, M. Minch, J. Sun, and B. Wang, "Automated support for classifying software failure reports," Proc. Intl. Conf. Software Engineering (ICSE 03), IEEE Computer Society, 2003, pp. 465-475.
- [15] P. Runeson, M. Alexanderson, and O. Nyholm, "Detection of duplicate defect reports using natural language processing," Proc. Intl. Conf. Software Engineering (ICSE 07), IEEE Computer Society, 2007, pp. 499-510.
- [16] X. Wang, L. Zhang, T. Xie, J. Anvik, and J. Sun, "An approach to detecting duplicate bug reports using natural language and execution information," Proc. Intl. Conf. Software Engineering (ICSE 08), ACM, 2008, pp. 461-471.
- [17] T. Zimmermann, Mining Software Repositories (MSR) Mining Challenge 2008. [http://pag.csail.mit.edu/msr\\_challenge2008/eclipse-bugs-000001-213000.zip](http://pag.csail.mit.edu/msr_challenge2008/eclipse-bugs-000001-213000.zip).
- [18] P. Zhang, "Model selection via multifold cross validation," The Annals of Statistics. Beachwood, OH, vol. 21, pp. 299-313, March 1993.

# Ensemble Feature Selection Technique for Software Quality Classification

Huanjing Wang  
Western Kentucky University  
Bowling Green, Kentucky 42101  
huanjing.wang@wku.edu

Taghi M. Khoshgoftaar  
Florida Atlantic University  
Boca Raton, Florida 33431  
taghi@cse.fau.edu

Kehan Gao  
Eastern Connecticut State University  
Willimantic, Connecticut 06226  
gaok@easternct.edu

**Abstract**—Feature selection is an important data preprocessing step in data mining applications. In this paper, we studied six filter-based feature ranking techniques and an ensemble technique using rank ordering of the features (mean or median). The best features are selected through either the individual ranker or an ensemble approach. Then the reduced data set is used to build classification models using three well-known classifiers within the domain of software quality engineering. The classification accuracy was evaluated in terms of the AUC (Area Under the Receiver Operating Characteristic Curve) performance metric. Results demonstrate that the ensemble technique performed better overall than any individual ranker and also possessed better robustness. The empirical study also shows that variations among rankers, learners and software projects significantly impacted the classification outcomes.

## I. INTRODUCTION

Software quality data (metrics) that are collected during the software development process include valuable information about a software project. Software practitioners strive to improve software quality by constructing quality prediction models using software metrics (attributes or features) and data mining techniques. Feature selection can be broadly classified as *feature ranking* and *feature subset selection*. *Feature ranking* sorts the attributes according to their individual predictive power, while *feature subset selection* finds subsets of attributes which collectively have good predictive power. Feature selection techniques can also be categorized as *filters*, *wrappers* and *hybrids*. *Filters* are algorithms in which a feature set is selected without involving any learning algorithm. *Wrappers* [1], [2] are algorithms that use feedback from a learner to select features. The *hybrids* [3] are combination of filters and wrappers, and thus exploit advantage from both methods. This work focuses on filter-based feature ranking.

Using a single method for feature selection may generate local optima. Ensemble methods attempt to combine multiple methods of different types instead of using a single one. These methods are an emerging area in machine learning research. Ensemble feature selection combines multiple feature subsets to get a single final feature subset [4], [5]. The final feature subset will be used for future learning. There are two types of ensemble feature selection: ensembles of multiple feature selection techniques and an ensemble of a single feature selection. Ensembles of multiple feature selection combine outcomes of various feature selection techniques. The feature selection can be feature ranking, feature subset selection, or combinations of feature ranking and feature subset selection techniques. An ensemble of a single feature selection combines various outcomes (multiple views) of a single feature selection technique, where feature selection is performed on different subsamples of the original data set, including sampling, bootstrap or n-fold cross-validation.

The focus of this paper is to evaluate six commonly used filter-based rankers (feature selection techniques) and our proposed ensemble approach. The latter consists of feature ranking techniques

followed by a combination method. The six feature ranking techniques considered are: chi-square (CS), information gain (IG), gain ratio (GR), two forms of the ReliefF algorithm (RF and RFW), and symmetrical uncertainty (SU). Two separate combination methods were used in the ensemble technique. One uses mean (EM) of ranking scores, while the other uses median (ED). These methods were evaluated using software measurement data in our case study, including the four consecutive releases of a very large telecommunications system and three data sets from NASA project KC1. The software quality prediction models were built using three different classification algorithms (classifiers or learners): naïve Bayes (NB), *K*-nearest neighbors (KNN), and support vector machine (SVM). The classification accuracy was evaluated in terms of the AUC performance metric. The results demonstrate that the performance of filter-based rankers may be significantly influenced by the data set and learner used in classification, while the ensemble approach generally performed better than the individual rankers and presented more stable and robust behavior with respect to various data sets and different learners.

The key contributions of this research are:

- Implementation and investigation of the ensemble technique. Although six filter-based feature ranking techniques and three learners are used in the study, any number of feature ranking techniques and learners can be used. This is also the first paper to investigate ensembles of such diverse ranking techniques.
- The use of imbalanced data from real-world software systems. Such an extensive range of feature ranking and ensemble approaches for software quality prediction (and other application domains) is unique to this study.

## II. RELATED WORK

This section provides a brief coverage of key ensembles feature selection works in the area of data mining and software engineering. An exhaustive discussion is avoided due to space considerations.

One of the main problems which needs to be considered when building an ensemble model is diversity. Diversity may be achieved through using different data sets, feature subsets, or classifiers. Lee [6] and Rokach et al. [7] combined outcomes of various non-ranker filter-based feature subset selection techniques. Souza et al. [8] and Loscalzo et al. [9] studied the ensemble of a single feature subset selection technique. Souza et al. applied the ensemble approach to feature selection by proposing a systematic way of combining various outcomes of a feature selection algorithm. Loscalzo et al. applied an ensemble approach to consensus feature groups.

Very limited research exists on ensemble feature ranking. Ensemble feature ranking can be classified as either ensembles of multiple feature ranking techniques or an ensemble of a single feature ranking technique. More recently, one approach was proposed by Saeys et

al. [5] using an ensemble of a single feature ranking technique. They studied an ensemble of a feature ranking technique which aggregates the top 1% of best features of the rankings. The rankers used in the study included two filter-based feature ranking techniques, SU, and RELIEF; and two embedded feature ranking techniques, Random Forests with 10 trees and SVM\_RFE. Each feature ranking technique was repeatedly applied on different bootstrapped samples with 40 bags of the same data set. The classifiers used in the study included SVM, Random Forests with 50 trees and 5-NN. Their results showed that the best classification results (accuracy) were obtained by using SVM classifier and an ensemble of SVM\_REF.

A recent study on ensembles of multiple feature ranking techniques was done by Olsson and Oard [4]. Their work considered combinations of three commonly-used filter-based feature ranking techniques, including document frequency thresholding, IG, and chi-square method ( $\chi_{max}^2$  and  $\chi_{avg}^2$ ) for text classification problems. To create ensembles, two or more ranking lists were combined. They concluded that the best performing combination was  $\chi_{max}^2$  and  $\chi_{avg}^2$ . The experimental results showed that the ensemble approach could achieve higher peak R-precision and  $F_1$  than a non-combined feature ranker at statistically significant levels when the classifier was built using 100-Nearest Neighbor. The technique we propose in this work is much more general than that of Olsson and Oard. Their work built ensembles for the text classification problem, while ours does not require binary data, making it more general and useful on any numeric data set. We combined six diverse commonly used filter-based rankers instead of three. Moreover, the ensemble technique presented in this study can be used with any number of feature ranking techniques which we leave to future work.

### III. METHODOLOGY

#### A. Filter-based Feature Ranking

Filter-based feature ranking techniques rank features independently without involving any learning algorithm. Feature ranking consists of scoring each feature according to a particular method, then selecting features based on their scores. This work used some commonly used filter-based feature ranking techniques including chi-square, information gain, gain ratio, symmetrical uncertainty, and ReliefF. For specific algorithmic details on these techniques, the reader is referred to the various cited references.

The chi-square (CS) [10] test is used to examine if there is ‘no association’ between two attributes, i.e., whether the two variables are independent. CS is more likely to find significance to the extent that (1) the relationship is strong, (2) the sample size is large, and/or (3) the number of values of the two associated features is large.

Information gain, gain ratio, and symmetrical uncertainty are measures based on the concept of entropy, which is based on information theory. Information gain (IG) [11] is the information provided about the target class attribute Y, given the value of independent attribute X. Information gain measures the decrease of the weighted average impurity of the partitions, compared with the impurity of the complete set of data. A drawback of IG is that it tends to prefer attributes with a larger number of possible values; that is, if one attribute has a larger number of values, it will appear to gain more information than those with fewer values, even if it is actually no more informative. One strategy to counter this problem is to use the gain ratio (GR), which penalizes multiple-valued attributes. Symmetrical uncertainty (SU) [12] is another way to overcome the problem of IG’s bias toward attributes with more values, doing so by dividing IG by the sum of the entropies of X and Y.

Relief is an instance-based feature ranking technique [13]. ReliefF is an extension of the Relief algorithm that can handle noise and multi-class data sets. When the ‘weightByDistance’ (weight nearest neighbors by their distance) parameter is set as default (false), the algorithm is referred to as RF; when the parameter is set to true, the algorithm is referred to as RFW.

#### B. Ensemble Method

We investigated the use of ensembles of multiple feature ranking techniques. These ranking techniques are combined to yield more stable and robust results. There are two essential steps in creating a single feature ranking list from multiple ranking lists. First, a set of different ranking lists is created using corresponding filter-based rankers and input to the next combining step; second, these ranking lists are integrated using rank ordering of the features. Diversity can be achieved by using various rankers. The combining methods used in the study include mean (EM) and median (ED). For mean combination, each feature’s score is determined by the average of the ranking scores of the feature in each ranking list, while for median combination, we give each feature’s combining score the median score in all ranking lists. Mean combination (average rank) has been used by Olsson and Oard [4]. We used median combination (median rank) in the paper as well.

#### C. Classifiers

Software quality prediction models were built with three well-known classification algorithms including naïve Bayes (NB),  $K$ -nearest neighbors (KNN), and support vector machine (SVM). These were selected because of their common use in software engineering and other data mining applications. Unless stated otherwise, we use default parameter settings for the different learners as specified in the WEKA [14] data mining tool. Parameter settings are changed only when a significant improvement in performance is obtained.

Naïve Bayes classifier (NB) [15] utilizes Bayes’s rule of conditional probability and is termed ‘naïve’ because it assumes conditional independence of the features.  $K$ -nearest neighbors (KNN) [16] classifier, also called instance-based learning, uses distance-based comparisons. The choice of distance metric is critical. KNN was built with changes to three parameters. The ‘distanceWeighting’ parameter was set to ‘Weight by 1/distance’, the ‘kNN’ parameter was set to ‘5’, and the ‘crossValidate’ parameter was turned on (set to ‘True’). The support vector machine (SVM) [17], also called SMO in WEKA, had two changes to the default parameters: the ‘complexity constant c’ was set to 5.0 and ‘build Logistic Models’ was set to true. By default, a linear kernel was used.

#### D. Performance Metric

Traditional performance measures such as F-measure, overall classification accuracy, or its complement, misclassification rate, are inappropriate when dealing with the classification of imbalanced data. In a domain such as software quality prediction, the number of *fp* (fault-prone) modules is much lower than the *nfp* (not fault-prone) modules. Instead, we used a performance metric that considers the ability of a classifier to differentiate between the two classes: the area under the ROC (Receiver Operating Characteristic) curve (AUC). It has been shown that AUC has lower variance and is more reliable than other performance metrics (such as precision, recall, F-measure) [18].

The AUC is a single-value measurement, whose value ranges from 0 to 1. The ROC curve is used to characterize the trade-off between hit (true positive) rate and false alarm (false positive) rate [19]. A classifier that provides a large area under the curve is preferable over a

TABLE I  
SOFTWARE DATA SETS CHARACTERISTICS

Data set	#Metrics	#Modules	%fp	%nfp
SP1	42	3649	6.28%	93.72%
SP2	42	3981	4.75%	95.25%
SP3	42	3541	1.33%	98.67%
SP4	42	3978	2.31%	97.69%
KC1-5	63	145	24.83%	75.17%
KC1-10	63	145	14.48%	85.52%
KC1-20	63	145	6.90%	93.10%

classifier with a smaller area under the curve. Traditional performance metrics consider only the default decision threshold of 0.5. ROC curves illustrate the performance across all decision thresholds. A perfect classifier provides an AUC that equals 1.

#### IV. EXPERIMENTS

##### A. Experimental Data Sets

Experiments conducted in this study used software metrics and defect data collected from real-world software projects, a very large telecommunications software system (denoted as LLTS) and NASA software project KC1. LLTS contains data from four consecutive releases, which are labeled as SP1, SP2, SP3 and SP4. The software measurement data sets consist of 42 software metrics, including 24 product metrics, 14 process metrics, and four execution metrics [20]. NASA project KC1 [21] contains three data sets KC1-5, KC1-10, KC1-20. After preprocessing, each of these data sets includes 63 attributes. The seven data sets used in this work represent software projects of different sizes with different levels of imbalance. Table I lists the characteristics of the seven data sets utilized in this work.

##### B. Experimental Design

We first used six filter-based rankers and our proposed ensemble technique to select the subsets of attributes. We rank the features and select the top  $\lceil \log_2 n \rceil$  features according to their respective scores, where  $n$  is the number of independent features for a given data set. The reasons why we select the top  $\lceil \log_2 n \rceil$  features include (1) no general guidance has been found in related literature on the number of features that should be selected when using a feature ranking technique; (2) a software engineering expert with more than 20 years experience recommended selecting  $\lceil \log_2 n \rceil$  number of metrics for software quality prediction; (3) a preliminary study showed that  $\lceil \log_2 n \rceil$  is appropriate for various learners; and (4) a recent study [22] showed that it was appropriate to use  $\lceil \log_2 n \rceil$  as the number of features when using WEKA to build Random Forests learners for binary classification in general and imbalanced data sets in particular. Subsequently, the selected features were used to build classification models.

The experiments were conducted to discover the impact of (1) six commonly used filter-based rankers vs. ensemble technique; (2) three different learners; and (3) two different software projects from the software quality prediction domain. We implemented the ensemble technique in WEKA and used it for the defect prediction model building and testing process. In the experiments, ten runs of five-fold cross-validation were performed. The five results from the five folds were then combined to produce a single estimation. In total, 8,400 models were evaluated during our experiments.

##### C. Experimental Results

The classification models were evaluated in terms of the AUC performance metric. All the results are reported in Table II. Note

that each value presented in the table is the average over the ten runs of five-fold cross-validation outcomes. The best model for each data set is indicated in underlined **boldfaced** print, while the worst performance is boldfaced *italic*. Each value in the table is determined by three dimensions: (1) feature ranking techniques (CS, GR, IG, RF, RFW, SU, EM, and ED); (2) classifiers (NB, KNN and SVM); and (3) data sets (SP1, SP2, SP3, SP4, KC1-5, KC1-10, and KC1-20). We compared the performance of NB, KNN and SVM models built after feature selection. A total of 168 values are included in the three tables.

##### D. Analysis of Results

Table III summarizes the number of best cases (positive numbers) and the number of worst cases (negative numbers) for all the rankers on each individual classifier as well as for all classifiers together. From Table II and III, we can observe the following facts:

- CS performed best for KC1-10 when using the KNN classifiers but worst when using NB classifier. This demonstrates that the performance of a ranker is influenced by the selected classifier. In addition, for a given classifier (for instance NB), RF performed best for SP3 and KC1-5 but worst for KC1-20. This displays that for a given classifier, the same ranker can produce different results for different data sets. In other words, the characteristics of data sets impact on the performance of rankers.
- Among the six feature ranking techniques, GR performed worst. This can be easily seen from the ratio between its numbers of the best and worst cases. RF and RFW showed a considerable number of worst cases even though they also displayed a number of best cases. This indicates that RF and RFW had unstable performance with respect to different classifiers. CS, IG and SU presented less frequently best and also worst cases than other techniques. In fact, they displayed moderate and stable behavior. The ensemble technique compared to each individual ranker (except SU) showed lower number of worst cases (actually no worst cases at all). ED also had the second highest number of best cases among all the ranking techniques.

We also performed a two-way ANOVA test to statistically examine the various effects on performance of the classification models. The two factors were designed as follows. Factor A represents the eight feature selection techniques (CS, GR, IG, RF, RFW, SU, EM and ED); and Factor B represents the three learners (NB, KNN and SVM) used in classifications. The interaction effect  $A \times B$  was also considered in the ANOVA test. A significance level of  $\alpha = 5\%$  was used for all statistical tests. Table IV presents the ANOVA results for the LLTS data sets. Note that the ANOVA tests were performed on the LLTS four releases together. From the table, we can see that the  $p$ -values for the main factors A and B, and the interaction term  $A \times B$  were less than 0.05, indicating the AUC values are not same for all groups in each of the main factors and also influenced by the interaction term  $A \times B$ , i.e., Factor A is different at every level of Factor B, and vice versa.

Additional multiple comparisons for the main factors and interaction term were performed to investigate the difference among the respective groups (levels). The test results are shown in Figure 1, where each sub-figure displays graphs with each group mean represented by a symbol (o) and 95% confidence interval. The summarized results reveal the following:

- For the eight feature selection methods (Figure 1(a)), the two ensemble methods on average outperformed the other six filter-based ranking techniques, among which GR, RF, RFW and SU

TABLE II  
EXPERIMENTAL RESULTS IN TERMS OF AUC

(a) NB								
Data	CS	GR	IG	RF	RFW	SU	EM	ED
SP1	0.7846	<b>0.7346</b>	0.7831	0.7879	<b>0.7882</b>	0.7865	0.7819	0.7822
SP2	0.8108	<b>0.7613</b>	0.8081	0.8053	0.8081	0.7729	0.8109	<b>0.8117</b>
SP3	0.8184	<b>0.7808</b>	0.8118	<b>0.8305</b>	0.8190	0.7882	0.8183	0.8145
SP4	0.7696	<b>0.7519</b>	0.7794	0.7731	0.7735	0.7592	<b>0.8039</b>	0.8031
KC1-5	0.7484	0.7489	<b>0.7438</b>	<b>0.7990</b>	0.7832	0.7468	0.7841	0.7679
KC1-10	<b>0.7513</b>	<b>0.7729</b>	0.7546	0.7585	0.7639	0.7719	0.7676	0.7705
KC1-20	0.8525	0.8669	0.8569	<b>0.8296</b>	<b>0.8987</b>	0.8531	0.8507	0.8521

(b) KNN								
Data	CS	GR	IG	RF	RFW	SU	EM	ED
SP1	0.7570	<b>0.7139</b>	0.7475	0.7495	0.7489	<b>0.7600</b>	0.7545	0.7545
SP2	0.7800	0.7515	0.7721	<b>0.7221</b>	0.7255	0.7796	0.7791	<b>0.7802</b>
SP3	0.7879	<b>0.7298</b>	0.7802	0.7898	0.7704	0.7602	<b>0.7921</b>	0.7865
SP4	0.7913	<b>0.6853</b>	<b>0.7967</b>	0.7631	0.7665	0.7433	0.7897	0.7895
KC1-5	0.7915	<b>0.7590</b>	0.7749	<b>0.7923</b>	0.7782	0.7693	0.7875	0.7711
KC1-10	<b>0.8449</b>	0.8219	0.8180	0.7078	<b>0.6963</b>	0.8247	0.8246	0.8364
KC1-20	0.8659	<b>0.8896</b>	<b>0.8495</b>	0.8674	0.8867	0.8761	0.8721	0.8744

(c) SVM								
Data	CS	GR	IG	RF	RFW	SU	EM	ED
SP1	0.6401	0.6532	0.6651	<b>0.6708</b>	<b>0.6368</b>	0.6632	0.6386	0.6538
SP2	<b>0.7060</b>	0.6577	0.6628	<b>0.6357</b>	0.6386	0.6572	0.6872	0.6905
SP3	0.6456	<b>0.6294</b>	0.6470	0.6341	0.6611	0.6601	0.6573	<b>0.7084</b>
SP4	0.6529	<b>0.6247</b>	0.6531	0.6248	0.6423	0.6399	0.6543	<b>0.6804</b>
KC1-5	0.7935	<b>0.7649</b>	0.7844	<b>0.8201</b>	0.7988	0.7768	0.7882	0.7819
KC1-10	0.7645	0.7712	0.7740	0.7521	<b>0.6798</b>	<b>0.7813</b>	0.7805	0.7784
KC1-20	0.8382	0.8432	0.8610	<b>0.8119</b>	<b>0.8824</b>	0.8396	0.8412	0.8398

TABLE III  
PERFORMANCE SUMMARIZATION IN TERMS OF AUC

		CS	GR	IG	RF	RFW	SU	EM	ED
NB	# of best	0	1	0	2	2	0	1	1
	# of worst	-1	-4	-1	-1	0	0	0	0
KNN	# of best	1	1	1	0	1	1	1	1
	# of worst	0	-4	-1	-1	-1	0	0	0
SVM	# of best	1	0	0	2	1	1	0	2
	# of worst	0	-3	0	-2	-2	0	0	0
ALL	# of best	2	2	1	5	3	2	2	4
	# of worst	-1	-11	-2	-4	-3	0	0	0

TABLE IV  
TWO-WAY ANOVA TABLES FOR LLTS DATA SETS

Source	Sum Sq.	d.f.	Mean Sq.	F	<i>p</i> -value
A	0.1859	7	0.0266	22.95	0
B	3.2717	2	1.6358	1413.38	0
A×B	0.0534	14	0.0038	3.29	0
Error	1.0833	936	0.0012		
Total	4.5943	959			

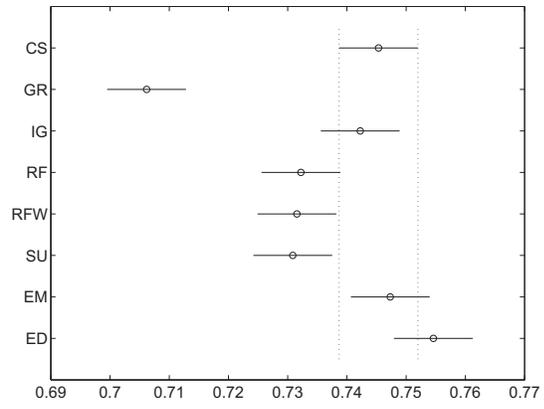
performed significantly worse than the ensemble methods while the other two, CS and IG, performed worse. Of those six filter-based rankers, CS and IG performed best (though still worse than the ensemble techniques), and the rest fared much worse.

- For the three learners (Figure 1(b)), their classification performances were significantly different from each other. NB performed best followed by KNN, then SVM.
- For interaction A×B, 24 groups produced by three learners combined with eight feature selection techniques are presented. The ensemble methods are highlighted with thick lines on the 95% interval in Figure 1(c). It can be seen that Factor A was

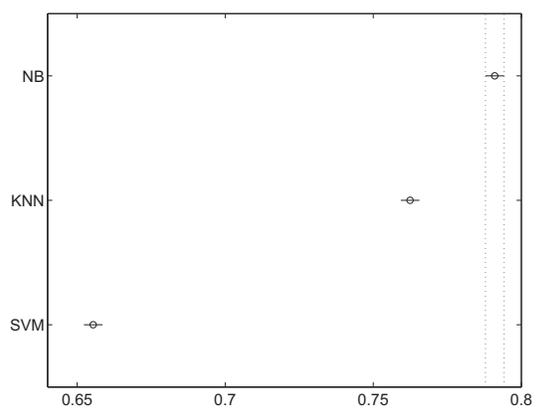
different at every level (group) of Factor B. For example, GR performed significantly worse than the other seven methods when the KNN learner was used, but did significantly worse than six of them when NB was used and only one of them when SVM was used. This demonstrates that the interaction has great impact on the results.

Table V presents the ANOVA result for the three KC1 data sets. The result demonstrates that there was no significant distinction between any pair of the group means for Factor A since the *p*-value (0.38) was greater than 0.05, while an obvious difference existed in at least one paired of group means for Factor B, because the *p*-value was zero. For interaction A×B, the conclusion was similar to the one obtained from the LLTS data sets, namely, Factor A was different at every level of Factor B. The multiple comparison tests are presented in Figure 2. The results also demonstrate that the ensemble methods on average were better than the other individual filter-based feature ranker, though not significant. In addition, the KNN learner significantly outperformed the other two learners, NB and SVM for the KC1 data sets. No distinction was found between the performances of the NB and SVM classifiers. In addition, from Figure 2(c), we can observe that the ensemble methods were more stable than the other individual ranking techniques. For instance, RFW presented the best performance among the eight feature selection techniques when the NB learner was used to build models, but showed the worst performance when the KNN and SVM learners were used to do so.

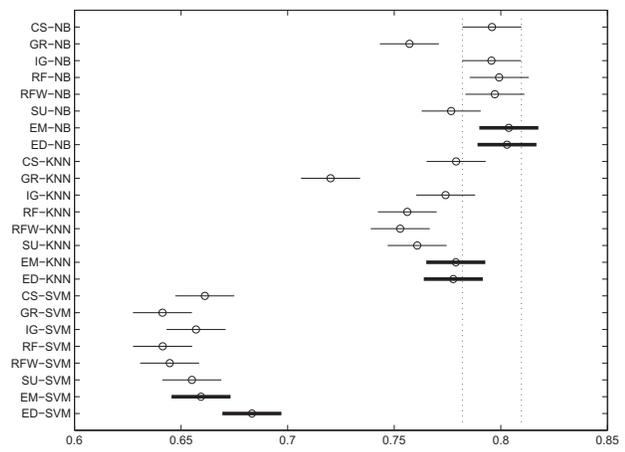
In summary, the ensemble technique with two different combination methods (mean and median), compared to each individual ranker, consistently performed very well, similar or superior to the other methods. In addition, the results indicate that the ensemble technique had better robustness. It is worthwhile to note that type of the ranker, classifier and software project (data set) played a key role in building



(a) Factor A



(b) Factor B

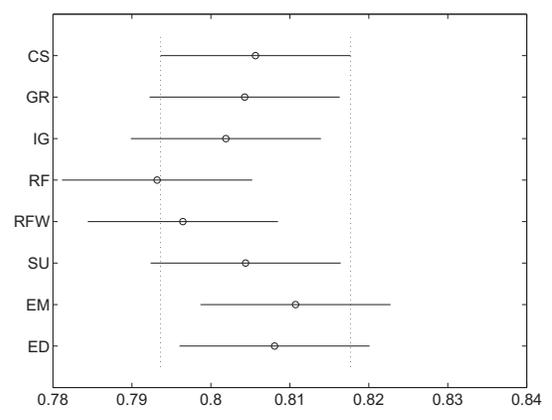


(c) Factor A x B

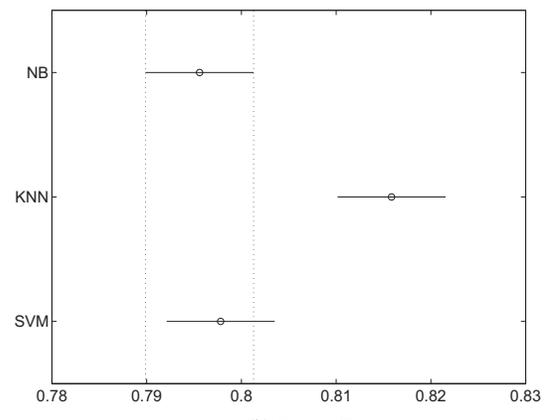
Fig. 1. LLTS: Multiple Comparison in Terms of AUC

TABLE V  
TWO-WAY ANOVA TABLES FOR KC1 DATA SETS

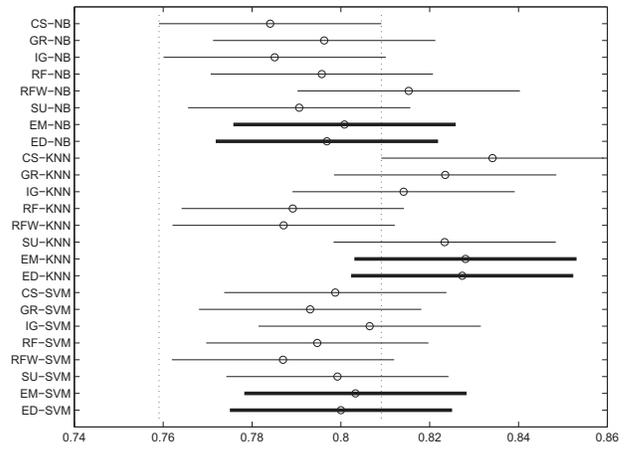
Source	Sum Sq.	d.f.	Mean Sq.	F	p-value
A	0.0213	7	0.0030	1.07	0.3816
B	0.0593	2	0.0296	10.43	0
A x B	0.0752	14	0.0054	1.89	0.0242
Error	1.9764	696	0.0028		
Total	2.1322	719			



(a) Factor A



(b) Factor B



(c) Factor A x B

Fig. 2. KC1: Multiple Comparison in Terms of AUC

a software defect prediction model. Our recent studies [20], [23] also demonstrate that the reduced feature subsets can have better or similar prediction performance compared to the complete set of attributes (original dataset).

## V. CONCLUSION

Filter-based ranker (feature selection) is one of the commonly used methodologies for feature selection. This work has presented detailed experiments using six frequently used rankers and our proposed ensemble approach and applied them to a very large telecommunications software system with four releases and a NASA project KC1 which includes three data sets. The classification models were built using NB, KNN and SVM learners. The classification accuracy was evaluated in terms of the AUC performance metric.

The experimental results show that the performance of rankers (feature selection techniques) may be significantly influenced by the data set and learner used in classification. It is frequently seen that one ranker performs well in a given data set for a particular classifier but becomes very bad when used on a different data set or with a different classifier. This study proposed and investigated an ensemble technique with two different combining methods using rank ordering of the features (mean or median) and results demonstrate that the ensemble technique performed better overall than any individual ranker and also possessed better robustness. The empirical study also shows that variations among rankers, learners and software projects significantly impacted the classification outcomes.

Future work may include experiments using additional data sets from the software engineering domain as well as from other application domains.

## REFERENCES

- [1] W. Altidor, T. M. Khoshgoftaar, and J. V. Hulse, "An empirical study on wrapper-based feature ranking," in *Proceedings of 21st IEEE International Conference on Tools with Artificial Intelligence*, Newark, NJ, USA, Nov. 2-5 2009, pp. 75–82.
- [2] K. Gao, T. M. Khoshgoftaar, and A. Napolitano, "Exploring software quality classification with a wrapper-based feature ranking technique," in *Proceedings of 21st IEEE International Conference on Tools with Artificial Intelligence*, Newark, NJ, USA, Nov. 2-5 2009, pp. 67–74.
- [3] H. Wang, T. M. Khoshgoftaar, K. Gao, and N. Seliya, "Mining data from multiple software development projects," in *Proceedings of 9th IEEE International Conference on Data Mining - Workshops*, 2009, pp. 551–557.
- [4] J. O. S. Olsson and D. W. Oard, "Combining feature selectors for text classification," in *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, New York, NY, USA, 2006, pp. 798–799.
- [5] Y. Saeys, T. Abeel, and Y. Peer, "Robust feature selection using ensemble feature selection techniques," in *ECML PKDD '08: Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases - Part II*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 313–325.
- [6] K. Lee, "Combining multiple feature selection methods," in *Mid-Atlantic Student Workshop on Programming Languages and Systems(MASPLAS'02)*, 2002, pp. 12.1–12.9.
- [7] L. Rokach, B. Chizi, and O. Maimon, "Feature selection by combining multiple methods," in *Advances in Web Intelligence and Data Mining*, 2006, pp. 295–304.
- [8] J. T. de Souza, N. Japkowicz, and S. Matwin, "Stochfs: A framework for combining feature selection outcomes through a stochastic process," in *Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases*, 2005, pp. 667–674.
- [9] S. Loscalzo, L. Yu, and C. Ding, "Consensus group stable feature selection," in *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, New York, NY, USA, 2009, pp. 567–576.
- [10] A. C. Cameron and P. K. Trivedi, *Regression Analysis of Count Data*. Cambridge University Press, 1998.
- [11] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.
- [12] M. A. Hall and G. Holmes, "Benchmarking attribute selection techniques for discrete class data mining," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 6, pp. 1437 – 1447, Nov/Dec 2003.
- [13] K. Kira and L. A. Rendell, "A practical approach to feature selection," in *Proceedings of 9th International Workshop on Machine Learning*, 1992, pp. 249–256.
- [14] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. Morgan Kaufmann, 2005.
- [15] G. H. John and P. Langley, "Estimating continuous distributions in bayesian classifiers," in *Proceedings of Eleventh Conference on Uncertainty in Artificial Intelligence*, vol. 2, San Mateo, 1995, pp. 338–345.
- [16] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Machine Learning*, vol. 6, no. 1, pp. 1573–0565, January 1991.
- [17] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.
- [18] Y. Jiang, J. Lin, B. Cukic, and T. Menzies, "Variance analysis in software fault prediction models," *Software Reliability Engineering, International Symposium on*, vol. 0, pp. 99–108, 2009.
- [19] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, June 2006.
- [20] K. Gao, T. M. Khoshgoftaar, and H. Wang, "An empirical investigation of filter attribute selection techniques for software quality classification," in *Proceedings of the 10th IEEE International Conference on Information Reuse and Integration*, Las Vegas, Nevada, August 10-12 2009, pp. 272–277.
- [21] A. G. Koru, D. Zhang, K. E. Emam, and H. Liu, "An investigation into the functional form of the size-defect relationship for software modules," *IEEE Trans. Software Eng.*, vol. 35, no. 2, pp. 293–304, 2009.
- [22] T. M. Khoshgoftaar, M. Golawala, and J. Van Hulse, "An empirical study of learning from imbalanced data using random forest," in *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence*, vol. 2, Washington, DC, USA, 2007, pp. 310–317.
- [23] H. Wang, T. M. Khoshgoftaar, K. Gao, and N. Seliya, "High-dimensional software engineering data and feature selection," in *Proceedings of 21st IEEE International Conference on Tools with Artificial Intelligence*, Newark, NJ, USA, Nov. 2-5 2009, pp. 83–90.

# Negotiating Software Acquisition Supported by Web Services in a Distributed Software Development Process

Gabriel Costa Silva<sup>1</sup>, Itana Maria de Souza Gimenes<sup>1</sup>, Marcelo Fantinato<sup>2</sup> and Maria Beatriz Felgar de Toledo<sup>3</sup>

<sup>1</sup> *Department of Informatics, State University of Maringá, Brazil*

<sup>2</sup> *School of Arts, Sciences and Humanities, University of São Paulo, Brazil*

<sup>3</sup> *Institute of Computing, University of Campinas, Brazil*

*gabriel\_costasilva@yahoo.com.br, itana@din.uem.br, m.fantinato@usp.br, beatriz@ic.unicamp.br*

## Abstract

*Several factors have motivated organizations to look for partnerships for software development. However, seeking, negotiating and contracting partners for Distributed Software Development (DSD) involve complex issues. This paper proposes a negotiation and contracting process applied to a DSD scenario. It uses PL4BPM, an approach based on web services technology that has electronic contract as its central element. Feature models representing negotiated web services are used as a common representation that can be managed by involved stakeholders.*

## 1. Introduction

Distributed Software Development (DSD) has been facing several challenges, such as: a myriad of technological resources, frequent market changes, a high demand for cost reduction and increasingly short schedules. To cope with these challenges and improve performance, companies are seeking third party contracts, thus focusing on the activities of their expertise area [1]. In order to ensure a reliable inter-organizational cooperation a contract must be established. This contract must define the activities and clauses which state the obligations, permissions and constraints of each party [2].

Web services have been providing an important technological support to third party contracting [3]. A software development activity can be represented as a service. Thus, several organizations can cooperate throughout a software process based on business processes (BP) composed of web services. This cooperation needs to be regulated by an electronic contract (e-contract) which defines quality attributes and specific properties for each contracted service. Such definitions need to be previously negotiated between organizations. During the negotiation phase, organizations establish the services to be contracted, their quality level and properties, in addition to variables such as price [5]. However, negotiation within the electronic context may require a long and complex negotiation process. Thus, a

negotiation support is required to allow organizations to establish e-contracts that regulated DSD.

This paper presents a negotiation process that aims at supporting the establishment of e-contracts. This negotiation process is part of a wider Business Process Management (BPM) approach based on web services and product line concepts [6], which include: reuse, contract establishment, dynamic execution environment and e-contract meta-models. In this paper, our negotiation process is applied to the domain of software acquisition and related services in DSD. The paper is organized as follows: Section 2 presents background on web services, BPs, e-contracts and the product line approach for business process (PL4BPM); Section 3 presents our negotiation process and its role in the PL4BPM. This process takes into account the standard ISO/IEC 12207:2008. Section 4 presents related work and Section 5 concludes the paper.

## 2. Background

Web services technology has been identified as the most promising for the implementation of Service-oriented Computing (SOC). According to the web service technology, a software can be decomposed into self-contained, loosely coupled and language independent units [4]. BP can be used to compose web services; integrate systems – including the legacy ones; compose complex applications through services grouping and coordination; and to establish partnerships in DSD. A BP consists of a set of tasks undertaken in a specific sequence to achieve a business goal [7]. It also represents constraints on activities execution order as well as possible interactions between them.

An e-contract is an electronic document used to represent an agreement between partner organizations which is basically composed of: i) product or service definition; ii) rights, obligations and prohibitions; and, iii) actions to be taken in case of disagreements. Contracts can be complex and, in general, its establishment process is often cumbersome due to the large number of parameters involved in the selection of Quality of Service

(QoS) attributes and levels. Thus, a negotiation between the parties is necessary to define the issues involved in establishing an e-contract [3].

PL4BPM aims at offering support to model variability in BPs and web services, and to monitor e-contracts throughout the process execution [6]. It is designed to model the artifacts involved in the negotiation between organizations willing to establish a common e-contract to regulate their cooperation. It is also used for renegotiations when a previously agreed e-contract cannot be executed as planned and the parties desire to keep the cooperation by readjusting the previous contract. In this paper, the e-contract negotiation and establishment process is oriented by feature models (FM) and their possible configurations.

### 3. Contract negotiation and establishment in DDS

Negotiation involves interaction between parties where each party has its aims at reaching an acceptable agreement [8]. Our negotiation process model is composed of 11 (eleven) activities grouped into two cycles. An e-contract model is used to guide the negotiation process. The activities were defined according to: the framework proposed by Kim et al. [10]; the process model of Wu et al. [11] and the standard ISO/IEC 12207:2008 [12]. Figure 1 shows the sequence of the activities of the proposed negotiation process.

The first cycle Planning and Agenda Settings defines the elements that compose the negotiation base. The second cycle – Negotiating and Establishing WS-Contract – uses the structure defined in the first cycle to support the negotiation between parties.

#### 3.2 Planning and agenda settings

The planning and agenda settings cycle produces the elements that compose the negotiation base. Its conceptual model is presented in Figure 2. The *Negotiation case* is the central element. It can be supported by several strategies in different moments of a negotiation. The goal of any negotiation is to reach an agreement, specified as an e-contract. In each *Negotiation case*, a *Negotiator* can play a different *Role*, which has *Cardinality*. It defines the number of possible winners on this *Role*. A *Feature Model Template* which contains the *e-services* that can be contracted is elaborated for every *Role*. It is a reference which contains the requirements elicited from the service consumer. The *Negotiator* partner is responsible for updating the FM according to the services it can offer, thus generating a *Feature Model Instance*. Throughout the negotiation execution, *Negotiation threads* are generated which group the *e-services* and their *Negotiation variables*.

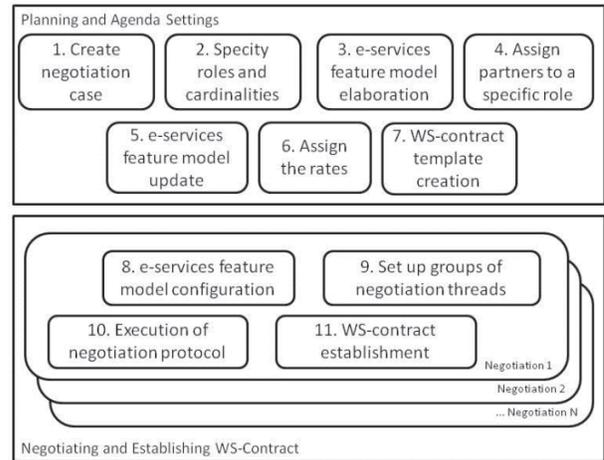


Figure 1. Negotiation process model life cycles.

**3.2.1. Create negotiation case.** In a negotiation, partners are guided by a common objective, such as providing a service. In order to achieve this objective, several elements must be present. In our approach these elements are grouped in a negotiation case. In addition, the negotiation case contains: i) a brief description that can be used in a directory where partners can find negotiations of their interest; ii) detailed information about this negotiation case; and, iii) the maximum deadline to end the negotiation and proceed to the e-contract establishment.

**3.2.2. Specify roles and cardinalities.** The organization partners act within specific roles. A role groups a set of partners capable of providing the same type of service. In the same case there might be several roles.

**3.2.3. E-services feature model elaboration.** Each service in the FM can have one or more negotiation variables. A negotiation variable is an item that acts as negotiation mark in respect to the service. A usual example of negotiation variable is price. PL4BPM considers two negotiation variables: i) e-services properties; and, ii) QoS attributes. In our scenario, e-services properties represent additional characteristics and/or specific requirements, such as an option in the control menu. QoS attributes represents non-functional requirements such as access control.

**3.2.4. Assign partners to a specific role.** The partners are the organizations responsible for providing the services. Each role may group several organizations. However, the final contract must comply with the cardinality already defined. This information is important for decision makers and may affect organization efforts within a negotiation.



support the establishment of partnerships in the e-services context. Shiping and Chen [5] present a web service for e-contract negotiation and establishment whose purpose is to facilitate dynamic collaboration among partners. It works as an intermediary in which partners can be discovered and then form a partnership. Moreover, a framework based on WS-CCDL (Web Service – Collaborative Context Definition Language) was developed to support the approach. Lin [14] presents architectural models for the construction of a negotiation process and contract establishment. The architecture is based on SOA. In another approach [15], negotiation protocols are encapsulated into components. It has a platform in which different negotiation protocols can be used by alternating components. It provides user-friendly interface for the negotiators.

Chiu et al. [9] present a contract negotiation process for a particular scenario, through the use of metamodels that specify the e-contract elements. The metamodels are used in the development of a web services-based framework. To demonstrate the framework applicability, a portal has been developed in which negotiations are conducted and sales contracts are established in an e-commerce environment.

## 6. Conclusion

A successful DSD process involves identifying, negotiating and establishing e-contracts for service provision. The process proposed here defines a sequence of activities that allow conducting a negotiation for the establishment of e-contracts that can be applied in building partnerships in the DSD domain.

FMs representing negotiation elements and variables allow involved experts to have a clear and direct view of the involved elements, regardless their expertise. In general, negotiation processes in a same area involve reuse of artifacts, as supported in the PL4BPM approach. It was therefore important to use the ISO/IEC 12207:2008 standard as a basis for the definition of the process presented both to ensure quality to the process and to provide internationally standardized parameters. Finally, the aim of this particular work is to add computational support to e-negotiation between organizations towards contracting e-services. Moreover, we draw attention to the current market of global software development. Our tool will provide Web interface to support negotiations between organizations located in different geographical areas. The set of techniques applied in this negotiation process allows organizations to save time and money in addition to gaining in the quality.

Future work in this direction includes: i) the support for other negotiation variables, like economic variables; ii) improvement of the decision support during negotiation; and, iii) the support of the cultural differences between partners.

## 7. References

- [1] B. Ghodeswar, and J. Vaidyanathan, "Business process outsourcing: An approach to gain access to world-class capabilities," *Business Process Management Journal*, vol. 14, 2008, pp. 23-38.
- [2] M. Fantinato, M.B.F. Toledo, I.M.S. Gimenes, "E-contract establishment with QoS: An approach based on feature modeling," *Int J Coop Inf Syst*, vol. 17, 2008, pp. 373-407.
- [3] P. Grefen, H. Ludwig, A. Dan, and S. Angelov, "An analysis of web services support for dynamic business process outsourcing," *Inform Software Tech*, vol. 48, 2006, pp. 1115-1134.
- [4] M.P. Papazoglou, *Web Services: Principles and Technology*, Pearson Education, 2008.
- [5] S.C.N. Shiping, and J.W.Z. Chen, "Facilitating Dynamic Collaborations with eContract Services," *Proceedings of the IEEE Int. Conf. on Web Services*, Beijing: IEEE, 2008, pp. 521-528.
- [6] M. Fantinato, I.M.S. Gimenes, B.F. Toledo, "Product Line in the Business Process Management Domain," *K.C. Kang, V. Sugumaran, S. Park*, "Applied Software Product Line Engineering," 2009, Auerbach Publications, London, pp. 497-530.
- [7] M. Weske, *Business Process Management*, Springer, Berlin 2007.
- [8] M. Parkin, D. Kuo, and J. Brooke, "A Framework & Negotiation Protocol for Service Contracts," *Proceedings of the 6th Int. Conf. on Services Computing*, Washington: IEEE, 2006, pp. 253-256.
- [9] D.K. Chiu, S.C. Cheung, P.C. Hung, S.Y. Chiu, and A.K. Chung, "Developing e-Negotiation support with a metamodeling approach in a web services environment," *Decision Support Systems*, vol. 40, 2005, pp. 51-69.
- [10] J.B. Kim, A. Segev, A. Patankar, and M.G. Cho, "Web Services and BPEL4WS for Dynamic eBusiness Negotiation Processes," *Int. Conf. on Web Services*, 2003, pp. 111-117.
- [11] S. Wu, G.E. Kersten, M. Benyoncef, "INSS - A new approach in designing Web-based negotiation support systems," *The Montreal Conf. on e-Technologies*, 2006, pp. 1-15.
- [12] The Int. Organization for Standardization and the Int. Electrotechnical Commission. ISO/IEC 12207:2008, Systems and software engineering, Software life cycle processes. ISO: Geneve, 2008.
- [13] J.C. Okika, and A.P. Ravn, "Classification of SOA Contract Specification Languages," *IEEE Int. Conf. on Web Services (ICWS '08)*, IEEE, Beijing: 2008, pp.433-440.
- [14] J. Lin, "A Conceptual Model for Negotiating in service-oriented environments," *Information Processing Letters*, vol. 108, 2008, pp. 192-203.
- [15] J.B. Kim, G.E. Kersten, K.P. Law, and S. Strecker, "E-negotiation system development: Using negotiation protocols to manage software components," *Group Decision and Negotiation*, vol. 16, 2007, pp. 321-334.

# Transforming Service-Oriented Business Models into Web Service Specifications

Hugo Estrada  
CENIDET  
Cuernavaca,  
Mexico  
hestrada@cenidet.  
edu.mx

Itzel Morales-  
Ramirez  
Universidad  
Veracruzana,  
Mexico  
itmorales@uv.mx

Alicia Martinez  
CENIDET,  
Cuernavaca,  
Mexico  
amartinez@cenidet.  
edu.mx

Oscar Pastor  
Technical  
University of  
Valencia, Spain  
opastor@dsic.upv.es

## Abstract

*At present, enterprises need sophisticated software applications to sell and promote their products or services in order to maintain their leadership in the business world. One of the most promising trends is the use of Web services technology as the appropriate mechanism to implement e-business, which uses Internet to replicate services offered by an enterprise. However, despite the clear advantages of Web services, there are problems in determining the initial functionalities required by them. Currently, the functionalities of Web services are not obtained in a systematic manner from the organizational environment that Web services try to replicate. Therefore, it is complicated to ensure that Web services fit the business user's needs. In this paper we face this problem by defining a methodological approach to generate Web services from service-oriented organizational descriptions. The Model-Driven Architecture has been used in this work in order to ensure the systematic translation of modeling primitives of the organizational model into their corresponding WSDL services descriptions.*

**Key Words:** Business services, Web services, Model-Driven Architecture.

## 1. Introduction

The Internet has become an essential tool in current enterprises' activities [3], where it is used as a successful strategy to guarantee their competitiveness in order to maintain their leadership. In this context, e-business has been implemented as a mechanism that allows the enterprises to offer their products and services remotely, so they can expand their scope that was previously limited by their location. E-business approach focuses on determining the processes needed by a company to offer their services through the Web.

In this context, the Web services are a key technology for the effective operation of e-business systems.

However, not all aspects involved in developing Web services have been properly solved. One of the main issues in defining Web services is the difficulty in determining what should be the expected functionality of such services. This problem arises from the following factors: 1) the current technology in Web service modeling focuses on defining its functionality without considering, in a systematic manner, the main needs of the organizational context; 2) the lack of reliable sources that allow the designers to implement a Web service reflecting the business tasks as well as the user's requirements; 3) the need of some mechanisms to establish the correspondence between the business functionalities and those which have been implemented in Web services.

We consider the most pressing of the current difficulties to specify the correct functionality of Web services to be the following: a) business models are not properly adapted to support service-oriented specifications and the result is the incompatibility between these models and those that implement the Web services; b) the lack of methodological approaches to generate services automatically from the business' features thus often forcing this process to be accomplished manually.

In this research work, a methodological approach is proposed as the first method to generate WSDL (Web Services Description Language) [5] descriptions of Web services, which are obtained from service-oriented business models. It is important to point out that service-oriented approach has been developed over the *i\** Framework [20] by adding new rules and properties to get an organizational vision of services. Therefore, this work establishes an approach of a generation of services where the business model is correctly adapted to the service technology, and the Web services are obtained in a systematic way using the MDA (Model Driven Architecture) approach [14].

This paper is structured as follows: Section 2 presents the related works. Section 3 presents the overview of the proposal. Section 4 presents our transformational approach, and finally, Section 5 presents the conclusions of this work.

## 2. Related works

One of the current interests in Web service computing is the definition of a design methodology to define the appropriate functionalities of Web services. One of the trends in this context is the use of business descriptions as source for the determination of Web services functionalities. Following, some of the more relevant works in this field are detailed:

Papazoglou and Yang [16] present a methodology for Web services design based on business processes. In this approach, strategies and principles are defined to describe business processes in order to facilitate the identification of functions of the Web services. Once the design methodology has been applied, the WSDL specification is created by the following steps: a) specifying the service interface; b) creating the operation parameters; c) establishing the message protocol; and finally d) implementing the services.

The research work defined by Lau and Mylopoulos [11] presents a methodology for Web services generation from Tropos' [18] specifications. As the first phase of this methodology, the capabilities of the agents in the model are identified and UML activity diagrams are used to represent the actor tasks. They also use UML sequence diagrams to describe agent communication. As result of the process, the authors propose the use of open standards to implement the Web Services: a) WSDL for service specification; b) SOAP for request and response procedures and; c) UDDI for service publication.

A guideline for Web service design has been pointed out by Ruiz et al. in [17], that extends the OOWS [8] method to Web services design. The main contributions of this work are: a) to determine the appropriate models to obtain service functionalities; b) to propose a methodological guide to design service functionalities; c) to identify the operations that support the functional requirements of an application and; d) to propose a design method to automatically generate the WSDL specification of the Web services.

In the work of Gronmo et al. [10], a MDD process is defined to develop Web services as an extension of UML. The main contribution of this work is to support bidirectional rules between UML models and WSDL descriptions.

The work of Bézivin et al. [2] presents an MDA approach to generate Web services descriptions from

UML source models. The main contribution of this work is the use of metamodels to define the source and target models in order to apply rules to generate the Web services.

An approach for the automatic generation of WSDL descriptions has been presented in Vara et al. [19] that uses UML as source model. In this work, the authors propose an extension of UML for WSDL. Similar to in the research of Bézivin [2], in this work the MDA approach is applied to obtain a service document from the UML specification.

The work of Yu Xiaofeng et al. [21] presents a transformation from EDOC models to Web service interface models. From the EDOC model is taken a Component Collaboration Architecture (CCA) metamodel as the main profile to mapping elements to WSDL elements, using names for the CCA elements, similar to the names of the WSDL elements to establish the map in a low level.

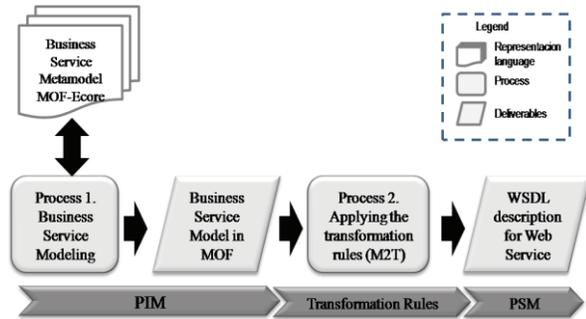
The main difference of our approach with that of those previously mentioned is that our proposal implements the use of business services models as a source for the automatic generation of Web services in an MDA approach. The research works presented in this section use traditional models (Tropos, UML, etc) which are not service-oriented models. In our approach, the use of a business service model is relevant because it will ensure the appropriate correspondence between business functionalities and WSDL specifications of Web services.

## 3. Overview of the proposal

In this section, we present a general overview of a method that allows obtaining the WSDL specifications of Web services from service-oriented organizational models [7]. This method can be divided into two main phases (see Fig. 1): in first phase, a business service model is created according to the work proposed in Estrada [7] to be used as the source to model the Web services functionalities. MOF-Ecore specifications [4] were applied in order to create the metamodel of the service-oriented business model; with this metamodel the transformation from a business environment to the implementation of Web services is achieved using an MDA approach. As result of first phase, a business service model described in MOF (Meta-Object Facility) [15] is created. This model represents the PIM of the transformation approach.

The second phase consists of applying the transformation rules in MOFScript [13] as the necessary to convert the business primitives into elements for Web service design. The rules used in this phase were based on M2T (Model to Text)

transformations because WSDL documents are generated from a service-oriented business model. As result of this phase, a WSDL description for Web services is created according the transformation rules. This specification represents the PSM of the transformation process.



**Fig. 1 Overview of the proposed approach**

The approach presented in this paper has been validated by developing a software tool, that we called MOS, which implements the proposed method.

Section 4.1 shows an example of the modeling stage of a Web service (first phase) and section 4.3 deploys the figures that represent the WSDL documents generated from this method applying the transformation rules M2T (second phase).

Section 4.2 describes the analysis to establish the relationship between elements to model, besides how the rules were designed.

#### 4. Our Web Service Generation Approach

The research work presented in this paper represents the first approach that proposes the use of the MDA standards to translate service-oriented business models into WSDL Web service descriptions. The MDA approach was seen as the suitable choice in carrying out the transformation. In consequence, a restriction was imposed to the transformation tool that consisted in achieving the transformation using models based on MOF specifications and using well-defined models to ensure a systematic generation when applying the transformation rules from a PIM-source model to a PSM-destination description.

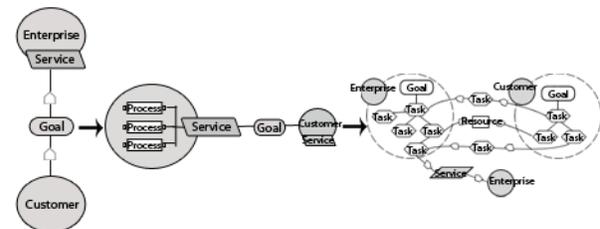
Following, the main phases of the proposed approach are shown in detail.

##### 4.1. Business service modeling phase

The work defined by Estrada in [7] is the basis of this work, where an organizational model is represented as a business service composition model. In this model, the services correspond with the

functionalities that are offered by the enterprise to customers. Estrada proposes an architecture of three models to define the business service view of an enterprise (Fig. 2):

- **Global Model.** The modeling process starts with the definition of a high-level view of the services offered and used by the enterprise. The global model permits the representation of the business services and the actor that plays the role of requester and provider. Extensions to  $i^*$  [20] conceptual primitives are used in this model
- **Process Model.** The business service must be decomposed into the set of concrete processes that perform it. The process model allows for the representation of the functional abstractions of the business process for a specific service. This model provides the mechanisms required to describe the flow of multiple processes. Extensions to  $i^*$  conceptual primitives are used in this model.
- **Protocol Model.** The semantic of the protocols and transactions of each business process are represented in an isolated diagram using the redefinition  $i^*$  conceptual constructs. This model provides a description of a set of structured and associated activities that produce a specific result or product for a business service.



**Fig. 2 Models of the service-oriented architecture**

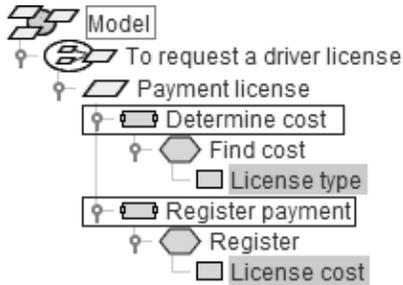
One of the main contributions of this research was the creation of an e-business metamodel which was obtained by applying the MOF-Ecore specifications [4] on the primitives of the service-oriented business model [7]. As a result of first phase in the proposed approach, a business service model is created by using MOF. This model represents the PIM of the transformation approach.

The metamodel called MOS Ecore is based on MOF and ensures the compatibility required for it to be stored in MOF repositories or handled through MOF tools.

Following, an example for the case study “to request a driver license” is shown in Fig. 3. In the model, the processes “determine cost” and “register

payment” were selected to be translated into Web services.

In this modeling stage, each one of the elements and their corresponding properties are described. The complete specification of the model produces the MOS model, a model to design services in the organizational context, ensuring that the subsequent generation of Web services is reliable.



**Fig. 3 Model Web service specification**

The modeling approach proposed in this work has been validated by defining a prototype (MOS tool) that allows for the creation of models with extension \*.mos. The \*.mos file represents the PIM source model that will be used to obtain the Web services specifications.

#### 4.2. Generating a Web service model from mapping rules (M2T) using an MDA approach

One of the key features of the MDA approach is the notion of mapping, which consists of a set of rules and techniques used to modify one model in order to get another model [14].

MDA [1] provides a separation between modeling and implementation details, making a distinction between platform independent models (PIM) and platform specific models (PSM). One of the key technologies used by MDA to define the source models (PIM) and objective (PSM) is MOF, which is a meta-language for creating domain-specific metamodels.

To generate the Web service specification it was necessary to define the elements of the business services model to be used to create the WSDL specification in order to implement the business services. It was not simple to establish the same level of correspondence between Web service and the service element of the business service model, due to the high level of abstraction in the business model. To match the Web service with the business elements, we analyzed the similar functionalities of them, and we found that a Web service corresponds to the process

element; it represents correctly the Web service while the service element represents the orchestration.

Table 1 shows the correspondence between the MOS elements which are represented in WSDL and BPEL elements. These elements are taken from the models proposed in work [7], the models correspond to different stages of refinement in the modeling process, and therefore the elements of each model are broken down into other elements up to the base unit which are resources and tasks.

**Table 1. MOS elements**

Service-Oriented Business elements	Modeling primitive	WSDL elements	BPEL elements
Aggregated service		-	-
Basic service		-	<process>... </process> <partnerLink/> <variable/> inputVariable outputVariable
Process		<service>... </service> <definitions> ...<definitions> </definitions> <port>... </port> <portType> ... </portType> <binding>... </binding>	<partnerLink/> portType
Task		<message>...</message> <operation> ...</operation>	<variable/> port Invoke_..._InputVariable Invoke_..._OutputVariable messageType
Resource		<part/>	Element/ns2:

In order to establish the transformation rules, it was necessary to define the mapping language. The language used to achieve the transformation was MOFScript, which is currently a candidate in the OMG RFP process [13] on MOF Model to Text Transformations. In addition to being a transformation language, it is a tool that is built as a plug-in in Eclipse [9].

One contribution of this work is the definition of the transformation rules in MOFScript in order to obtain the mapping from a MOS model to the WSDL description of Web services that implement business

functionality. The rules are applied to models serialized in XML and they take the \*.mos files as an input to generate WSDL specifications as the outputs of the method. The transformation rules are created following the syntax shown in [13]. Below are just two of the transformation rules defined in this research to produce WSDL documents from a model \*.mos. The complete specification of the rules may be consulted in [12].

**Rule 1.** The tasks involved in a process are mapped into operations of a Web service, through the rule `mmos.Process::portTypesOperations`.

```
mmos.Process::portTypesOperations(): String {
    var operation:String =''
    self.tasks->forEach(t:mmos.Task){
        if(t.transactional == "T"){
            operation= operation + '<!--' +
t.description + '--> \n'
            operation= operation + '<operation
name="' + t.name.replace(" ", "").toLowerCase() +
'"> \n'+
                '<input name="input1"
message="tns:' + t.name.replace(" ",
"" ).toLowerCase() + 'Request"/> \n' +
                '<output name="output1"
message="tns:' + t.name.replace(" ",
"" ).toLowerCase() + 'Response" /> \n' +
                '</operation> \n' }
        }result = operation }
```

This rule creates operations from the tasks included in the processes. Each task has the name property, by which it indicates the name of the operation through instruction `t.nombre.replace(" ", "").toLowerCase()`, as well as being used to specify the messages that are sent or received.

**Rule 2.** The process is mapping to the service described in the WSDL description, through the rule `mmos.Process::serviceWSDL`.

```
mmos.Process::serviceWSDL(): String {
    result = '<service name="' +
self.name.replace(" ", "").toLowerCase() +
'"> \n' +
        '<port name="' +
self.name.replace(" ", "").toLowerCase() +
'HttpSoapEndpoint" binding="tns:' +
self.name.replace(" ", "").toLowerCase() +
'SoapBinding"> \n' +
        '<soap:address
location="http://localhost:${HttpDefaultPort}/' +
self.name.replace(" ", "").toLowerCase() + '/' +
self.name.replace(" ", "").toLowerCase() +
'HttpSoapEndpoint/" /> \n' +
        '</port> \n' +
        '</service> \n' }
```

This rule creates the service, the description of the location address and the port, where the Web service implementation is established in some programming language. This example has written an address that

must be replaced by the server where services will be implemented.

### 4.3. Creation of the Web service specification

The next stage of the method is the application of transformation rules to the source model (business services model) to generate the objective (WSDL description of a Web service).

The MOS tool was developed using NetBeans IDE 6.5 as a mechanism to validate the proposed approach. It has the following features: 1) it allows for creating and opening service models; 2) it serializes the MOS model to XML; 3) it stores the models as files with extension \*.mos and; 4) it executes Eclipse V3.3.2 to apply the transformation rules using the MOFScript plug-in.

Once the WSDL document is created, it just needs to be checked in well-formed XML syntax and the only property that needs to be changed is the address where the implementation is located.

WSDL documents generated were tested in the NetBeans IDE 6.5 platform [6], generating Web services from them. There is a straight relationship between the processes modeled in Fig. 3 and the Web services interfaces obtained from the transformation. Documents generated from the model in Fig. 3, are displayed in Fig. 4 and Fig.5 for services “register payment” and “determine cost”. The figures show the main elements of a WSDL document, for example, the messages that are received or sent, the type of message and portTypes indicating input and output. Besides, each element can be traceable through the correspondences established in Table 1.



**Fig. 4**  
**registerpayment.**  
**wsdl**



**Fig. 5**  
**determinecost.**  
**wsdl**

With the objective of validating this approach five case studies were carried out to analyze if the tool generates the required Web services. Results indicate the correct generation of Web services. For more information regarding these studies, consult [12]. However, we consider that is necessary to perform an empirical evaluation to prove a real utility about the proposed approach against to create manually the Web services or by applying any of the actual approaches in automatic Web service generation.

## 5. Conclusions and future works

In this paper, a methodological approach is proposed to generate Web services from a business service model. We propose that by using a service-oriented model at an organizational level, it is possible to facilitate the work of defining Web services under a methodological approach such as MDA. We have done an initial analysis to generate the BPEL document to obtain a composition of the Web service specifications generated. The result of this work offers a promising approach for BPEL generation.

In future work, we will be dealing with the generation of complete functionality of the Web service. We are currently working on methods to use the organizational descriptions (business service models) to generate precisely the choreography and orchestration of services, using the BPEL language. Another proposal for future work is the extension of the MOS Ecore metamodel to integrate all the modeling stages proposed by the business service architecture [7].

## 6. References

- [1] J. Bézivin, A. Vallecillo, J. García, G. Rossi, "Presentación. Siete años de MDA®: pasado, presente y futuro" (In Spanish). *Novática* No. 192, 2008, pp. 4-8.
- [2] J. Bézivin, S. Hammoudi, D. Lopes, F. Jouault, "Applying MDA Approach for Web Service Platform". *Enterprise Distributed Object Computing Conference, Eighth IEEE International (EDOC'04)*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 58-70.
- [3] N. Bieberstein, S. Bose, L. Walker, A. Lynch, "Impact of service-oriented architecture on enterprise systems, organizational structures, and individuals". *IBM System Journal*, 2005, pp. 691-708.
- [4] F. Budinsky, D. Steinberg, E. Merks, R. Ellersick, T. Grose, *Eclipse Modeling Framework*. Addison-Wesley, 2003.
- [5] E. Christensen, F. Curbera, G. Meredith, W. Sanjiva, *Web Services Description Language (WSDL) 1.1*. <http://www.w3.org/TR/wsdl>.
- [6] NetBeans Community. NetBeans IDE. <http://www.netbeans.org/features/web/web-services.html>.
- [7] H. Estrada, *A service-oriented approach for the i\* framework*. PhD Thesis, Valencia University of Technology, Valencia, Spain, 2008.
- [8] J. Fons, V. Pelechano, M. Albert, O. Pastor, "Development of Web Applications from Web Enhanced Conceptual Schemas". In: *Proceedings of the International Conference on Conceptual Modelling*. Chicago, EE.UU, 13 - 16 October: Springer-Verlag, Lecture Notes, 2003, pp. 232-245.
- [9] Eclipse Foundation. About MOFScript. <http://www.eclipse.org/gmt/mofscript/about.php>.
- [10] R. Grønmo, D. Skogan, I. Solheim, J. Oldevik, "Model-driven Web Service Development". *Idea Group Inc. International Journal of Web Services Research*, Vol.1, No. 4, 2004, pp. 1-13.
- [11] D. Lau, J. Mylopoulos, "Designing Web Services with Tropos". In: *Proceedings of the IEEE International Conference on Web Services (ICWS'04)*. San Diego, California, USA, 2004.
- [12] I. Morales, *Generation of WSDL specifications of Web services from service-oriented organizational models (in Spanish)*. Master Thesis, National Center of Research and Technological Development, Morelos, Mexico, 2009.
- [13] J. Oldevik. Eclipse. MOFScript Documentation. <http://www.eclipse.org/gmt/mofscript/doc/MOFScript-User-Guide.pdf>.
- [14] OMG Object Management Group. MDA Guide Version 1.0.1. Specification, Needham, MA, OMG (2003).
- [15] OMG Object Management Group. Meta-Object Facility (MOF) Specification Version 1.4. Needham, MA, OMG (2002).
- [16] M. Papazoglou, J. Yang, "Design Methodology for Web Services and Business Processes". In: A. Buchmann et al. (eds.) *TES 2002*. LNCS, vol. 2444, Springer, Heidelberg, 2002, pp.54-64.
- [17] M. Ruiz, P. Valderas, V. Torres, V. Pelechano, "A Model Driven Approach to Design Web Services in a Web Engineering Method". In: *Proceedings of the CAiSE'05 Forum*, Portugal, 2005, pp. 183-188.
- [18] A. Susi, A. Perini, J. Mylopoulos, P. Giorgini, "The Tropos Metamodel and its Use". *Informatica journal* 29, No. 4, 2005, pp. 401-408.
- [19] J. Vara, V. De Castro, M. Esperanza, "WSDL Automatic Generation from UML Models in a MDA Framework". *International Journal of Web Services Practices*, Vol. 1, No. 1-2, 2005, pp. 1-12.
- [20] E. Yu, "Modelling Strategic Relationships for Process Reengineering", Ph.D. Thesis, Department of Computer Science, University of Toronto, 1995.
- [21] Yu Xiaofeng, Hu Jun, Zhang Yan, Tian Zhang Tian, Wang Linzhang, Zhao Jianhua, Li Xuandong, "A Model Driven Development Framework for Enterprise Web Services," *edoc*, pp.75-84, 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC'06), 2006

# Reliable Web Service Selection based on Transactional Risk

Ying Yin, Xizhe Zhang, Bin Zhang  
 School of Information Science and Engineering  
 Northeastern University  
 Shenyang, China, 110004  
 yinying@ise.neu.edu.cn

**Abstract**—Composite Web services can be completed complex transactions or workflows by combining existing ones. However, they always run in the highly dynamic and change internet environment, for the applications and environment, advanced transactional support is required to ensure the execution quality of composite service. In order to ensure the selective services execute reliability, this paper proposes a reliable selection model which selects web services not only according to their functional requirements and QoS characteristics but also to their behavioral properties (such as transactional risk) and user's intent(SLA). In order to evaluate the risk of different transactional granularity, a way of mining transactional granularity in advance by considering full multi-relation among transactional Web services is proposed in this paper. Further more, a new comprehensive, objective TRQoS-driven service selection model with transactional risk support is presented.

**Keywords**—Web service; transactional risk; TRQoS; selection; reliability

## I. INTRODUCTION

Web service techniques can offer new added-value services by integrating existing services into a new composite service<sup>[1]</sup>. As a consequence of the rapid growth of web services (especially similar functional), selecting the "optimal" individual services to use becomes a challenging work<sup>[2]</sup> especially in the dynamic environment. Most of the previous work have been to devoted to appropriate service selection based on QoS<sup>[3-5]</sup> (such as price, time, reputation and so on) which can help customers to select a distinguished service with higher qualities. These methods to some extent promote an efficient service selection strategy. However, very few consider behavior properties (such as transactional). Service selection methods ignoring transactional behavior, especially for business process, will be incomplete, unreliable and infeasible enough to express a given customer's intentions and norms<sup>[6]</sup>.

The following scenario in Figure 1 shows the whole business process modeled as a WSDL, service and each task can be implemented by invoking a set of services which span over a single or multiple Web service operations. For simplicity, we assume that one task( $T_i$ ) is corresponding with one transactional Web service in the form of WSDL. In one WSDL document, several port types are defined, each of which acts as the static interface of this Web service. A port type is

composed of multiple operations, which are described in the form of input or output of messages. The execution of business process tasks can be turned into WSDL operation invocations.

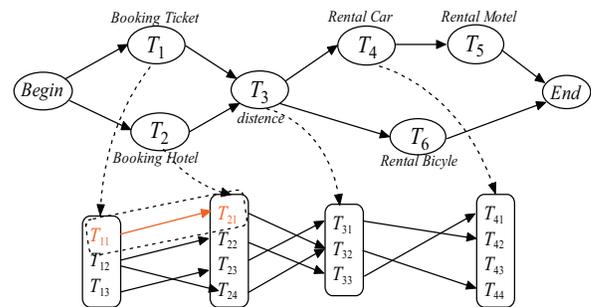


Figure 1. A Business Process Model

For example, if task  $T_{11}$  and task  $T_{21}$  belong to one transactional Web service, then the failure of one Web service ( $T_{21}$ ) will induce another cascading service ( $T_{11}$ ) failure too. If we select  $T_{11}$  and  $T_{21}$  to composite, since the execution cost (including price and duration) of  $T_{11}$  and  $T_{21}$  are less than the cost of  $T_{12}$  and  $T_{22}$ , i.e.  $\text{cost}(T_{11} + T_{21}) < \text{cost}(T_{12} + T_{22})$ . However, the risk with regard to composite transactional Web service of  $T_{11}$  and  $T_{21}$  is higher than the composite Web services of  $T_{12}$  and  $T_{22}$ , i.e.  $\text{risk}(T_{11} + T_{21}) > \text{risk}(T_{12} + T_{22})$ . Due to the transactional Web service ACID properties, if the composite transactional Web service has completed successfully, their duration and price are both lower. Otherwise, The failure of one Web service ( $T_{11}$ ) will induce another cascading service ( $T_{21}$ ) failure, furthermore, the process will induce the entire replacement of the cascading failure services. Therefore, Web service with transactional support is the basic guarantee of reliable and correct execution. In order to support reliable Web service reselection, compensation supporting has become an important solution, since it is the key problem for business process to recovery the system from exceptional failures and execution reliability. Several Web service standards relevant to transaction support have been proposed, such as BPEL4WS, WS-coordination, WS-Transaction, WSCI and WS-CDL and so on<sup>[7]</sup>. These standards and policy have been applied for supporting Web service transaction well. This paper discusses the considered transactional behavior

properties and addresses an efficient service selection model. We summarize our contributions to the above problem as follows:

(1). We have defined a metric, called composite reliability (CR), to measure the probability that a given state in the composite transactional service will lead to successful execution in an error-prone environment;

(2). In order to obtain the behavior of transactional composite Web service, the granularity of transactional Web services can be identify by mining various dependencies exist in tasks;

(3). A new comprehensive, objective transactional-driven services selection model TRQoS which considering not only general QoS but transactional risk and the user's intent (SLA) was sufficiently proposed;

The rest of the paper is organized as follows. In Section 2, we give the basic definition. Section 3 we describe Reliable TR-QoS Driven Web Service Selection model based on Transactional Risk. Finally, conclusion is given in section 4.

## II. BASIC DEFINITION

In this section, we introduce the concept of an atomic transactional Web service (ATS) and composite transactional Web service (CTS). Further, we show how to construct business process logic model by combining a set of transactional Web service.

### Definition 1. Atomic Transactional Web Service.

A component service is atomic if the elements of it can be treated as a unit of work. That is it can use compensation mechanism to ensure that all of its component services complete successfully or none of them do, we say it is atomic transactional service (ATS). ATS is the minimal granularity of composite transactional Web service (CTS).

### Definition 2. Composite Transactional Service.

A composite transactional Service(CTS) is a composite Web service, i.e.  $CTS = \{tws_1, tws_2, \dots, tws_i, \dots, tws_n\}$ , where  $tws_i$  denotes atomic transactional Web service or transactional Web service,  $tws_i \in CTS, i=1,2, \dots, n$ .

The main transactional properties of a Web service we are considering are:

(1) *Retriable*. A service  $s$  is said to be retrieble if it is sure to complete after several finite activations;

(2) *Compensatable*. A service  $s$  is said to be compensatable if it offers compensation policies to semantically undo its effects.

(3) *Pivot*. A service  $s$  is said to be pivot if once it successfully completes, its effects remains forever and cannot be semantically undone.

This shows compensation is the basic properties for transactional Web service. A CTS takes advantage of transactional service behavior properties to specify mechanisms for failure handling and recovery. The goal of this paper is

selecting the reliable composite services based on the above properties of transactional Web service.

**Definition 2. Business Process Model.** A business process model (or a composite Web service) can be modelled in the form of  $BPM = \langle TCS, s, \Sigma CR, \Sigma DR, \Sigma BR \rangle$ , where:

(1)  $TCS$  is a set of transactional Web service( $TWS$ ), i.e.  $TCS = \{tws_1, tws_2, \dots, tws_i, \dots, tws_n\}$ , where  $tws_i \in TCS, i=1,2,\dots,n$ .  $TWS$  was shortly by task. Each task was executed by a set of operations within.

(2)  $\Sigma CR$  is control relation set of tasks,  $\Sigma CR = \{\text{Sequence, And-Join, And-Split, Or-Join, Or-Split}\}$ ;

(3)  $\Sigma DR$  is data relation set of tasks,  $\Sigma DR = \{0,1\}$ , 1 denotes there being data dependency between tasks; 0 denotes there not being data dependency between tasks.

(4)  $\Sigma BR$  is business relation set of tasks,  $\Sigma BR = \{0,1\}$ , 1 denotes there being business dependency between tasks, 0 denotes there being not business dependency between tasks.

(5)  $s: t \rightarrow state$  is a mapping function, where  $state = \{\text{initial, active, failed, completed, aborted, canceled}\}$ .

**Problem Statement:** Given: (1)  $D$ , a service set, and SLA, the task of mining is to find top  $k$ -path, which satisfy TRQoS requirement.

## III. RELIABLE WEB SERVICE SELECTION MODEL BASED ON TRANSACTIONAL RISK

The key of reliable execution of composite Web service is construct a reliable transactional Web service selection QoS model. In this section, we introduce the transactional-driven QoS concept and proposes objective QoS-driven service selective model.

When several functionally and transactional equivalent Web services are available to perform the same activity, traditional Web service selection methods mostly considers price, time, available, successful and so on. However, transactional properties may impact the qualities of web service. In order to reckon the real QoS of Web services, a model is needed which captures the descriptions of these properties from a user perspective. Different previous QoS metric, such model must take into account the fact that transactional QoS including transactional risk and nested reliability and so on except several generic quality criteria for a Web service. In this section, we will give some definitions concerning transactional web service.

### A. Identification the Granularity of Transactional Services

Transactional Web service can be denotes an entity, all the components in which will be done or do nothing. We called the cascading component sets the basic transactional service granularity.

From the structural aspect, we can classify the execution scenarios into five types, namely "Sequential" tasks, "And-Join" tasks, "And-Split" tasks, "Or-Join" tasks, and "Or-Split" tasks<sup>[8]</sup>. According to these scenarios, if task  $tws_i$  and  $tws_j$  satisfy the following conditions, we can obtain the direct service dependency (DD) between tasks as below.

**Definition 5. Direct Compensation Dependency.** If one task  $tws_i$  is fail and need to be compensate, based on the state of task and multi-relation in tasks, we define direct compensation dependency( $DCD$ ) between tasks as follows:

(1) When  $tws_j$  is the direct preceding task of  $tws_i$ , i.e.  $CR(tws_i, tws_j) = \text{"sequence"}$ , if they satisfy the following conditions:  $BR(tws_i, tws_j) = 1$  and  $DR(tws_i, tws_j) = 1$ , we say there being compensation dependency relation, i.e.  $DCD(tws_i, tws_j) = 1$ ;

(2) When two services( $tws_i$  and  $tws_j$ ) both completed before activating another service, i.e.  $CR(tws_i, tws_j) = \text{"And-Joint"}$ , if they satisfy  $BR(tws_i, tws_j) = 1$ , we say there being compensation dependency relation, i.e.  $DCD(tws_i, tws_j) = 1$ ;

(3) When a Web service is activated only when both of its predecessor Web services ( $tws_i$  and  $tws_j$ ) have completed, i.e.  $CR(tws_i, tws_j) = \text{"And-Split"}$ , if they satisfy the following conditions:  $BR(tws_i, tws_j) = 1$  and  $s(tws_i) = \text{"Completed"}$ , we say there being compensation dependency relation, i.e.  $DCD(tws_i, tws_j) = 1$ ;

(4) When only one task will be select from *Or-joint* multitasks or *Or-Split* multi-tasks, i.e.  $CR(tws_i, tws_j) = \text{"Or-joint or Or-Split"}$ , we say there is not compensation dependency relation, i.e.  $DCD(tws_i, tws_j) = 0$ ;

However, in a dynamic composite environment, identifying the granularity of transactional services is the key issue for reliable selection. The granularity of transactional services is the range of cascading compensation that has indirect compensation dependency relationship with failure service node. Indirect compensation dependency relation can exported by direct compensation dependency among services. According to different scenarios from structural aspect, we induce the indirect dependency (ID) as follows:

(1) Sequential as Case 1, if  $DCD(tws_i, tws_j) = 1$  and  $DCD(tws_j, tws_k) = 1$ , when a Sequential task,  $tws_k$ , is aborted or compensated,  $tws_i$  should be compensated, i.e.  $ICD(tws_i, tws_k) = 1$ .

(2) And-Joint as Case 2, if  $DCD(tws_i, tws_j) = 1$  and  $DCD(tws_j, tws_k) = 1$ , when task  $tws_k$ , is aborted or compensated, its And-Joint task,  $tws_i$  should be compensated, i.e.  $ICD(tws_i, tws_k) = 1$ .

(3) And-Split as Case 3, if  $DCD(tws_i, tws_j) = 1$  and  $DCD(tws_j, tws_k) = 1$ , when task  $tws_k$ , is aborted or compensated:

- When  $s(tws_i) = \text{completed}$ , the task,  $tws_i$  should be compensated, i.e.  $ICD(tws_i, tws_k) = 1$ .
- When  $s(tws_i) = \text{initial}$ , no compensation need.

(4) Or-Joint and Or-Split as Case 4, for Or-Join tasks and Or-Split tasks, their proceeding tasks and succeeding tasks will be specific, task  $tws_k$  and one of the proceeding (succeeding) tasks were executed while others are not. Therefore, we can treat them as sequential tasks.

Based on above, we can identify the transactional services granularity, Such as, given a pair of transactional  $T_i$  and  $T_j$ . if they satisfies the following conditions:  $DR(T_i, T_j) = 1$  or  $BR(T_i, T_j) = 1$ , then we say  $T_i$  and  $T_j$  belong to a transactional granularity, shortly by  $CT = \{T_i, T_j\}$ . Further, a hierarchical of nested transactions can be deduce recursively shows in Figure 2, where, lower case letters, such as a, b, c, d, e with black dashed ellipse, denote the atomic transaction and the capital letters, such as A, B with red dashed ellipse, denote the nested transaction. Limit the space, we omit the details.

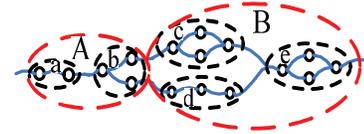


Figure 2. Atomic (nested) Transaction

For example, given a business flow in Figure 1,  $T = \{T_1, T_2, \langle CT_1 \rangle, \langle CT_2 \rangle\} = \{T_1, T_2, \langle T_3, T_4 \rangle, \langle T_5, T_6 \rangle\}$ ,  $T_3$  and  $T_4$  belongs to a compound transactional service( $CT_1$ ) and  $T_5$  and  $T_6$  belongs to a compound transactional service( $CT_2$ ). Furthermore, composite reliability can be induce by considering the transactional granularity.

**Definition 4. Composite Reliability (CR).** Given a business process  $BP$ ,  $BP = \{CT_1, CT_2, \dots, CT_n\}$ ,  $CT_i$  denotes the composite transactional service. Its composite reliability is:

$$CR(BP) = \sum_{i=1}^n CR(CT_n) = \begin{cases} IR(CT_1) + IR(CT_2) + \dots + IR(CT_n), & \text{sequence} \\ IR(CT_1) \times IR(CT_2) \times \dots \times IR(CT_n), & \text{parrell} \end{cases}$$

Where,  $IR$  denotes the inter Reliability (shortly by IR) for an atomic transactional web service,  $T_i = \{tws_1, tws_2, \dots, tws_n\}$ ,

$$IR(T_i) = \begin{cases} R(tws_1) + R(tws_2) + \dots + R(tws_n), & \text{sequence} \\ R(tws_1) \times R(tws_2) \times \dots \times R(tws_n), & \text{parrell} \end{cases}$$

In addition, other quality of Web service, such as time, available, successful and so on are realized by using the aggregation functions similar as Definition 4. We don't discuss the process of TRQoS aggregation in this work. Beside this, we more focus the transactional risk. In the following, we give the risk of composite transactional Web service.

**Definition 4. Business Process Risk**

Given a business process,  $BP = \{CT_1, CT_2, \dots, CT_i, \dots, CT_n\}$ ,  $CT_i$  denotes the composite transactional service, the transactional risk of BP can be denotes by:

$$Risk(BP) = \sum_{i=1}^n Risk(CT_n) = \frac{n(c/cr) + n(r/rc)}{n(c/cr) + n(r/rc) + n(p/pr)}$$

Where,  $n(c/cr)$  is the numbers of services with compensation or compensation&retrial properties,  $n(r)$  is the numbers of retrial service,  $n(p)$  is the numbers of pivot.

### B. TR-QoS Model

In addition, based on granularity of transactional Web service and atomic transactional behavior properties, i.e. pivot, compensational and retrial, web can induce the behavior of different granularity transactional service. The behavior of composite services may be different by compositing different behavior services with the same function which induce the different transaction risk. Limited by space, we only list a pair of atomic transactional behavioral property of sequential execution and parallel execution shows in Table 1. From Table 1, we can export transactional properties of the whole business process.

Table 1. Transactional Behavioral Property

	$tws_1$	$tws_2$	$tws_1; tws_2$	$tws_2; tws_1$	$tws_1 // tws_2$
1	$p$	$p$	$p$	$p$	$\bar{a}$
2	$p$	$c$	$\bar{a}$	$p$	$\bar{a}$
3	$p$	$pr$	$p$	$\bar{a}$	$\bar{a}$
4	$p$	$cr$	$p$	$p$	$p$
5	$c$	$c$	$c$	$c$	$c$
6	$c$	$pr$	$p$	$\bar{a}$	$\bar{a}$
7	$c$	$cr$	$c$	$c$	$c$
8	$pr$	$pr$	$pr$	$pr$	$pr$
9	$pr$	$cr$	$pr$	$pr$	$pr$
10	$cr$	$cr$	$cr$	$cr$	$cr$

Different previous service selection model, based on the above definitions, we proposed a comprehensive, objective and effective service selection model with transactional support. Our TR-QoS driven model not only ensures select the reliable service but also satisfy the users SLA. The objective TR-QoS driven service selection model shows as formula (1):

$$\begin{aligned}
 Score(CWS) &= w_1 \times UF(QoS) + w_2 \times UF(Behavior) \\
 &= w_1 \times (\sum UF_{pQoS} + \sum UF_{nQoS}) + w_2 \times \sum UF_{Risk}
 \end{aligned} \quad (1)$$

where,  $UF(QoS)$  denotes the utility function of composite service QoS,  $UF(Behavior)$  denotes the utility function of composite service risk.  $w_1$  and  $w_2$  denotes the weight of QoS and weight of risk appointed by user separately.  $\sum UF_{pQoS}$  denotes positive quality description of service parameters and  $\sum UF_{nQoS}$  denotes negative quality description of service parameters respectively. In this work, we select the path that has the Web service with the maximal score. If there

are several paths with the maximal score, one of them is selected randomly. Limited by space, we omit the reliable algorithm in details.

### IV. CONCLUSION

This paper proposes a reliable select model based on transactional web service. The model selects web services not only according to their functional requirements and QoS characteristics but also to their behavioral properties (e.g. transactions) and user's intent(SLA). Furthermore, a way of mining transactional granularity in advance by considering full multi-relation among transaction Web services is proposed in this paper. Further more, a new comprehensive, objective TR-QoS service select model with transactional support is presented.

### ACKNOWLEDGMENT

This work was supported by a grant from the National High Technology Research and Development Program of China (863 Program)(No. 2009AA012122), the Fundamental Research Funds for the Central Universities under grants (No. N090304006), National Natural Science Foundation of China under grants (No. 60903009, 60803026).

Corresponding author: zhangbin@mail.neu.edu.cn.

### REFERENCES

- [1] R. M. Kim S. A survey of public web service[C]. In Proc. of Service-Oriented e-Commerce Applications, pp. 96–105, 2004.
- [2] Q. Yu, X.M. Liu, A. Bouguettaya and B. Medjahed. Deploying and managing Web services: issues, solutions, and directions[J], The VLDB Journal, 17:537–572, 2008.
- [3] L. Zeng, B. Benatallah, A.H.H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, QoS-Aware Middleware for Web Services Composition[J], IEEE Trans. Software Eng., vol. 30, no. 5, pp. 311-327, 2004.
- [4] T. Yu, Y. Zhang, and K.J. Lin, Efficient Algorithms for Web Services Selection with End-to-End QoS Constraints[C], ACM Trans. Web, vol. 1, no. 1, pp. 1-25, 2007.
- [5] V. Grassi and S. Patella, Reliability Prediction for Service-Oriented Computing Environments[J], IEEE Internet Computing, vol. 10, no. 3, pp. 43-49, 2006.
- [6] C. L. Li Li and J. Wang. Deriving transactional properties of compositeweb services[C]. In Proc. of the IEEE International Conference on Web Services, USA, pages 631–638, 2007.
- [7] F. C. Luis, C. George and F. Max, "Web services transactions specifications", In Available at <http://www.ibm.com/developerworks/library/specification/ws-tx/>, 2005.
- [8] Y. Yin and B. Zhang. Self-healing Composite Web Service for Reliable Execution[C]. In Proc. of IEEE Asia-Pacific service computing conference, Singapore, pp. 307-312, 2009.

# Selecting Web Services for Choreography Implementation: Compatibility Checking Approach with Access Control<sup>1</sup>

Emad Elabd, Emmanuel Coquery, Mohand-Said Hacid  
Université de Lyon

Université Claude Bernard, Lyon 1

LIRIS CNRS UMR 5205 - UFR d'Informatique

43, boulevard du 11 Novembre 1918, 69622 Villeurbanne cedex - France  
{emad.elabd,emmanuel.coquery, mohand-said.hacid}@liris.cnrs.fr

## Abstract

Nowadays, Web services technologies are adequate for designing and implementing complex inter-enterprise business applications. Web services choreography defines the required behaviors of Web services which participate in implementing such applications with their interactions through message exchanges. The designer of the application collects the Web services that implement the defined choreography. The selected services must be compatible and perform the required operations of the application. Therefore, checking the compatibility between Web services to guarantee that they can interact correctly is a main step in the verification process. This type of checking is based on the services descriptions. Enriching services descriptions by including their behaviours is becoming more and more important. This behaviour can be described by business protocols representing the possible sequences of message exchanges. Since a lot of Web services use access control policies to restrict the access to authorized consumers, these policies should be a part of the service description. Selecting compatible Web services for implementing service choreography is the main contribution of this work. This is achieved by modeling and checking the compatibility between Web services by analyzing their business protocols after assigning the access control policies which will be presented using an ontology.

## 1. Introduction

Web services are loosely coupled applications designed to support interoperable machine-to-machine interaction over a network. This type of interaction between components is formally defined by the service-oriented architecture (SOA) [1]. Therefore, all the information about the service that is needed by the client should be included in service description in order to allow the client to correctly interact with it. A business protocol of a Web service, that is the possible message exchange sequences supported by the service, should be included in the service descriptions (see, e.g., [2]).

Web services choreography is used in the design phase of complex peer-to-peer applications in which each peer can be implemented by a Web service. The behavior of each peer must be specified in the choreography of the application. Any Web service that would like to join the choreography would need to conform to that specification. Several research efforts focus on the issue of determining whether the behavior of the Web services implementing choreography matches the one described by the choreography specification [13, 14, 15]. Behavior conformance must include the

satisfaction of the access control policies (ACP) between services. In this paper, web service behavior description is enriched by annotating the ACP on the business protocols. Moreover, we assume that the invocation of each Web service operation is controlled by access control policies; such policies establish which credentials the invoker Web service must possess in order to be able to invoke the operation.

A business protocol (BP) of a service represents the possible sequences of message exchanges between this service and the other services. This protocol model is presented using state chart which is a suitable model for describing behaviors. Figure 1 shows an example of a business protocol of an authentication service based on a specific identification card. In this model, states represent the various stages that a service may go through while transitions are triggered when a message is received or sent. The positive polarity indicates that the message is an incoming message and the negative polarity indicates that the message is an outgoing message. There is a unique initial state (e.g.,  $s_0$ ) and one or more final states (e.g.,  $s_5$ ).

This work discusses the results of modeling and analyzing Web services business protocols augmented with access control policies to verify the implementation of Web services choreography. Access control policies are expressed using ontology in order to benefit from the flexibility offered by subsumption on concepts together with the possibility to use ontology alignment in the context of the semantic Web. We define and verify Web service compatibility in order to see if (and how)  $n$  services can have interactions based on their protocols.

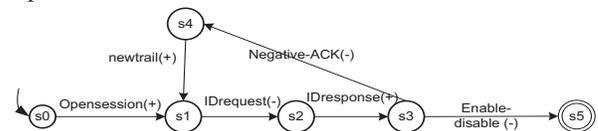


Figure 1. Authentication Web service business protocol.

This paper is organized as follows. Section 2 presents the previous work. Section 3 presents Web service choreography modeling. Section 4 describes the formal methods and algorithms applied on Web service business protocols. Section 5 explains the verification process. Finally, Section 6 presents the conclusion.

## 2. Related work

The need for formal methods and software tools for automatically analyzing service descriptions is widely recognized, and many approaches have been developed to this

<sup>1</sup> This work is partially funded by the European FP7 project COMPAS, GAN 215175, <http://www.compas-ict.eu/>.

end. State-based models are commonly used to model the behaviours of systems, due to the fact that they are simple and intuitive. Various Web services models have been proposed for capturing different types of abstractions. For instance, Fu et al. in [4] present a modeling for Web services interactions by formalising the specification and verification of electronic services for composition purposes. Qui et al. in [5] propose a language for Web services choreographies called **Chor** as a simplification of WS-CDL [6].

Formal analysis of service protocols in terms of automated support to service interoperability at the business protocol level has been discussed in some recent works [7, 8, 9, 10]. A model for business protocols and a framework for protocol-based analysis had been presented by Benatalah et al. [11]. This model captures all the conversations that are supported by a service. Bordeaux et al. in [8] present three different definitions for compatibility: (a) two services A and B are compatible if they have opposite behaviors; (b) two Web services are compatible if they do not have unspecified reception, and (c) two services are compatible if there is at least one execution leading to a pair of final states. There is a drawback with the first and the second definitions, that is they do not check whether the interaction will reach a final state or not.

In our definitions of compatibility, we merge the second and the third definition of Bordeaux [8]. Therefore, n business protocols are compatible if and only if any potential sent message from one service can be received by one of the other services during their interaction and vice versa, and any reachable state is not in a deadlock, i.e., there is at least one execution leading to a pair of final states. Based on our definition, if n protocols are compatible, we guarantee that no error can happen during the interaction.

Foster et al. [17] present a formalization of Web services composition and Web services choreographies based on finite state process algebra. Busi et al. [14] and Kazhamiakin et al. [15] propose two formal calculi to model Web services orchestration and Web services choreography. They investigate the interdependencies between choreography and orchestration and propose a bisimulation-like notion of conformance between choreography and orchestration of Web services.

Development of suitable access control models, able to restrict access to Web services to authorized users is an important issue. Mecella et al. [12] present a conversation-based Web services access control model. Robinson et al. [16] investigates the problem of how to enforce access control in Web services choreographies. They propose a mechanism to derive access control policies to be enforced by each Web service covering a choreography role and architecture to enforce such policies at runtime. Access control policies enforcement is enabled and disabled in a just-in-time manner that matches the control flow described in the choreography.

Paci et al. in [13] present an approach to determine at the design time whether a choreography can be imple-

mented by a set of services based on their access control policies and the disclosed policies regulating the release of their credentials. Our verification approach is different from their work because it based on checking the compatibility between the services which are selected for implementing the choreography by using their business protocols. In addition, we will use the ontology in presenting the ACP and credentials. Description logics (DLs) [3] as policy formalisms technique can be used to present access control policy ontology. Descriptions logics are very useful for defining, integrating, and maintaining ontology, which provide the semantic Web with a common understanding of the basic semantic concepts used to annotate Web pages. Access control policy will be formalized using DL. We will perform subsumption ( $\sqsubseteq$ ), union ( $\cup$ ), and intersection ( $\cap$ ) operations on the ontology during our algorithms. Presenting ACP of Web services as ontology will enables us to use ontology alignment tools to find classes of data that are "semantically equivalent".

### 3. Modeling Web Services Choreography

Web service choreography is modeled as a non deterministic transition system. In this model, business protocol defines Web service behavior that must be followed by the selected web service to participate on the choreography. Each state of the model presents the status of each business protocol. The transitions between the states of the model are annotated by the messages exchange between services with ACP or credentials. Each message exchange is associated with an operation offered by a service and implies an exchange of information between the invoker service and the service providing the operation. The message's sender and receiver are also annotated on the transition. A *conversation* is a sequence of message exchanges starts by the start state and ends by a final state.

**Definition 1: (WS-Choreography Transition System with ACP)** A choreography is represented by a non deterministic transition system  $TS = (S, M, T, s_o, s_f)$ .  $S$  is a set of choreography states. Each state is a tuple of the form  $(Na, (p_1, state_1), (p_2, state_2), \dots, (p_n, state_n))$  where  $Na$  is the state name and in each tuple  $(p_i, state_i)$ ,  $state_i$  represents the state of business protocol  $p_i$ .  $s_o \in S$  is the initial state and  $s_f \in S$  is the final state.  $M$  is a set of message exchanges. Each message exchange is represented by a tuple  $(p_s, p_d, o, AC, c, mt)$ , where  $p_s$  is the business protocol of the message's sender,  $p_d$  is the business protocol of the receiver of the message,  $o$  is the operation of the  $p_d$  that invoked by  $p_s$  that triggers the message exchange,  $AC$  is the access control policy,  $c$  is the set of credentials, and  $mt$  is the message type.  $T \subseteq S \times M \times S$  is the transition relation. A transition  $(s, m, s') \in T$  if  $m = (p_s, p_d, o, AC, c, mt)$  and the tuples  $(p_s, state_s)$  and  $(p_d, state_d)$  in state  $s$  are replaced by the tuples  $(p_s, state'_s)$  and  $(p_d, state'_d)$  in state  $s'$  (respectively) due to the invocation of the operation  $o$ .

**Example 1:** Figure 2 shows an example of Web service choreography describing shopping process between Seller,

Buyer, and Broker. Credit agency partner is used for checking the credentials that are provided by the partners. Each state of the choreography contains information about the current state of all the participants (seller, buyer, credit agency, and broker). For example, the state *ARTICLE SPECIFICATION SUBMIT* indicates that the Buyer is in the start state, the Seller is in the *Sent\_Req* state, the Broker is in the *Rec\_Req* state, and the CreditAgency is in the start state. The process is based on a broker between the seller and the buyer. The sellers send the articles to the broker and the buyers buy these articles from the broker. Usually, this system is used when the sellers are not specialists and they want to sell some articles.

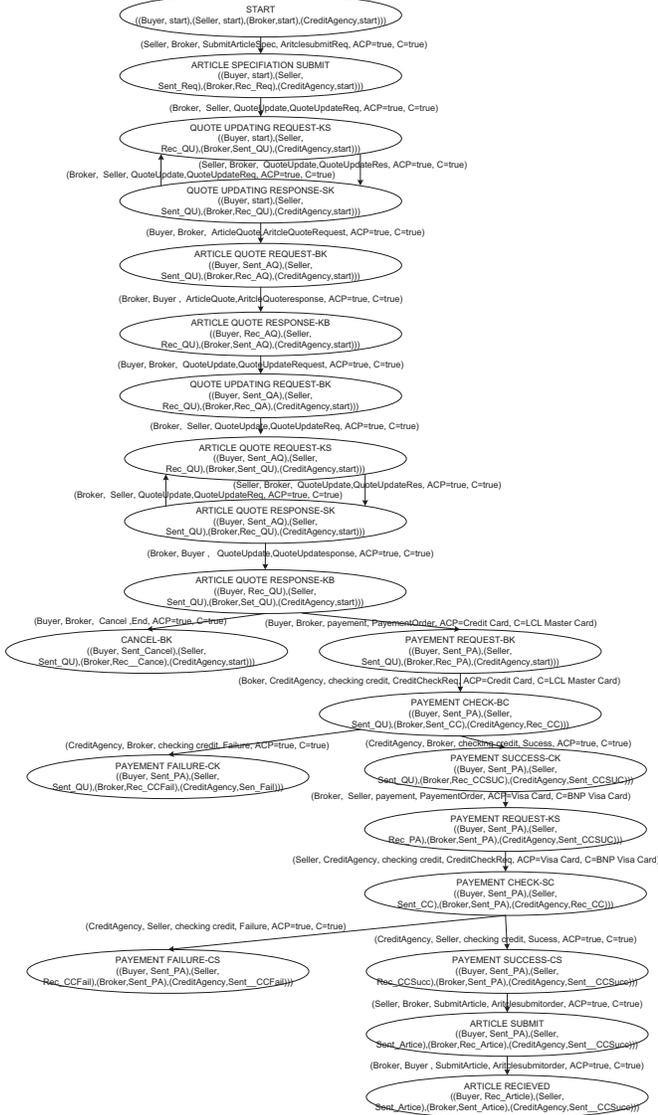


Figure 2: Web services choreography describing shopping process

## 4. Formalization and Algorithms

This section presents the formal definitions of business protocol and the algorithms used in the analysis. The busi-

ness protocol definitions is based on the definition of Benattallah in [2], augmented with ACP and the sender or the receiver of each message. This protocol is deterministic (i.e. all the outputs transition from any state are different).

**Definition 2:** A business protocol assigned with ACP is a tuple  $P = (S; s_0; T; F)$  which consists of the following elements:

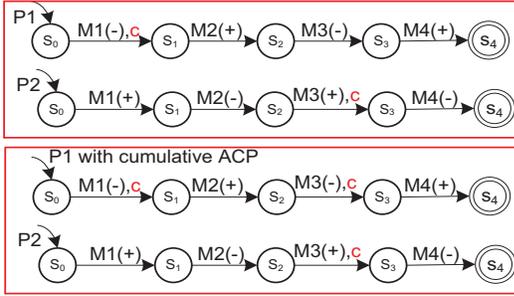
- $S$  is a finite set of states and  $s_0 \in S$  is the initial state.
- $T \subseteq S^2 \times M \times ((\{-\} \times 2^c) \cup (\{+\} \times AC)) \times P'$ , is a finite set of explicit transition, where  $M$  is a set of messages assigned to the explicit transitions between the states,  $AC$  is the set of access control policies,  $c$  is the set of credentials, and  $P'$  is the protocol which receives or sends the message on this transition.
- This protocol is deterministic (i.e. a message cannot correspond to more than one output transition of a state).
- All states in the protocol are accessible and co-accessible (i.e. there a path from the start state to this state and from this state to a final state).
- $F \subseteq S$  is a set of final states. If  $F = \{\emptyset\}$  then  $P$  is said to be an empty protocol.

**Output ( $s_i$ )** defines all the outgoing transitions triggered from the state ( $s_i$ ) and **input ( $s_i$ )** defines all the incoming transitions to the state ( $s_i$ ).

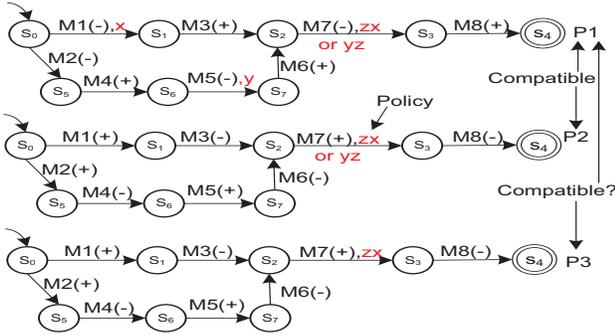
### 4.1. Compatibility

There is a difference in the methodology between checking the compatibility in terms of message exchange and in terms of ACP. Checking the compatibility in terms of message exchange depends on the current message of each protocol and corresponding current message in the other protocol. But checking the compatibility in terms of ACP depends on the current ACP and the previous or current credentials of the corresponding transition. Therefore, there is a need to update each transition with all the credentials that can be provided before reaching it (i.e. transition credentials updated to be all the credential resulted from the current credentials and the previous provided credentials). We called this set of credentials “cumulative Access Control Credentials (ACC)”. Figure 3 shows an example of two business protocols  $P_1$  and  $P_2$  where  $P_1$  provide its  $c$  credential in the first transition but  $P_2$  asked for it in the third transition. If we compare the two protocols without calculating the cumulative ACC then we will find that they are incompatible but after calculating the cumulative ACC for  $P_1$  we find that they are compatible. The question is then; in which step in the checking process the satisfaction between ACP and credential can be checked? Figure 4 shows an example to answer this question. In this figure,  $P_1$  is a client protocol and  $P_2$  is a service protocol and  $P_1$  is compatible with  $P_2$ .  $P_3$  is another service protocol which is not compatible with  $P_1$  because the policy  $xz$  in  $M_7$  in  $P_3$  can not satisfied by ( $zx$  or  $yz$ ) credentials in  $P_1$ . Figure 5 shows an example where protocol  $P_2$  has a policy  $zx$  not satisfied by ( $zx$  or  $yz$ ) credentials in  $P_1$  but the two protocols are compatible because there is a part of the credentials ( $yz$ ) will not provided

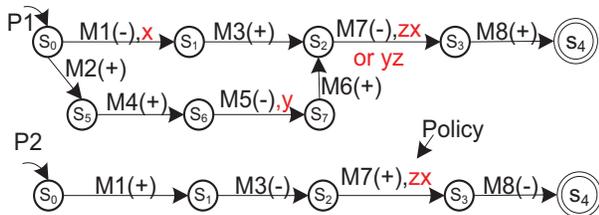
by the protocol P1 if it interacts with P2 and the provided credentials in M7 is zx which satisfy the zx in P2. Therefore, we must not calculate the cumulative before determining the transition which will be used in the interaction between the two protocols. So, when we update the cumulative ACC, we only consider the transition which will potentially share in the interaction.



**Figure 3: Two protocols indicate the importance of calculating the cumulative access control credentials.**



**Figure 4: Three different business protocols P1, P2, and P3. P1 is compatible with P2 but not compatible with P3.**



**Figure 5: P1 is compatible with P2 (this show the importance of calculating the cumulative ACC after determining the transition which will be used in the interaction and this is accomplished by calculating the product automata.)**

**Definition 3.** The product automata  $A^P$  of  $n$  business protocols  $P^1 = (S^1; s_0^1; T^1; F^1), \dots, P^n = (S^n; s_0^n; T^n; F^n)$  is defined as  $A^P = (S^P; s_0^P; T^P; F^P)$  where:

- $S^P = S^1 \times \dots \times S^n, s_0^P = (s_0^1; \dots; s_0^n)$ .
- $T^P$  is the greatest subset of  $(S^P \times S^P \times M \times AC \times 2^c \times P^X \times P^Y)$  such that for each transition  $(s_i^P; s_{i+1}^P; m_i; AC_i^P; c_i^P; p^X; p^Y) \in T^P$  there exist two transitions  $(s_i^X; s_{i+1}^X; m_i; (AC_i^X \text{ or } c_i^X); p^Y) \in T^X$  and  $(s_i^Y; s_{i+1}^Y; m_i; (AC_i^Y \text{ or } c_i^Y); p^X) \in T^Y$  with:

- $s_i^X, s_i^Y$  are in the states which composes  $s_i^P$  and  $s_{i+1}^X, s_{i+1}^Y$  are in the states which composes  $s_{i+1}^P$ .
  - $\text{Polarity}(m_i; T^X) \neq \text{polarity}(m_i; T^Y)$  and
    - If  $\text{polarity}(m_i; P^X) = -$  then  $AC^X = \text{True}, AC_i^P = AC_i^Y, c_i^P = c_i^X, c_i^Y = \text{true}$
    - Otherwise  $(m_i; P^Y) = -$  then  $AC^Y = \text{True}, AC_i^P = AC_i^X, c_i^P = c_i^Y, c_i^X = \text{true}$
- $F^P = F^1 \times \dots \times F^n$ .

**Definition 4.** (Cumulative path in the product automata):

$PA^P = s_i^P \xrightarrow{CU_i^{xy}} s_{i+1}^P; \dots; s_n^P \xrightarrow{CU_n^{xy}} s_{n+1}^P$  is a cumulative path in the product automata  $A^P = (S^P; s_0^P; T^P; F^P)$  where

- $CU_i^{xy}$  is the set of cumulative credentials which is received from the protocol  $P^X$  to the protocol  $P^Y$ .  $CU_i^{xy}$  is the union of the previous set of cumulative credentials  $CU_{i-1}^{xy}$  and the current set of credentials  $c_i^{xy}$  where  $c_i^{xy}$  is the set of credentials on the transition between the state  $(s_i^X \text{ and } s_{i+1}^X)$  of the protocol  $P^X$  where  $s_i^X$  is one of the states that produce  $s_i^P$  and  $s_{i+1}^X$  is one of the states that produce  $s_{i+1}^P$ .
- $CU_0^{xy} = C_0^{xy}$

– A complete cumulative path in the product automata is the cumulative path which starts with the initial state  $s_0^P$  and ends with a final state  $s_f^P \in F^P$ .

**Definition 5.** (Co-accessibility of a state in the product of automata):  $A^P = P^1 \times \dots \times P^n = (S^P; s_0^P; T^P; F^P)$  is the product of automata of  $n$  BP, state  $s_i^P \in S^P$  is co-accessible if there exist two paths  $PA^1$  and  $PA^2$  where  $PA^2 = s_i^P \cdot PA^1 \cdot s_f^P$ , and  $s_f^P \in F^P$ .

**Definition 6.** (Compatibility in terms of product automata assigned with ACP) Protocols  $P^1 = (S^1; s_0^1; T^1; F^1), \dots, P^n = (S^n; s_0^n; T^n; F^n)$  and  $A^P = (S^P; s_0^P; T^P; F^P)$  is their product automata assigned with ACP, we say that the  $n$  protocols are compatible using their product automata if there is a relation  $R = S^1 \times \dots \times S^n$  where for all  $(s_i^1, \dots, s_i^n) \in R$ :

- $\forall (s_i^X; s_{i+1}^X; m; c_i^X; p^Y) \in T^X, \exists (s_i^Y; s_{i+1}^Y; m^+; AC_i^Y; p^X) \in T^Y$  where  $(s_i^P; s_{i+1}^P; m; c_i^X; p^X; p_i^Y) \in T^P$ , and  $s_{i+1}^P \in R$
- $\forall (s_i^Y; s_{i+1}^Y; m^-; c_i^Y; p^X) \in T^Y, \exists (s_i^X; s_{i+1}^X; m^+; AC_i^X; p^Y) \in T^X$  where  $(s_i^P; s_{i+1}^P; m; c_i^Y; p^Y; p_i^X) \in T^P$ , and  $s_{i+1}^P \in R$ .
- $s_i^X, s_i^Y$  are in the states which composes  $s_i^P$  and  $s_{i+1}^X, s_{i+1}^Y$  are in the states which composes  $s_{i+1}^P$
- $s_{i+1}^P \in S^P$  is co-accessible; there is a path in the product automata from this state to final state.
- $s_0^P \in R$

– For all the complete non looping cumulative paths  $PA^P = s_0^P \xrightarrow{CU_0^{xy}} s_1^P; \dots; s_n^P \xrightarrow{CU_n^{xy}} s_{n+1}^P$ , in the product automata, each policy  $AC_i^{yx}$  is satisfied by the set of cumulative credentials  $CU_i^{xy}$ .

The algorithm which is used for checking the compatibility between  $n$  protocols in terms of product automata with ACP can be divided into two parts. The first part is for checking compatibility in terms of message exchange and this can be done by constructing the product automata and

traversing through it, starting by the initial state, using breadth first approach and checking that if there is a state does not included in this relation set R (i.e. each state have two corresponding states of the n protocols and all the outgoing messages from this state in one protocol can be received by one of the other protocols) then the algorithm stops and the n protocols are not compatible, else if all states in the product automata are included in this relation set then the two protocols are compatible in terms of message exchange and go to the second part. The second part is for calculating the cumulative credentials on each transition on the product automata.

Algorithm 1 presents the second part of the algorithm. The idea of this algorithm is to use the queue data structure to cumulate the credentials. Each element of the queue consists of the state, cumulative credentials which are corresponding to the n protocols in this state. The algorithm traverses through the automata for updating these credentials of the states and in the same time updates the cumulative credentials on the transitions. After calculating the cumulative credentials on each transition, if any ACP related to one of the protocols on any transitions is not satisfied by the cumulative credentials on this transition then the n protocols are not compatible in terms of ACP. The AND operator between two credentials in an ACP expression means that these two credentials are required to satisfy this policy and the OR operator between them means that one of these credentials is sufficient for satisfying the policy.

---

**Algorithm 1: Compatibility between n protocols in terms of ACP using cumulative product automata.**

---

**Input:**  $P^1 = (S^1; s_0^1; T^1; F^1), \dots, P^n = (S^n; s_0^n; T^n; F^n)$  and  $A^p = (S^p; s_0^p; T^p; F^p)$  their product automata  $A^p = (S^p; s_0^p; T^p; F^p)$

**Output:** The n protocols are compatible in terms of ACP or not.

**1-Calculate the cumulative credentials on the automata.**

$CU_i^{xy}$ : Cumulative credentials sent by protocol  $P^x$  to the protocol  $P^y$  before reaching to the state  $s_i^p$ .

$CU_{ij}^{xy}$ : Cumulative credentials sent by protocol  $P^x$  to the protocol  $P^y$  and assigned to the transition between  $s_i^p$  and  $s_j^p$  (i.e. union of set of credentials in those transitions).

**For each** state  $s_i^p \in \text{output}(s_0^p)$  **do**

$CU_i^{xy} = CU_{0i}^{xy}$   
 ENQUEUE( $s_i^p; CU_i^{xy}$ )

**while** Q  $\neq$  empty **do**

Temp\_Q = DEQUEUE(Q)  
 for each  $s_j^p \in \text{output}(s_i^p)$  in which  $(s_i^p; CU_i^{xy}) = \text{Temp\_Q}$   
 do  
    $CU_j^{xy\_temp} = CU_j^{xy}$   
   if  $CU_j^{xy} \neq \text{null}$  then  
      $CU_{ij}^{xy} = (CU_{ij}^{xy} \text{ AND } CU_i^{xy}) \text{ OR } CU_j^{xy}$   
   else  
      $CU_j^{xy} = (CU_{ij}^{xy} \text{ AND } CU_i^{xy})$   
    $CU_{ij}^{xy} = CU_{ij}^{xy} \text{ AND } CU_i^{xy}$   
   if - (( $CU_j^{xy} = CU_j^{xy\_temp}$ ) and  $CU_j^{xy} \neq \text{null}$ ) then

| ENQUEUE(Q;  $s_j^p; CU_j^{xy}$ )  
 2- **If**  $\forall AC_i^{xy}, CU_i^{xy} \subseteq AC_i^{xy}$  **in the cumulative product automata satisfying it then**  
 | The n protocols are compatible in terms of ACP.  
 Else  
 | The n protocols are not compatible in terms of ACP.

---

**Complexity analysis:** Let  $T_1, \dots, T_n$  be the number of transitions of the n protocols  $P^1, \dots, P^n$  respectively, the construction of the product automata will take  $(T_1 \times T_2 \times \dots \times T_n)$ . The calculation of the cumulative credentials will take number of states in the product automata  $S^p$  multiplied by the size of the longest non looping path which equals the numbers of states  $S^p$  multiplied by the maximum numbers of paths which also equals  $S^p$  (i.e. cumulative credentials takes  $(S^p)^3$ ). As a result, the complexity for the algorithm will be  $((T_1 \times T_2 \times \dots \times T_n) + (S^p)^3)$ .

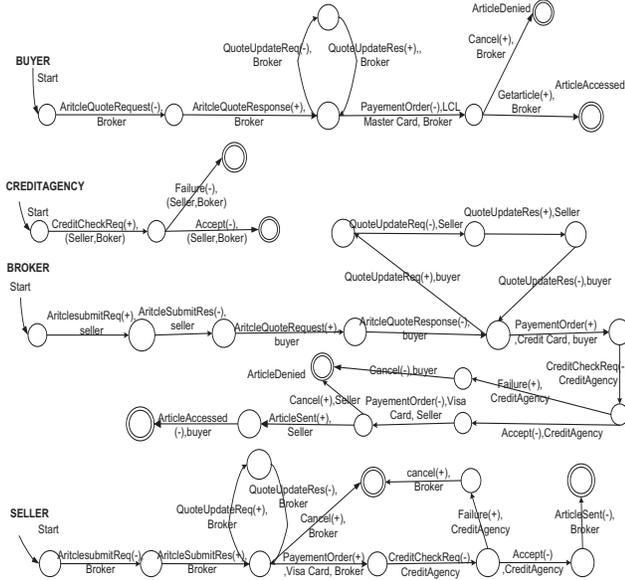
## 5. The verification process

The verification process can be summarized in the following steps:

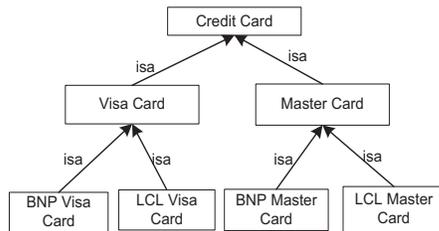
1. Select the Web services and get its business protocols assigned with the ACP and credentials.
2. Create the product automata between these protocols (as defined on definition 3).
3. Calculate the cumulative ACC on the product automata (as defined on definition 4).
4. Check the compatibility in terms of ACP between these protocols (as defined on definition 6) using algorithm 1 for calculating and checking the ACP on the product automata.
5. If the business protocols are compatible in terms of message exchange and ACP and the product automata presents the same behaviour as the choreography then the set of services which have these business protocols can implement this choreography. Otherwise, this choreography cannot be implemented by these services.

**Example:** figure 6 shows set of business protocols of Web services for implementing the choreography of figure 2. The protocols are annotated with the ACP. Figure 7 shows a graphical representation of a simple Credit Card ontology will be used on the verification process. During the compatibility checking algorithm, the provided credentials are checked against the required ACP. The ontology is used during this checking to calculate the subsumption between credentials. For instance in figure 6, the Broker required policy in the transition ‘PaymentOrder(+), Credit Card, buyer’ is “Credit Card” and the corresponding Buyer provided credentials in the transition “PaymentOrder(-), LCL Master Card, Broker” is the “LCL Master Card”. The ontology of figure 7 shows that the “LCL Master Card” is a “Master Card” which is a “Credit Card”. As a consequence, the “LCL Master Card” satisfies the policy “Credit

*Card*. Without using the ontology, the two protocols are not compatible because the checker will not detect the relation between the policy and the provided credentials.



**Figure 6: Set of business protocols of services that can be used for implementing the choreography of figure 2.**



**Figure 7: Graphical representation of a simple Credit Card ontology which is used in the verification process.**

## 6. Conclusion

In this paper we propose a verification approach to verify the behaviours specified by processes choreographies and the selected web services for implementing these choreographies. Therefore, this work investigated a high-level analysis and management of business protocols of Web services that is aware of access control policies assigned to service operations. Beside protocol annotation with policies and credentials, we defined notions of compatibility based on those annotated protocols. Together with those notions, we proposed algorithms for checking compatibility between any set of services.

The contributions of this paper can be extended in several directions. First, we will extend this work to include time constraints. Another direction for future work consists in abstracting annotations in order to deal with other functional and non functional properties, such as quality of service or privacy. A third possible extension of this work is to automatically build adapters allowing set of services to work together even though they are not directly compatible.

A fourth extension is to use these tools for web service composition.

## References

- [1] Mike P. Papazoglou, Willem-Jan van den Heuvel. Service oriented architectures: approaches, technologies and research issues. *VLDB Journal*, 16(3):389–415, 2007.
- [2] Boualem Benatallah, Fabio Casati, Julien Ponge, Farouk Toumani. Compatibility and replaceability analysis for timed web service protocols, *BDA 2005*.
- [3] Franz Baader, Werner Nutt: Basic Description Logics. *Description Logic Handbook 2003*: 43-95.
- [4] Xiang Fu, Tefvik Bultan, Jianwen Su. Conversation protocols: a formalism for specification and verification of reactive electronic services. *Theoretical Computer Science (TCS) 328(1-2)* (2004).
- [5] Zongyan Qiu, Xiangpeng Zhao, Chao Cai, and Hongli Yang. Towards the Theoretical Foundation of Choreography. *WWW'07*.
- [6] Nickolas Kavantzias and al. Web service choreography description language (wscdl)1.0.<http://www.w3.org/TR/ws-cdl-10/>.
- [7] Julien Ponge, Boualem Benatallah, Fabio Casati, Farouk Toumani. Fine-grained compatibility and replaceability analysis of timed Web service protocols. In: *ER. (2007)* 599–614: Auckland, New Zealand
- [8] Bordeaux L., Salaun G., Berardi D., Marcella M. When are two Web Services Compatible? In: *VLDB TES'04*, Toronto, Canada (2004).
- [9] Tefvik Bultan, Xiang Fu, Richard Hull, and Jianwen Su. Conversation specification: a new approach to design and analysis of e-service composition. In *WWW 2003*, Budapest, Hungary, pages 403-410. ACM, May 2003.
- [10] Andreas Wombacher, Peter Fankhauser, Bendick Mahleko, Erich J. Neuhold: Matchmaking for Business Processes Based on Choreographies. *EEE 2004*: 359-368
- [11] Boualem Benatallah, Fabio Casati, Farouk Toumani. Web service conversation modeling: A cornerstone for e-business automation. *IEEE Internet Computing* 8 (2004) 46–54.
- [12] Massimo Mecella, Mourad Ouzzani, Federica Paci, Elisa Bertino. Access control enforcement for conversation-based Web services, *WWW 2006*:257-266.
- [13] Federica Paci, Mourad Ouzzani, Massimo Mecella. Verification of Access Control Requirements in Web Services Choreography. *IEEE SCC 2008*:5-12.
- [14] Busi, N., Gorrieri,R., Guidi,C., Lucchi,R. and Zavattaro, G. Choreography and Orchestration Conformance for System Design. In *Proc. of 8th International Conference on Coordination Models and Languages*, 2006.
- [15] Kazhamiak, R., Pistore, M. Choreography conformance analysis:asynchronous communications and information alignment. In *Proc. of Web Services and Formal Methods, Third International Workshop, WS-FM*, 2006.
- [16] Robinson, P., Kerschbaum, F., Schaad, A.: From Business Process Choreography to Authorization Policies. In *Proceedings of 20th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, Sophia Antipolis, France, July 31-August 2, 2006.
- [17] Foster, H., Uchitel, S., Magee, J., Kramer, J. WSEngineer: A Rigorous Approach to Engineering Web Service Compositions and Choreography. In *Proc. Of XML 2006 Conference*, Boston, USA, December, 2006.

# A Model-driven Approach to Flexible Multi-Level Customization of SaaS Applications

Zakwan Jaroucheh, Xiaodong Liu, Sally Smith  
School of Computing  
Edinburgh Napier University, UK  
{z.jaroucheh, x.liu, s.smith}@napier.ac.uk

**Abstract**—Recently, Software as a Service (SaaS) has become a popular software service mode. Context-awareness and customizability are important and desirable properties for providing the same application for different customers. Most of existing approaches tackle application customization by explicitly specifying some form of variation points where parts of the application remain unspecified or are defaulted and can be customized by each customer to suit its particular needs. This, however, leads to a mismatch between how the architect or developer logically views and interprets differences in a SaaS application family and the actual modeling constructs through which the logical differences must be expressed. Hence, in order to capture the variability in SaaS applications in a more logical and independent way we propose the concept of change fragment and change primitives. A novel approach to effective customization of SaaS at levels of control flow and component framework is proposed and evaluated.

**Keywords**—context-awareness; SaaS; MDD; SaaS customization

## I. INTRODUCTION

SaaS, a new delivery model for software, can be characterized as *software deployed as a hosted service and accessed over the Internet* [2]. Indeed, moving from traditional “on-premise” software that is deployed and run in a data center at the customer’s premise, to offering software as a service has a set of advantages for software customers [5] and requires software vendors to shift their thinking in business model and application architecture.

Changing the business model involves shifting the ownership of the software from the customer to an external provider, reallocating the responsibility for the management of the underlying hardware and software infrastructure for that software from the customer to the provider, and reducing the cost of providing software services, through specialization and economy of scale. From an application architect's point of view, the service provider provides the same application for several different customers; and thus the software must be built following the model of a multi-tenant architecture [2]. However, each individual tenant or customer has different requirements for the same application logic. To achieve this, the customer will be provided by an *application template* [5] in which some parts of the application remain unspecified or are defaulted and can be customized by each customer to suit their particular needs.

On the other hand, context-awareness refers to the capability of an application or a service being aware of its physical environment or situation (e.g. context) and to respond proactively and intelligently based on this awareness [1]. We define the context-aware SaaS application customization as: *the modification of an application behavior as to make it suitable to the special and unique needs of the customer and her context considered relevant to that application.*

Many different solutions have been proposed by researchers to the problem of context-aware customization during SaaS application development and provision. However, two main issues could be identified in the existing approaches.

Firstly, SaaS application modeling must be flexible enough to deal with constant changes – both at the business level (e.g. evolving business rules) and at the technical level (e.g. contextual information). The flexibility could be provided or addressed by incorporating variabilities into a system [5]. Most of the approaches tackle SaaS application customization by explicitly specifying some form of variation points. These approaches (e.g. [4][5]) first identify the variation points in the SaaS application and its associated alternatives (variants) that specify different implementation of the system. Second, either at design time or at runtime, they specify variants selection mechanisms (based on ranking rules, preferences, etc). These approaches enjoy the inherent power of a software product line in dealing with variability, automation and consistency. However, the problem is that, for example, each task in the application is modeled as a variation point in and of itself, each governed by its own clause to determine inclusion or exclusion. This is in contradiction with how the developer or architect logically views the application variant i.e. in terms of the features that determine the difference between application variants in each usage context. Moreover, these approaches are not scalable enough because of the difficulty in variation management when the number of variation points increase.

Secondly, SaaS are built following service oriented architecture (SOA). An application is a composition of services that is orchestrated by workflows such as the standard Business Process Execution Language (BPEL). Thanks to SOA, SaaS applications enjoy the power of programming in the large (i.e. service orchestration) which is separated from the programming in the small (i.e. service implementation). Thus, a deep customization can occur when considering the two basic levels: that of the *individual service* and the *service*

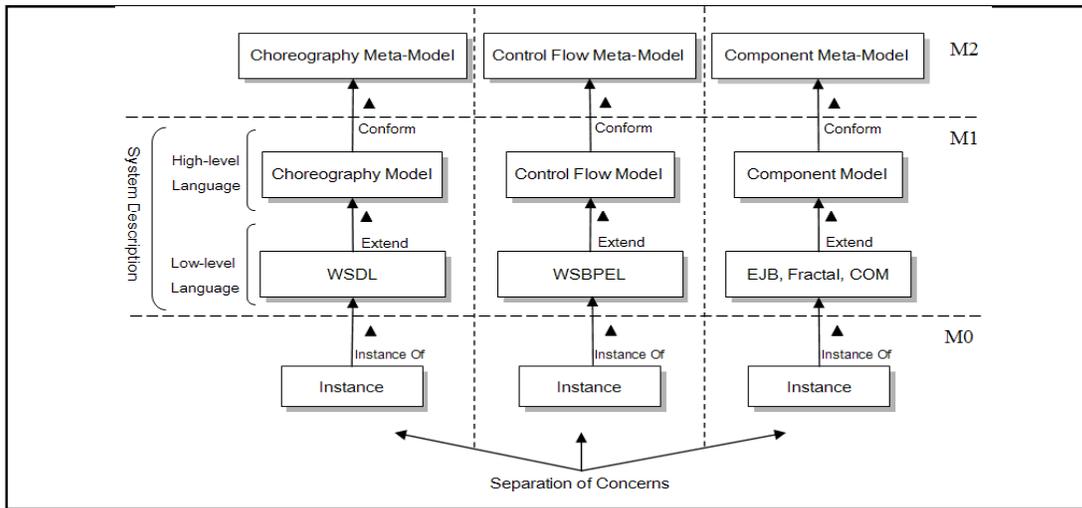


Figure 1. Levelled views of SaaS application

*orchestration*. The approach presented in this paper therefore could be applied to the layers of the SaaS application as long as this layer has been modeled as we will see later.

Motivated by these problems and directives in mind, we propose an MDD-based approach that introduces the notion of *change fragment* and *change primitive* to capture the variability in a more logical and independent form. The proposed approach contributes to a solution to automatically generating a customized SaaS application based on the relevant context.

The rest of the paper is structured as follows: Section II describes the rationale behind the proposed approach. Section III presents the proposed approach for the SaaS application customization. In Section IV, we describe how to instantiate a SaaS application. In section V we present the proof-of-concept prototype; and in section VI we illustrate the proposed approach by giving a simple example of an event advisor. The related work and concluding remarks end the paper.

## II. THE RATIONALE OF THE PROPOSED APPROACH

In the context of SaaS applications, the SaaS developer has to include not only business process in a process language (such as BPEL), but also business rules, policies, constraints, as

well as customization mechanisms. Obviously, mixing process with business rules and customization issues weaken the modularity of the system. Typically, according the separation of concern principle, the SaaS application developer has to focus on the core application business logic and then define separately the customization and business rules, and weave them to the core application.

Therefore, modularization and separation of concerns are the driving principles of our approach to target the SaaS application customization. We propose a model-driven development (MDD) approach for developing such applications. MDD emphasizes using models to capture the application knowledge that are independently of any underlying computing infrastructure, e.g. middleware, programming languages operating systems etc; which will ease the reuse, adaptation, and evolution of applications.

As the number of services involved in a SaaS application grows, the complexity of developing and maintaining these applications also increases. One of the successful approaches to managing this complexity is to represent the application by different architectural views [8]. Examples of these views are orchestration view, control flow view, and component view (see Fig. 1). The idea is to give the developer the possibility of applying the necessary change fragments in each view and then the automated tool verifies the integrity of the changes and

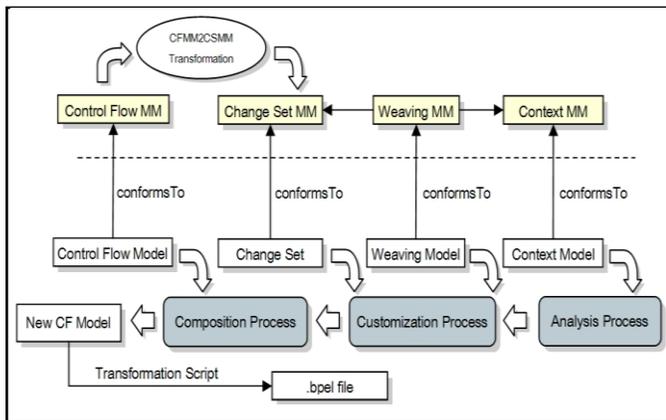


Figure 2. The proposed framework

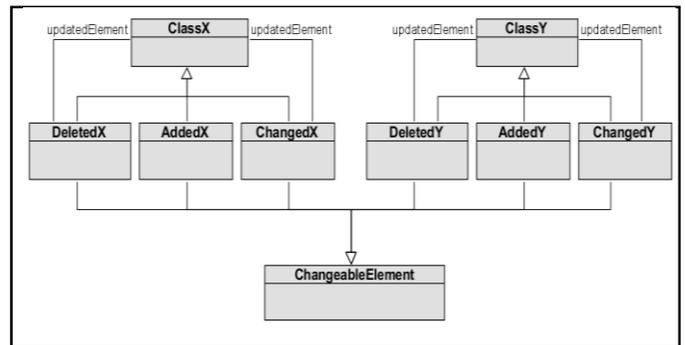


Figure 3. Deriving the change meta-model

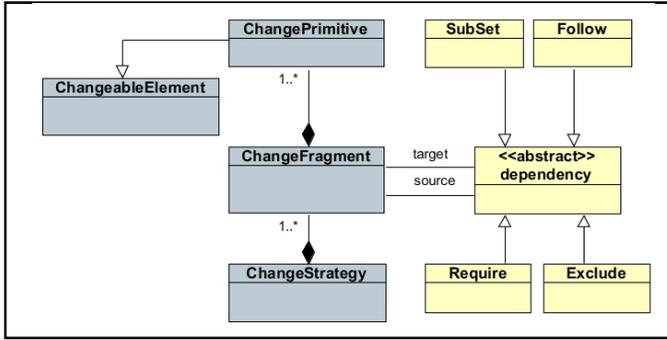


Figure 4. The change set meta-model

generates the customized application variant artifacts accordingly. This modelling respects the separation of concern principle so that we will have multiple views of the systems; each view will model a specific concern.

In this paper we focus on the control flow view; however the proposed approach could be extended to the other views. In addition, the approach aims to tackle context-aware customization without interfering with the core functionality of the application.

### III. THE PROPOSED APPROACH

Typically, the developer first focuses on the functional (business logic) aspect of the SaaS application which will yield a basic model of the system. Then, he defines the change set and the different possible context scenarios. "Weaving" a group of correspondent change fragments with the basic model will yield a new SaaS application instance.

The proposed conceptual model is structured in four main sections that address, respectively, the modeling of the control flow, context information, change set and the weaving model that links between change model and context model (Fig. 2).

During runtime, the customer and environmental context will be gathered when the SaaS application is invoked by the customer. The Context Analysis module evaluates all context constraints of the context model. Using the constraints elements evaluated to "true" and the weaving model we are able to determine the relevant change fragments (See Section IV) and the order in which they should be applied to the basic control flow model.

The customization process determines -according to the mapping between the change fragments and context elements- the set of change fragments to be applied to the application instance. The model composer combines the set of change fragments to the control flow model and verifies the integrity of the system. The result is a new control flow model which corresponds to the current context. All these operations are fulfilled in the model level. Thus, the resulting process model has to be translated to concrete artifacts e.g. BPEL. Now it is the role of the infrastructure to create a new instance corresponding to the new control flow model which satisfies the user requirements and context. This transformation from the model to the code is achieved using one of the model-to-text transformation tools.

```

create OUT : ChangeMM from IN1 : ControlFlowMM, IN2 :
MinimalChangeMM;
helper def: changeableElement: MinimalChangeMM!EClass =
MinimalChangeMM!EClass.allInstances()->select(i | i.name =
'ChangeableElement');

rule copyChangeEvolutionMM {
from s : MinimalChangeMM!EClass
to t: ChangeMM!EClass (
name <- s.name,
interface <- s.interface,
eSuperTypes <- s.eSuperTypes,
eStructuralFeatures <- Sequence {s.eStructuralFeatures}
...
)
}

rule generateChangeMMElements {
from s : ControlFlowMM!EClass (s.name << 'Process' and not
s.abstract)
to t: ChangeMM!EClass (
name <- s.name,
interface <- s.interface,
eSuperTypes <- s.eSuperTypes,
eStructuralFeatures <- Sequence {s.eStructuralFeatures}
...
),
added_element: ChangeMM!EClass (
name <- 'Added' + s.name,
eSuperTypes <- Sequence {t, thisModule.changeableElement}
),
changed_element: ChangeMM!EClass (
name <- 'Changed' + s.name,
eSuperTypes <- Sequence {t, thisModule.changeableElement}
),
deleted_element: ChangeMM!EClass (
name <- 'Deleted' + s.name,
eSuperTypes <- thisModule.changeableElement
)
}

```

Figure 5. Change metamodel generation script

The most interesting aspect of the proposed approach is that the relevant characteristics of the different concerns (context, change, application logic) will be abstracted in models and made observable to the application developers. Moreover, she will be able to do any type of change to the application model. In this way, developers will be able to manage customizability to a greater degree of flexibility.

#### A. Context Model

As in previous work [3], the main construct for representing context knowledge is the `ContextPrimitive` which represents the base context constructs (primitives): entity classes, entity attributes and entities associations. Also, we presented an approach for context-aware software development based on a flexible product line based context model which significantly enhances reusability of context information by providing context variability constructs to satisfy different application needs.

In addition, we introduce here the *context-dependent constraint* construct which allows us to specify conditions that must hold to introduce some kind of context-aware customization by specifying the change fragments (CFs) that should be applied to the basic model as described in the next sections. The Object Constraint Language OCL language is leveraged to express the constraint expression.

#### B. Change Set Model

Customizing SaaS applications usually involves adding, dropping and replacing tasks in the application. For instance, in component model this involves adding, dropping and replacing components. In this respect, and in order to achieve deep

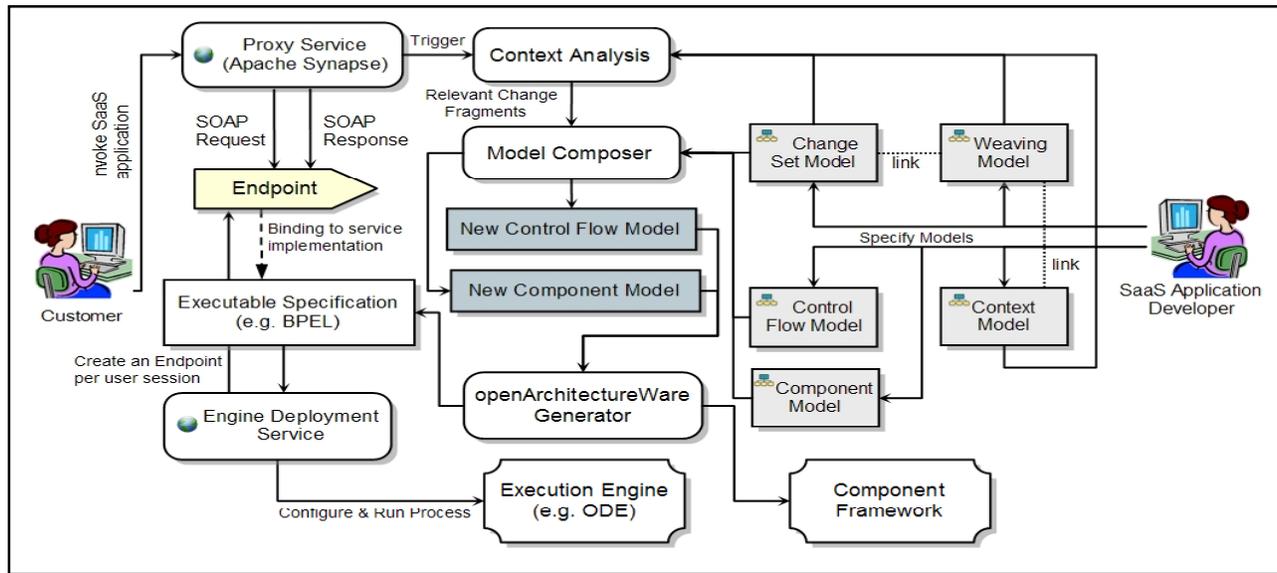


Figure 6. The prototype architecture

change ability, we propose to add for each component  $x$  three classes:  $AddedX$ ,  $DeletedX$ , and  $ChangedX$  describing the difference between the basic component model and the respective variant model (Fig. 3). Other change types can be mapped to variations and combinations of these ones. For instance to achieve the plugin and plugout capability a combination of  $DeletedX$  and  $AddedX$  could be used. In the same manner, for each class  $Y$  in control flow metamodel we add three classes  $AddedY$ ,  $DeletedY$ , and  $ChangedY$ . The change set metamodel (Fig. 4) consists of a `ChangeStrategy` class that contains one or more `ChangeFragments`. The `ChangeFragment` in turn consolidates related `ChangePrimitives` (a set of elements of type `ChangeableElement`) into a single conceptual variation. Our approach promotes CFs to be first-class entities consisting of closely-related additions, deletions and changes performed on the basic model. Dependencies are used to describe relations between CFs in order to constrain their use. The relations supported are as follows: dependency (*Require*), compatibility (*Exclude*), execution order constraint (*Follow*), and hierarchy (*SubSet*). Further, the CF concept is used to specify the application customization during runtime namely the customization strategy. But, what about the evolution of the customization strategy? This is the role of the change strategy concept. An example of strategy evolution is that the business owner may choose to apply a different customization strategy during the Christmas days and later to return to the basic strategy. To this end, the change strategy could also be linked to a specific context constraint.

The change set metamodel of each view could be automatically generated from the model of that view. One possible approach is to use the ATL transformation language<sup>1</sup> as in the script of Fig. 5.

On the other hand, in order to link the context model and change set model, and because in the MDD world everything

should be a model, the mapping between the context constraints and the CFs will be represented by a weaving model. This mapping will be used as information for driving the model transformation.

#### IV. SAAS APPLICATION INSTANTIATION

The selection of a SaaS application variant in a particular context should be done automatically. Therefore the application context in which this selection takes place has to be considered. It is important to distinguish here between two type of changes: i) permanent change which lead to change of the SaaS application specification (structure and behavior) due for instance to the business rules or application logic, and ii) instance-level change which affect only the current application instance. We generate the customized control flow model by applying a number of CFs and their related change primitives to the corresponding basic model as follows: i) Select the CFs whose context constraints associated with it evaluate to “true”, ii) Check CFs relations to ensure model consistency, iii) Apply the CFs to the basic model, and iv) Check for consistency to avoid any deadlock or data inconsistency in the resultant application variant. A consistency check is necessary and it is considered for our future work.

The proposed approach is flexible enough to accommodate the “permanent changes” that are due to changes of the regulation or the business rules by assigning them to a context constraint always evaluated to true. One of the advantages of this approach is that the change in the SaaS application specification can be easily documented.

#### V. PROTOTYPE ARCHITECTURE

We have developed a Java application for the SaaS application variant generation. The Eclipse Modeling Framework (EMF) was used to model the aforementioned models. In this prototype we consider both the control flow view model and component model of the SaaS application.

<sup>1</sup> ATL Language <http://www.eclipse.org/m2m/atl/>

Having specified these models, the developed application generates a context-aware customized control flow model or new plug-in/plug-out component model based on customer request (Fig. 6). The customer request for the SaaS application is intercepted by the Proxy service which in turn triggers the Context Analysis module which evaluates all context constraints of the context model. The Model Composer module applies only those CFs relevant in the SaaS application usage context to the basic control flow model and component model. The resultant control flow model and component model are automatically transformed, using a set of transformation rules, to generate the executable specification of the target platform i.e. BPEL, or component framework. They will be dynamically deployed to the execution engine; and the customer request is then transferred to the new deployed and personalized process.

### VI. CASE STUDY

To demonstrate the approach a small case study is done, namely, the Event Advisor application. This application provides the conference attendee (the customer) with a personalized suggestion for the conference events (i.e. paper, poster, and industrial demo presentations) according to the customer preferences and context. We consider a generic service application that customers can access through a wireless connection using their own portable devices. Fig. 7 depicts a part of the static structure of this application. This application could be enhanced by automatically filling in the ClientType parameter, using for this purpose information provided by the context infrastructure. Being a research-oriented customer means that she is not interested in getting suggestions for the industrial demos. Therefore there is a need to change the process structure so that the activity that invokes the IndustrialDemo is deleted.

Fig. 8 shows a simple example of the context model that contains two entities: Alice and Bob. The association elements assign the attributes to the entities so that Alice has an attribute ClientType whose value is ResearchOriented whereas Bob's ClientType is IndustrialOriented. The context constraint named CustomerIsResearchOriented is an example of the constraints having OCL-based parameterized expression. It

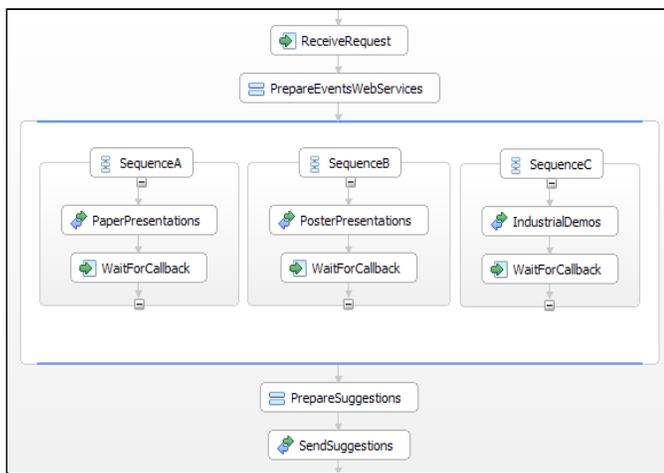


Figure 7. Event advisor application

contains a variable named \$CustomerName whose value is extracted either from the customer request information or from any other data source. In either case the above-mentioned proxy service is responsible for assigning the variable value.

Fig. 9 shows a sample of the change fragments cf1 that regroups different change primitives that should be applied when the customer type is research oriented. The weaving model (Fig. 10) contains one link element that links between the context constraint named CustomerIsResearchOriented and the CF named cf1. Finally, the developed prototype will generate the customized process which contains only the suggestions for paper and poster presentation events.

### VII. OVERHEAD EVALUATION

In this experiment, the cost of generating the customized control flow model is evaluated in terms of response time using a Pentium4 PC with RAM of 3GB. For this purpose we use an example of a control flow model consisting of twenty activities. In each experiment we increment the number of change primitives to be applied to the basic model and measure the time required to derive the BPEL artifact and to deploy the new

```
<ctxt:ContextModel xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:ctxt="http://napier.ac.uk/context">
  <associations name="Alice_attributes"
entities="//@entities.0" attributes="//@attributes.0"/>
  <associations name="Bob_attributes" entities="//@entities.1"
attributes="//@attributes.1"/>
  <entities name="Alice"/>
  <entities name="Bob"/>
  <attributes name="ClientType" value="ResearchOriented"/>
  <attributes name="ClientType" value="IndustrialOriented"/>
  <contextconstraints expression="associations->select(a |
a.entities->exists(e | e.name='$CustomerName') and
a.attributes->exists(al |al.name = 'ClientType' and
al.value='ResearchOriented'))"
name="CustomerIsResearchOriented"/>
  ...
</ctxt:ContextModel>
```

Figure 8. The context model

```
<cs:ChangeStrategy xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:cs="http://napier.ac.uk/cs">
  <changeFragments name="cf1">
    <children xsi:type="cs:DeletedSequence"
updatedElement="SequenceC"/>
    <children xsi:type="cs:DeletedCopy"
updatedElement="DemosSuggestion"/>
    <children xsi:type="cs:ChangedCopy"
updatedElement="SuggestionResponse">
      <to variable="..." part="suggestionsData"/><from
literal="..." />
    </children>
  </changeFragments>
</cs:ChangeStrategy>
```

Figure 9. The change strategy model

```
<weaving:WeavingModel xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:weaving="http://napier.ac.uk/weaving">
  <links name="l1"
contextConstraintName="CustomerIsIndustrialOriented"
changeFragmentName="cf1"/>
</weaving:WeavingModel>
```

Figure 10. The weaving model

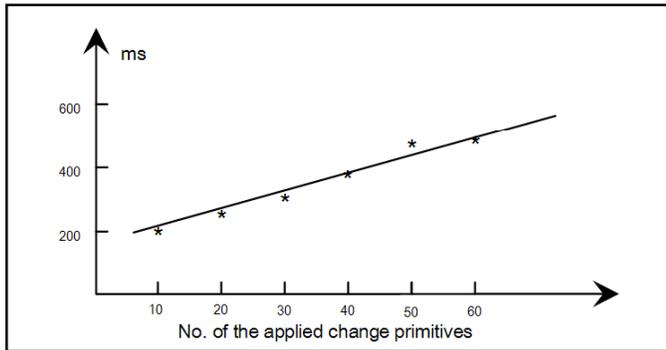


Figure 11. The overhead evaluation

process into the BPEL engine. Fig. 11 shows that the overhead is negligible. For 10 change primitives the overhead is around 200ms. For 60 change primitives the overhead is just less than 0.5s which still acceptable in SaaS kind of applications.

### VIII. RELATED WORK

One of the most successful research directions in the field of software engineering and particularly in software reuse was the software product line SPL (e.g. [6]). Variation points are one of the key concepts in SPL to express variability. However, as aforementioned capturing the application variability using the change fragments and primitives is more intuitive and logical from the developer point of view.

In [5] the authors present an approach that allows the generation of customization processes out of variability descriptors. The proposed approach is different in the way it presents the variation points and variants. It regroups the different variants into more abstract and meaningful constructs to ease the adjustments of the basic application.

Similar to the proposed approach, Provop [7] provides a flexible solution for managing process variants following an operational approach to configure the process variant out of a basic process. This is achieved by applying a set of well-defined change operations to it. However, the proposed approach deviates from Provop in that it uses the MDD approach and defines the CFs as change model elements not as change operations.

VxBPEL [4] is an adaptation language that is able to capture variability in processes developed in the BPEL language. VxBPEL provides the possibility to capture variation points, variants and realization relations between these variation points. Unlike the proposed approach, VxBPEL works on the code level and the variants are mixed with the process business logic which may add complexity to the process developer task. Further, unlike the proposed generative approach, VxBPEL is specific to BPEL language.

### IX. CONCLUSION

The challenge for the SaaS architect is to ensure that the task of configuring applications is simple and easy for the customers, without incurring extra development or operation costs for each configuration. Therefore, we have described a MDD approach for managing and automatic generating customized SaaS application variants. Based on logically-viewed well-defined CFs and change primitive constructs; on the ability to regroup CFs in reusable components; and on the ability to regroup these components in a constrained way, necessary adjustments of the basic application can be correctly and easily realized when creating or configuring an application variant. The proposed approach may provide the possibility of “plugging” more easily within the same basic application different customization strategies tailored for different contexts. Future work includes providing the developer with verification tools to verify the change fragments composition regarding the application consistency and integrity at design time. This will give the developer the flexibility to define profound changes to the SaaS applications in different views.

### REFERENCES

- [1] M. Baldauf, S. Dustdar, and F. Rosenberg, "A survey on context-aware systems," *Int. J. Ad Hoc and Ubiquitous Computing*, vol. 2, 2007.
- [2] F. Chong and G. Carraro. "Architecture Strategies for Catching the Long Tail". MSDN Library, Microsoft Corporation, April, 2006.
- [3] Z. Jaroucheh, X. Liu, and S. Smith, "CANDEL: Product Line Based Dynamic Context Management for Pervasive Applications," *Int. Conference on Complex, Intelligent and Software Intensive Systems*, 2010.
- [4] M. Koning, C. Sun, M. Sinnema, and P. Avgeriou, "VxBPEL: Supporting variability for Web services in BPEL," *Information and Software Technology*, vol. 51, 2009, pp. 258-269.
- [5] R. Mietzner and F. Leymann, "Generation of BPEL Customization Processes for SaaS Applications from Variability Descriptors," *2008 IEEE International Conference on Services Computing*, 2008.
- [6] K. Pohl, G. Bockle, and F. van der Linden. "Software Product Line Engineering: Foundations, Principles, and Techniques". Springer, 2005.
- [7] M. Reichert, S. Rechtenbach, A. Hallerbach, and T. Bauer, "Extending a Business Process Modeling Tool with Process Configuration Facilities: The Provop Demonstrator," *BPM'09 Demonstration Track, Business Process Management Conference*, Germany, 2009.
- [8] H. Tran, U. Zdun, and S. Dustdar, "View-based and Model-driven Approach for Reducing the Development Complexity in Process-Driven SOA," *Int. Conference BPSC'07*, 2007, pp. 105-124.

# Weaving Functional and Non-Functional Attributes for Dynamic Web Service Composition

Ajay Bansal, Srividya Kona  
Georgetown University  
{bansal, kona}  
@cs.georgetown.edu

M. Brian Blake  
Notre Dame University  
M.Brian.Blake@nd.edu

Gopal Gupta  
The Univ. of Texas at Dallas  
gupta@utdallas.edu

## Abstract

*Given the numerous, potentially reusable, Web services available on the Internet, search and composition techniques that efficiently discover viable services will be a strong requirement. A major challenge for dynamic Web service composition will be the ability to measure the quality or reliability of services that are delivered. In this paper, we present a solution for dynamic Web service composition that leverages non-functional attributes provided in the form of Service-Level Agreements (SLA's). The objective of our work is to understand the most efficient algorithms for discovering and composing web services into capabilities with predictable quality. As such, we analyze different approaches to composition when web service composition routines must take into account functional and nonfunctional information. We present our algorithm, a prototype implementation, and experimental results obtained from the various approaches to "weaving" attributes (of different dimensions) as a part of the composition process.*

## 1. Introduction

Automatic composition of Web services has been an active area of research [11, 12], where various methodologies and techniques to dynamically or semi-automatically create composite services have been investigated. The current challenge in automatic composition of Web services also includes finding a composite Web service that can be trusted by consumers before using it. Hence there is a need to consider non-functional attributes of Web services when composite services are being built. The non-functional attributes can be represented in the form of a Service-Level Agreement (SLA), i.e., a formal agreement or contract between a consumer and a service provider or between two service providers [9, 10]. The agreement is on the level of service that is to be provided in terms of the availability, performance guarantee, or other attributes of the service. Web Service-Level Agreement (WSLA) [1] is a standard for unambiguous and clear specification of Service Level Agreements that can be monitored by the service provider, customer, or a third party for compliance. Web Service Agreements (WS-Agreements) [2] is an XML standard and

Web services protocol used for advertising the capabilities of resource providers, establishing agreement between two parties (such as, a service provider and a consumer) and for monitoring compliance at runtime.

This paper investigates the following research issues in designing a dynamic and trusted Web service Composition:

(i) Obtain only those composition solutions that satisfy the service-level agreement parameters specified in the query along with the functional requirements.

(ii) Is it good to obtain a composition solution based on functional attributes first, and then apply non-functional attributes (i.e., attributes specified in a Service-level agreement of the query) to further prune the results?

(iii) Can we get better performance by looking at the non-functional attributes first and consider only those services that comply with the service-level agreement provided by the query? Thereby, only the complying services are used for a composition solution.

(iv) Is it better to weave functional and non-functional attributes while looking for a composition solution?

(v) Can we get better performance by using a subset of non-functional attributes in the first stage of filtering and then obtain composition solutions based on functional attributes. At the end, perform another pass to further narrow down results based on the remaining non-functional attributes that were not used in the first pass.

Our aim is to find the best possible approach to provide the user with a dynamic and trusted composition in an efficient manner.

Our research makes the following novel contributions:

(i) Formalization of Dynamic Web Service Composition that considers Service-Level agreements along with functional attributes of a service.

(ii) Algorithms for different approaches to applying functional and non-functional attributes during the generation of a composition solution.

(iii) Prototype implementation of the algorithms and performance results.

The rest of the paper is organized as follows. In the next section we present related work followed by the background material on composition and SLA's. In Section 4, we present our approaches to finding dynamic composition solutions. In Section 5, we present the experimental results for each of the approaches described in Section 4. Finally, we present our conclusions/future work.

## 2. Related Work

Dynamic Web service Composition and QoS-aware composition has been active area of research [11, 12]. Research on a QoS-aware composition [5, 8, 13] consider applying SLA's to workflow compositions or Web service compositions, although they do not perform dynamic composition. They use one of the existing composition languages to create the composite service manually or create a template that is later used to select appropriate services for each stage of composition. After obtaining composition solutions manually or semi-automatically, these approaches present a QoS model and apply the non-functional attributes on the potential solutions to confirm that they comply with the pre-defined agreements. Thus, the solutions are pruned based on SLA compliance.

Our previous work on Workflow Composition of Service-level Agreements [3] presents a set of SLA measures and principles that best support QoS-based Composition. A model and representation of SLA attributes was introduced and an approach to compose SLA's associated with a workflow of Web services was presented. The research on creating a QoS-Aware middleware for Web service Composition in [7] is similar to our work as they identify services that can fit into a useful composition based on QoS measures. They use two approaches for selection: one based on local (task-level) selection of services and the second is based on a global allocation of tasks to services. They also use a template for composition, in this case a state chart that has the generic service tasks defined. Finding a composite service involves finding concrete services that fit into the template. In contrast, we do not use any template but instead find the composition solution automatically. The work presented in [6] combine semantic annotations and SLA's thereby providing better approach to specification of SLA's in contrast to our approach that uses SLA's in dynamic service composition.

## 3. Background

In this section we present the necessary background information on Web service Composition and Service-Level Agreements.

### 3.1. Service-Level Agreements

Web Service-Level Agreements provide a standard specification that allows authors of Web services to specify the performance metrics associated with their Web service application. In addition, the authors can also specify the desired performance targets and actions to be taken in case the performance targets are not met.

#### 3.1.1. WS-Agreement

In order to store and manage Service-Level Agreements (SLAs), the attributes must be represented in a format conducive for distributed data management. An XML-based approach to represent SLA information facilitates quick

interpretation of data and use of translation techniques, such as the eXtensible Stylesheet Language, XSL, allows the comparison and margining of the underlying information. Web Service Agreements (WSAG) [2] is based on the WSLA [1] semantics whereas the data is represented hierarchically underneath the notion of an agreement. An agreement can be further specified with a name or identifying string. An agreement can also be described by its context. Context information includes the name of the consumer and producer, the timeframe for the validity of the agreement, and other related template information. Each agreement encapsulates a list of terms. Of most importance to this work are the *guarantee* terms which are described within the terms node. Guarantee terms consist of a service scope which contains the name of the specific service relevant to the guarantee. The service level objective contains a predicate for the metrics that quantitatively define the guarantee. The service level objective contains parameter name, value, and unit of measure.

Capturing service level agreements in XML-based notations allows the SLA attributes to be represented in a format similar to the WS Description Language (WSDL) files that represent the operational specifications of the service. WSAG can be embedded, transported, and negotiated within WSDL files. This represents a benefit if a service stakeholder wants to evaluate multiple services, side-by-side. Another benefit of capturing SLA attributes in XML-based files is the ability of enhancing attributes with semantics. It is possible that organizations will name attributes with their own customized naming schemes. Semantics would allow disparate organizations to mediate SLA attributes that are the same but may be named differently. This approach leverages the numerous projects that use semantics to highlight web services for mediation.

### 3.2. Web Service Composition

Informally, the Web service Composition problem can be defined as follows: given a repository of service descriptions, and a query with requirements of the requested service, in case a matching service is not found, the composition problem involves automatically finding a directed acyclic graph of services that can be composed to obtain the desired service. Figure 1 shows an example composite service made up of five services  $S_1$  to  $S_5$ . In the figure,  $I'$  and  $CI'$  are the query input parameters and pre-conditions respectively.  $O'$  and  $CO'$  are the query output parameters and post-conditions respectively. Informally, the directed arc between nodes  $S_i$  and  $S_j$  indicates that outputs of  $S_i$  constitute (some of) the inputs of  $S_j$ .

### 3.3. Functional and Non-functional Attributes of Web services

#### 3.3.1. Functional Attributes

The generalized Composition problem is defined in our previous work [4] in terms of functional attributes of the

Web service, i.e., inputs, outputs, pre-conditions, post-conditions, and side-effects of the Web service. The generalized Composition problem can be defined as automatically finding a directed acyclic graph  $G = (V, E)$  of services from repository  $R$ , given query  $Q = (CI', I', A', AO', O', CO')$ , where  $V$  is the set of vertices and  $E$  is the set of edges of the graph. Each vertex in the graph either represents a service involved in the composition or post-condition of the immediate predecessor service in the graph, whose outcome can be determined only after the execution of the service. Each outgoing edge of a node (service) represents the outputs and post-conditions produced by the service. Each incoming edge of a node represents the inputs and pre-conditions of the service. The following conditions should hold on the nodes of the graph:

1. for all  $i$ ,  $S_i \in V$  where  $S_i$  has exactly one incoming edge that represents the query inputs and pre-conditions,  $I'$  subsumed by  $U_i I_i$ ,  $CI' \rightarrow \bigwedge_i CI_i$ .
2. for all  $i$ ,  $S_i \in V$  where  $S_i$  has exactly one outgoing edge that represents the query outputs and post-conditions,  $O'$  subsumes  $U_i O_i$ ,  $\bigwedge_i CO_i \rightarrow CO'$ .
3. for all  $i$ ,  $S_i \in V$  where  $S_i$  represents a service and has at least one incoming edge, let  $S_{i1}, S_{i2}, \dots, S_{im}$  be the nodes such that there is a directed edge from each of these nodes to  $S_i$ . Then  $I_i$  subsumes  $U_k O_{ik} \cup I'$ ,  $(CO_{i1} \wedge CO_{i2} \dots \wedge CO_{im} \wedge CI') \rightarrow CI_i$ .
4. for all  $i$ ,  $S_i \in V$  where  $S_i$  represents a condition that is evaluated at run-time and has exactly one incoming edge, let  $S_j$  be its immediate predecessor node such that there is a directed edge from  $S_j$  to  $S_i$ . Then the inputs and pre-conditions at node  $S_i$  are  $I_i = O_j \cup I'$ ,  $CI_i = CO_j$ . The outgoing edges from  $S_i$  represent the outputs that are same as the inputs  $I_i$  and the post-conditions that are the result of the condition evaluation at run-time.

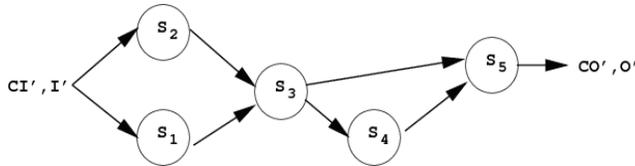


Figure 1: Example of a Composite Service as a Directed Acyclic Graph

The meaning of the symbol  $\rightarrow$  is the implication relation. In other words, a service at any stage in the composition can potentially have as its inputs all the outputs from its predecessors as well as the query inputs. The services in the first stage of composition can only use the query inputs. The union of the outputs produced by the services in the last stage of composition should contain all the outputs that the query requires to be produced. Also the post-conditions of services at any stage in composition should imply the pre-conditions of services in the next stage. When it cannot be determined at compile time whether the post-conditions imply the pre-conditions or not, a *conditional* node is

created in the graph. The outgoing edges of the conditional node represent the possible conditions which will be evaluated at run-time. Depending on the condition that holds, the corresponding services are executed. That is, if a subservice  $S_1$  is composed with subservice  $S_2$ , then the post-conditions  $CO_1$  of  $S_1$  must imply the preconditions  $CI_2$  of  $S_2$ . The following conditions are evaluated at run-time:

- if  $(CO_1 \rightarrow CI_2)$  then execute  $S_1$ ;
- else if  $(CO_1 \rightarrow \neg CI_2)$  then no-op;
- else if  $(CI_2)$  then execute  $S_1$ ;

### 3.3.2. Non-Functional Attributes

A typical SLA has a large number of measures and criteria. However, in this work, we attempt to choose the measures that are most closely aligned to Web service composition. In the general functional notion of Web service composition, a basic Web service workflow system must ensure that the input information supplied by the consumer ultimately leads to the required actions and outputs required by that consumer. In parallel, the workflow management system must ensure that the predicates and requisites match (either by syntactical or semantic techniques) in each step of the workflow. Although the functional Web service composition has been investigated a great depth in literature, the assessment criteria associated with SLA terms increases the requirements for composition threefold.

In this work, we stratify the SLA criteria by introducing three principles associated with composing SLAs, Compliance (Suitability), Sustainability, and Resiliency.

#### i. Compliance (Suitability):

Compliance is the principle that ensures that the consumer receives the requested composite capability at the service level that is required. The functional notion of Web service composition fits within this principle, since the consumer specifies their required outputs of the composition. Considering SLA terms, the composition process must assure that the aggregate cost, uptime, and service rate are compliant with the user requirements. Cost is the sum total price of all services participating in the solution process. Uptime is a guarantee by the service providers that their services will be available a specified percentage of the time per day or month. Finally, service rate is the time it takes to complete the process by adding the response times of each service in the composition.

#### ii. Sustainability:

Sustainability is the ability to maintain the underlying services in a timely fashion. Negotiation, renegotiation and problem resolution are strongly correlated to sustainability. A consumer will require assurance that a particular business is capable of agreeing on contract terms (i.e. negotiation/renegotiation) in a timely manner. In addition, the service providers must be capable of resolving high-impact problems (perhaps identified by the consumer) in a timely manner. Both negotiation and problem resolution

times ensure that a consumer can meet the demands of their end users.

### iii. Resiliency:

Resiliency is the principle of a service to perform at high service levels over an extended period of time. If a service is frequently taken off-line for maintenance or if the frequency of updates impedes the predictability of its operation, then that service is not very resilient.

## 4. Dynamic Web Service Composition

The current challenge in automatic composition of Web services includes finding a composite Web service that can be trusted by consumers before using it. Hence there is a need to consider non-functional attributes of Web services when composite services are being built. In this section we present our approach that considers functional attributes described in a Web service description document and the non-functional attributes described in a WS-Agreement document.

### 4.1 Composition using Functional Attributes

In order to produce the composite service represented as a graph shown in the Figure 1, we use the definition of the generalized Composition presented in section 3.3.1. We filter out services that are not useful for the composition at multiple stages as described in our previous work on a Generalized Semantics-based Service Composition [4, 14].

<i>FuncAttr-First Routine</i>
<p>Algorithm: FuncAttr-First            (Input: QI - QueryInputs, QO - QueryOutputs,            QCI - Pre-Cond, QCO - Post-Cond            Output: FinalResult – ListOfServices)</p> <ol style="list-style-type: none"> <li>1. <math>L \leftarrow \text{NarrowServiceList}(QI, QCI);</math></li> <li>2. <math>O \leftarrow \text{GetAllOutputParameters}(L);</math></li> <li>3. <math>CO \leftarrow \text{GetAllPostConditions}(L);</math></li> <li>4. While Not (<math>O</math> subsumed by <math>QO</math>)</li> <li>5.   <math>I = QI \cup O; CI = QCI \wedge CO;</math></li> <li>6.   <math>L' \leftarrow \text{NarrowServiceList}(I, CI);</math></li> <li>7. End While;</li> <li>8. <math>\text{Result} \leftarrow \text{RemoveRedundantServices}(QO, QCO);</math></li> <li>9. <math>\text{FinalResult} \leftarrow \text{SLACompose}(\text{Result});</math></li> <li>10. return FinalResult;</li> </ol>

Table 1: FuncAttr-First Algorithm

### 4.2 Composition of SLA measures (Non-Functional Attributes)

Our approach also uses SLA's of the individual services involved in the composition and determines the SLA of the composite service obtained. SLA of a service consists of specific Web service-oriented measures, as defined in the research presented in [3].

Let  $A_{s_i}$  be the SLA of a Web service  $s_i$ .  $A_{s_i}$  can be represented as a tuple shown below:

$$A_{s_i} = (Up_{s_i}, SRate_{s_i}, SResp_{s_i}, Cost_{s_i}, PreNT_{s_i}, RenegT_{s_i}, Rep_{s_i}, Rel_{s_i})$$

where  $Up_{s_i}$  is the uptime of the service specified by the agreement,  $SRate_{s_i}$  is the allowable service rate,  $SResp_{s_i}$  is the allowable service response time,  $Cost_{s_i}$  is the subscription cost or price of the service,  $PreNT_{s_i}$  is the maintenance pre-notification time,  $RenegT_{s_i}$  is the expiration/renewal time of the agreement,  $Rep_{s_i}$  is the reputation of the service and  $Rel_{s_i}$  is the reliability rating of the service. The SLA for a workflow composition or a composite web service is obtained by composing the set of SLAs of all the services participating in the composition. Hence the SLA of the workflow composition also consists of uptime of the composite service, the allowable service rate, the service response time, the subscription cost or price, the maintenance pre-notification time, and the expiration/renewal time of the agreement. Computing the SLA of a workflow composition involves computing these individual SLA measures of the workflow that are presented in detail in [3].

### 4.3 Weaving Functional & Non-Functional Attributes

The main focus of our research presented in this paper is to weave the functional and non-functional attributes in a composition algorithm. In our prototype implementation we use a repository of Web service descriptions and their corresponding WS-Agreements. The query requesting a composite Web service has specification of its functional and non-functional requirements. The repository is pre-processed and all the functional (list of inputs, list of outputs, list of pre-conditions and post-conditions and side-effects) and non-functional (various SLA measures) attributes are indexed. Service description documents contain semantic descriptions provided using an ontology. In our prototype, we use a semantic relations generator module that extracts and indexes all the semantic relations. After the pre-processing stage, the composition routine is executed using the query data and all the preprocessed data. We present three different approaches for a dynamic Web service composition.

#### 4.3.1 Approach 1 (FuncAttr-First)

In this approach, functional attributes are used first to filter services and find appropriate services that can form a composite service that matches the query requirements. The repository of service descriptions and service-level agreements is pre-processed and all the functional and non-functional attributes are indexed. Then the multi-stage filtering technique that is described in detail in our previous work [4] is applied to find composition solutions. In this approach, functional attributes that are specified in the query and the service description documents are considered first. The conditions described in the formal definition of the generalized composition problem (presented in section 3.3.1) are applied by the multi-stage filtering algorithm.

After the solutions are obtained, the non-functional attributes such as uptime, cost, renegotiation time, etc. are checked on the composite services obtained to ensure compliance. The SLA of a composite service is determined as described in section 4.2. Table 1 presents the algorithm for this approach.

#### 4.3.2 Approach 2 (NonFuncAttr-First)

In the NonFuncAttr-First approach, after the pre-processing stage, we first filter services based on their individual SLA measures obtained from WS-Agreements and the query requirements. Then the multi-stage filtering technique is applied to the filtered set of Web services to obtain composition solutions as shown in Table 2.

<i>NonFuncAttr-First Routine</i>
Algorithm: NonFuncAttr-First (Input: QI - QueryInputs, QO - QueryOutputs, QCI - Pre-Cond, QCO - Post-Cond Output: Result – ListOfServices)
<ol style="list-style-type: none"> <li>1. FilteredList <math>\leftarrow</math> FilterBasedOnSLAMeasures();</li> <li>2. L <math>\leftarrow</math> NarrowFilteredServiceList(QI, QCI);</li> <li>3. O <math>\leftarrow</math> GetAllOutputParameters(L);</li> <li>4. CO <math>\leftarrow</math> GetAllPostConditions(L);</li> <li>5. While Not (O <i>subsumed by</i> QO)</li> <li>6.   I = QI <math>\cup</math> O; CI = QCI <math>\wedge</math> CO;</li> <li>7.   L' <math>\leftarrow</math> NarrowServiceList(I, CI);</li> <li>8. End While;</li> <li>9. Result <math>\leftarrow</math> RemoveRedundantServices(QO, QCO);</li> <li>10. return Result;</li> </ol>

Table 2: NonFuncAttr-First Algorithm

#### 4.3.3 Approach 3 (WeaveAttr)

In the WeaveAttr approach, after the pre-processing stage both the functional and non-functional attributes are used by the multi-stage filtering technique. In this case, all the conditions specified in the formal definition of a genera-

<i>WeaveAttr Routine</i>
Algorithm: WeaveAttr (Input: QI - QueryInputs, QO - QueryOutputs, QCI - Pre-Cond, QCO - Post-Cond Output: Result – ListOfServices)
<ol style="list-style-type: none"> <li>1. L <math>\leftarrow</math> NarrowServiceList(QI, QCI);</li> <li>2. SL <math>\leftarrow</math> FilterBasedOnSLAMeasures(L);</li> <li>3. O <math>\leftarrow</math> GetAllOutputParameters(SL);</li> <li>4. CO <math>\leftarrow</math> GetAllPostConditions(SL);</li> <li>5. While Not (O <i>subsumed by</i> QO)</li> <li>6.   I = QI <math>\cup</math> O; CI = QCI <math>\wedge</math> CO;</li> <li>7.   L' <math>\leftarrow</math> NarrowServiceList(I, CI);</li> <li>8.   SL' <math>\leftarrow</math> FilterBasedOnSLAMeasures(L');</li> <li>9. End While;</li> <li>10. Result <math>\leftarrow</math> RemoveRedundantServices(QO, QCO);</li> <li>11. return Result;</li> </ol>

Table 3: WeaveAttr Algorithm

lized composition are applied at each stage of composition. Along with those conditions, the SLA measures are also checked for compliance at each stage of the composition. Any services that do not comply with the specified agreement are filtered out. At the end, composition solutions obtained are already SLA compliant. Table 3 presents the algorithm for this approach.

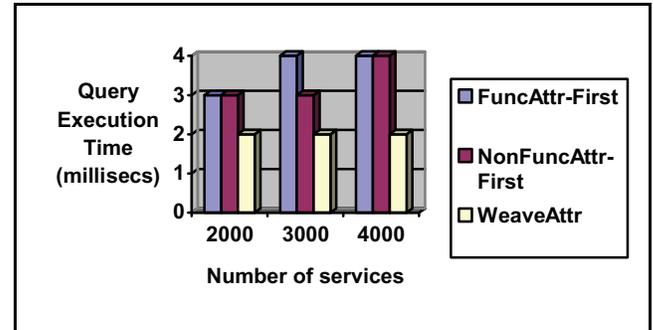


Fig. 2: Query Execution Time (4-8 I/O Parameters)

## 5. Experimental Results

The prototype of a dynamic composition engine presented in section 4 is implemented using Constraint Logic Programming and semantic descriptions of Web services are specified using the language USDL [14]. Our impleme-

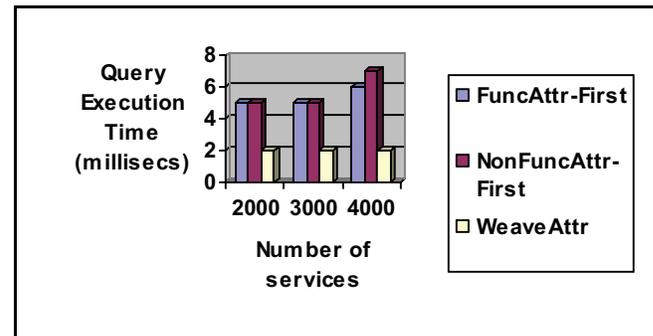


Fig. 3: Query Execution Time (16-20 I/O Parameters)

ntation is generic enough to work with other semantic description languages also. The SLA measures are specified using WS-Agreements. We tested our implementation on repositories of different sizes and with different number of

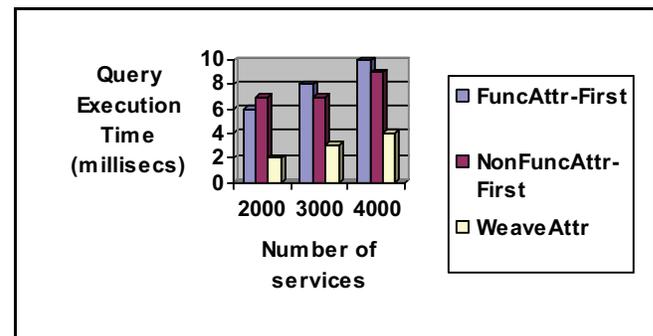


Fig. 4: Query Execution Time (32-36 I/O Parameters)

Repository Size (number of Services)	Number of I/O parameters	Preprocessing time (in secs)	Query Execution Time (in milliseconds) (FuncAttr-First)	Query Execution Time (in milliseconds) (NonFuncAttr-First)	Query Execution Time (in milliseconds) (WeaveAttr)
2000	4-8	39.3	3	3	2
2000	16-20	49.7	5	5	2
2000	32-36	63.5	6	7	2
3000	4-8	61.3	4	3	2
3000	16-20	63.8	5	5	2
3000	32-36	106.1	8	7	3
4000	4-8	73.2	4	4	2
4000	16-20	89.8	6	7	2
4000	32-36	134.6	10	9	4

Table 4: Performance of Composition Routine

input and output parameters. Each of the WS-Agreement documents had six SLA measures which include Uptime, Cost, Service Response Time, Renegotiation time, Maintenance Pre-Notification Time, and Reputation of the service. We noted the pre-processing time for each of the repositories. We performed composition using the three approaches presented in section 4 with various queries and tabulated the query execution times as shown in table 4. The results from approaches 1 and 2 (*FuncAttrFirst* and *NonFuncAttrFirst*) are similar whereas the query execution time of *WeaveAttr* approach is lower than the other two approaches as presented in figures 2, 3, and 4. Hence, the *WeaveAttr* approach seems to be the better approach over the other two.

## 6. Conclusions and Future Work

Due to the growing number of services on the Web, we need automatic and dynamic Web service composition in order to utilize and reuse existing services effectively. It is also important that the composition solutions obtained can be trusted and are reliable. In this paper, we presented a composition technique that uses both functional and non-functional attributes to obtain a trusted dynamic composition. The non-functional attributes are represented as SLA measures in a WS-Agreement document. The experiments show that *WeaveAttr* routine performed better than the other two approaches. In this work we considered six different types of SLA measures. Next, we will explore the possibility of introducing semantics for describing SLA measures. We also plan to extend our approach and implement it in a complete operational setting.

## 7. References

- [1] WSLA: [www.research.ibm.com/wsla/](http://www.research.ibm.com/wsla/)
- [2] WS-Agreement: [www.ogf.org/documents/GFD.107.pdf](http://www.ogf.org/documents/GFD.107.pdf)
- [3] Blake, M.B. "Decomposing Composition: Service-Oriented Software Engineers", Spl. Issue on Realizing Service-Centric Software Systems, IEEE Software, Vol. 24, No. 6, pp 68-77.
- [4] Kona, S., Bansal, A., Blake, M.B., and Gupta, G. Generalized Semantics-based Service Composition in Proceedings of IEEE International Conference on Web Services (ICWS) (September 2008).
- [5] Dong, W., and Jian, L. QoS-Aware Web Service Composition based on SLA in Proceedings of Fourth International Conference on Natural Computation.
- [6] Cardoso, J., Sheth, A.P., Miller, J.A., Arnold, J., and Kochut, K.J. Modeling Quality of Services for Work-flows & Web Service Processes in Science, Services, and Agents on the World Wide Web Journal, 2004, 1(3), pp. 281-308.
- [7] Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., and Chang, H. QoS-Aware Middleware for Web Services Composition. IEEE Trans. on Software Engg. 30(5):311-327, May 2004, IEEE Computer Society.
- [8] Yan, J., Kowalczyk, R., Lin, J., Chhetri, M.B., Goh, S.K., Zhang, J. Autonomous service level agreement negotiation for service composition provision, Future Generation Computer Systems, Vol. 23, No. 6, pp 748-759, July 2007.
- [9] Sorteberg, I., and Kure, O. The use of service level agreements in tactical military coalition force networks, IEEE Communications Magazine, pp. 107-114, Nov. 2005.
- [10] Sun, W., Xu, Y., Liu, F. The role of XML in service level agreements management in Proc. of Intl. Conf. on Services Systems & Services Management, pp. 1118-1120 June '05.
- [11] Rao, J., Dimitrov, D., Hofmann, P., Sadeh, N. A Mixed-Initiative Approach to Semantic Web Service Discovery & Composition in Proc. of Intl. Conf. on Web Services, 2006.
- [12] Claro, D., Albers, P., Hao, J. Selecting Web services for Optimal Compositions in Proc. of Workshop on Semantic Web Services & WS Composition, 2004.
- [13] Wada, H., Champrasert, P., Suzuki, J., and Oba, K. Multiobjective Optimization of SLA-aware Service Composition in Proc. of IEEE Wk. on Methodologies for Non-functional Properties in Services Computing, July '08.
- [14] Kona, S., Bansal, A., Simon, L., Mallya, A., Gupta, G., Hite, T. USDL: A Service-Semantics Description Language for Automatic Web Service Discovery & Composition in Intl. Journal of Web Services Research, Jan. 2009.

# Improving Cluster Selection Techniques of Regression Testing by Slice Filtering

Yongwei Duan<sup>1,2</sup>, Zhenyu Chen<sup>1,2</sup>, Zhihong Zhao<sup>1,2,§</sup>, Ju Qian<sup>3</sup>, and Zhongjun Yang<sup>1,2</sup>

<sup>1</sup> State Key Laboratory for Novel Software Technology, Nanjing University, China

<sup>2</sup> Software Institute, Nanjing University, China

<sup>3</sup> College of Information Science and Technology, Nanjing University of Aeronautics and Astronautics, China

<sup>§</sup> Corresponding author: zhaozh@software.nju.edu.cn

## Abstract

*Test selection technique is used to select a sufficient subset of the existing test suite in regression testing. In this paper, we combine program slicing and cluster analysis to improve regression test selection techniques. A static slice is computed from the modified code. The intersection of the program slice and the execution profile of each test case, which is considered as a lightweight dynamic slice, is computed to highlight the parts of software affected by modification. It reduces the size of the execution traces of test cases significantly. Then cluster analysis is conducted on these intersection features and few test cases will be sampled from each cluster to form a small regression test suite. The slice filtering gives more potential chances to deal with large software by reducing the dimensions of cluster analysis. The experiment results show that fault-revealing test cases tend to cluster together after slice filtering and a regression test suite sampled from these clusters has sufficient fault detection capability.*

**Keywords:** test case selection, program slicing, cluster analysis, cluster filtering, regression testing

## I. INTRODUCTION

Due to the expensiveness of regression testing, test selection techniques are employed to reduce the cost of retesting a modified program by selecting a subset of existing test suite to run. Regression test selection techniques reduce the size of test suite, but they may omit fault-revealing test cases and therefore lose some fault detection capability.

Many regression test selection techniques have been proposed. Fischer [7] used the systems of linear equations to select test suites that yield segment coverage of modified code. Yau and Kishimoto [20] used input partitions and data-driven symbolic execution techniques to achieve test case selection. Leung and White [9] presented a technique that placed a firewall around modified modules and selected unit tests for modified modules that lay within the firewall and integration tests for groups of interaction modules that lay within the firewall. Chen [4] developed a technique to select test cases that traverse the modified code entities such as functions, variables, etc. Rothermel et al. [14] proposed a safe test selection technique. A selection method is safe if it selects all the fault-revealing test cases. However, a safe technique may select many non-fault-revealing test cases such that test resources are wasted.

Dickinson [5,6] et al. used cluster analysis on execution profiles induced by the original test cases. It was expected that test cases

with similar behaviors would be grouped into the same cluster. One or more test cases were sampled from each cluster and manually checked whether they are fault-revealing test cases. This so-called observation-based testing [5,6]. Yan et al. [19] proposed a new test case sampling strategy by leveraging test case spectra information to select test cases that are most likely to be failed. Recently, Zhang et al. [23] introduce cluster analysis in regression testing to improve the precision of test selection techniques.

Many properties of the program's execution could be profiled to characterize the behaviors of test cases. Such properties include: statement, function, basic-block, data-flow, control-flow, event sequence, etc. [10,21]. Although Zhang [23] achieves a satisfactory result, there is a main shortcoming of it: the function call traces are too simple to capture the behaviors of test cases. However, if we use more detailed properties, such as statement or basic-block, it may be infeasible for cluster analysis with a high dimensional cluster space for large software. A main challenge of further improvement is to select fewer property features to highlight the parts of software affected by modification, such that cluster filtering can be efficiently conducted on large software.

Agrawal [1] discussed the application of execution slice in regression test selection. The set of statements executed under a test case is referred to as the execution slice of the program with respect to a test case. During the off-line processing [1], the execution slices of the program with respect to all test cases in the regression test suite are collected. After the program is modified, only those test cases whose execution slices contain a modified statement are selected to rerun. This is a typical application of program slicing in regression testing.

Program slicing can remove the irrelevant code with respect to the modification, such that it not only can reduce the dimensions of cluster analysis, but also can highlight the parts related to the modification. In this paper, we propose a new test selection approach that combines program slicing and execution profiles before cluster analysis. The cluster filtering of regression test selection is based on statement level execution traces, but we only used the modified statements and statements affected by the modification.

The remainder of this paper is organized as follows: in the next section, our approach is described in detail. The experiment and its result analysis are done in Section III. Section IV concludes this study.

---

The work described in this article was partially supported by the National Natural Science Foundation of China (90818027, 60803007) and the Opening Project of Shanghai Key Laboratory of Computer Software Evaluating & Testing (09DZ2272600).

## II. OUR APPROACH

### 1. Overview

We use the following notations throughout the rest of this paper:  $P$  denotes the original version of a program,  $P'$  denotes its modified version. Before the discussion of our approach, several definitions are given as following. A test case is fault-revealing if it detects a failure in a program. A test case is modification-traversing if and only if it executes new or modified code, or executes formerly executed code that has been deleted. For statement  $s$  and variable  $v$ , the slice  $S$  of program  $P$  with respect to the slicing criterion  $\langle s, v \rangle$  includes only those statements of  $P$  needed to capture the behavior of  $v$  at  $s$  [18, 2, 22]. In other words, the slice  $S$  of program  $P$  includes those statements that are directly or indirectly dependent on the value of variable  $v$  at statement  $s$ .

Dickinson [5, 6] shows that fault-revealing test cases attributed to the same fault exhibit similar behaviors and share similarity in their execution profiles. Our approach employs this observation and can be presented briefly as following: let  $T = \{t_1, t_2, \dots\}$  denote the existing test suite, where  $t_i$  denotes a single test case. The execution traces set are denoted by  $E = \{e_1, e_2, \dots\}$  where  $e_i$  is the corresponding execution trace of  $t_i$  collected previously. Let  $M = \{s_1, s_2, \dots\}$  denote the set of the statements been modified in  $P'$ . Let  $S$  denote program slice with respect to the modification locale in  $M$ . To filter out the unaffected statements, we intersect  $S$  with each execution trace  $e_i$  in  $E$ . To reduce the traces further, statements belonging to a basic-block are condensed so that only the first statement of the basic-block is remained. We refer the final set of execution traces as  $F = \{f_1, f_2, \dots\}$ , and we refer  $f_i$  as the *feature* of test case  $t_i$ . A cluster analysis is conducted on  $F$  in the following step. At last, a sampling strategy is employed to select potential fault-revealing test cases. We assume that test cases are independent in this paper.

```

void f(int m, int n) {
1.  int a=1, b=0, c=0, d=0;
2.  if( m < n )
3.      a = 10;
   else
4.      a = 20;
5.  while(a>0) {
6.      c = c + a;
7.      a = a - 1; }
8.  printf("%s", c);
9.  if( n == 0 ) {
10.     b = 10;
11.     while( b>0 ) {
12.         d=d + b;
13.         b= b - 1; }
14.     printf("%s", d); }
}

```

Figure 1. An example

We construct a simple program to demonstrate the necessity of the filtering process using program slice, as is shown in Figure 1. Table 1 gives 3 test cases of this program. Suppose the statement  $s_2$  “if ( $m < n$ )” is changed to “if ( $m <= n$ )”. The execution traces for each test case are shown in Table 1.  $t_1$  and  $t_2$  are similar since they share many statements in their execution traces. Their execution traces differ in only one statement.  $t_3$  is dissimilar to both  $t_1$  and  $t_2$  since its execution trace is quite different from both  $t_1$  and  $t_2$ . However, if we compute the forward and backward slice at statement  $s_2$ , which, in statement level, is  $\langle s_2, s_3, s_4, s_5, s_6, s_7, s_8 \rangle$ , and intersect it with each

test case’s execution trace to filter out the irrelevant code, the result is quite different as is shown in Table 1. Traces of  $t_2$  and  $t_3$  are actually the same by filtering, and the difference between  $t_1$  and  $t_2$  is magnified since those statements that are irrelevant of  $s_2$  are filtered out. By focusing on only the affected code, the distances between each test case are justified in the clustering space.

Table 1. 3 Test cases

Test cases	Input		Execution trace (Statement no.)	Statement no. by filtering
	m	n		
$t_1$	1	0	1,2,4,5,6,7,8,9,10, 11,12,13,14	2,4,5,6,7,8
$t_2$	-1	0	1,2,3,5,6,7,8,9,10, 11,12,13,14	2,3,5,6,7,8
$t_3$	-1	1	1,2,3,5,6,7,8,9	2,3,5,6,7,8

This intersection of static slice and execution trace could be considered as a lightweight dynamic slice. The exact dynamic slice is more precise than static slice. But the computation of dynamic slice is always more complex and time-consuming than static slice. The historical execution profiles, such as covered statements, are collected during regression testing. Assisting a low cost static slice, the parts of software affected by modification are computed easily.

Please note that our approach is not safe due to the indeterminacy of cluster analysis and randomness in a sampling strategy. This un-safeness means our approach may miss some fault-revealing test cases. However, in a resource-limited testing scenario, running all the test cases selected by a safe regression test selection techniques may be superfluous. On the other hand, an appropriate subset of all fault-revealing test cases is sufficient to detect and debug the faults in many cases. Our technique selects a significantly smaller subset than a safe selection technique does, thus achieving a better reduction of the set of potential test cases to be run. Moreover, with an appropriate sampling strategy and sampling rate, our approach can select approximately the same number of fault-revealing test cases as a safe selection technique does. And the number of the falsely selected non-fault-revealing test cases is very low in most cases.

By applying slice filtering, our approach achieves high performance over large sized program. In Section III, we conduct an empirical study on a relatively large sized program named *space*. On the other hand, our approach has certain limitation when dealing with small sized program. This is because execution profiles of test cases on small programs tend to be alike. Therefore, bug-revealing test cases are not isolated into small clusters and are unlikely to be sampled out.

### 2. Execution Traces Collection

Test suite  $T$  is executed on program  $P$  and the statement level coverage information is collected using GNU call-coverage profiler, *gcov* [11]. This information is then stored to form the execution traces  $E$ . An execution trace  $e_i \in E$  is a list of statements exercised during the running of test case  $t_i \in T$ .

### 3. Program Slice

We consider 3 different kinds of changes [17] in  $P'$ : define change, use change and control change. Informally, a define change is a change on the left-hand side of an assignment statement; a use change is a change on the right-hand side of an assignment. Given a statement  $s_i$  in  $P$ :  $var_x = var_z$  is changed in  $P'$  to  $var_y = var_z$ , the union

set of the following 3 slices is used: forward slice at  $\langle s_i, var_x \rangle$ , forward slice at  $\langle s_i, var_y \rangle$  and backward slice at  $\langle s_i, var_z \rangle$ . In the case of use change, if a statement  $s_i, var_x=var_y$ , in  $P$  is changed to  $var_x=var_z$  in  $P'$ , then the union set of the forward slice at  $\langle s_i, var_x \rangle$  and backward slice at  $\langle s_i, var_z \rangle$  is used. Control changes require special treatment. Control change is defined as follows: a change is a control change if it affects the original control-flow [17]. Consider the program in Figure 1, if we change statement  $s_2$  from  $if(m < n)$  to  $if(m \leq n)$ , then this change is a control change. We compute the slice of control change in 3 steps: first, we compute all the control dependent of changed statement  $s_2$ , the result will be  $s_3$  and  $s_4$ . In the second step, we compute and combine forward slices over each statement computed in previous step: forward slice at  $\langle s_3, a \rangle$  and forward slice at  $\langle s_4, a \rangle$ . Thirdly, we add in the modified statement  $s_2$ . The result will be  $\langle s_2, s_3, s_4, s_5, s_6, s_7, s_8 \rangle$ . Please note that if we directly use forward slice for  $\langle s_2, (m, n) \rangle$ , the result will be  $\langle s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10}, s_{11}, s_{12}, s_{13}, s_{14} \rangle$ . This is an imprecise result because statements from  $s_9$  to  $s_{14}$  are irrelevant to the modified statement  $s_2$ .

#### 4. Slice Filtering

The execution traces are recorded in statement level during previous phase of testing. But they need to be preprocessed before feed to the clustering step. The preprocessing is carried out using the program slice computed previously. We refer this phase as slice filtering. The quality of slice filtering can be improved in 3 steps.

Firstly, many statements in an execution trace are not relevant to the program modification and including them in the execution traces will not achieve any gain. Those irrelevant statements should be filtered out from the execution traces. Let  $E = \{e_1, e_2, \dots\}$  be the set of execution traces before filtering.  $e_i$  is an  $n1$ -dimensional vector  $\langle a_{i1}, a_{i2}, \dots, a_{in1} \rangle$ , wherein  $a_{ik}$  is the execution count of statement  $s_k$  and  $\{s_k \mid 1 < k < n1\}$  is the set of all the statements of the program. If a statement  $s_k$  is not executed in execution trace  $e_i$ , the associated execution count  $a_{ik}$  is 0. So after this step of filtering, the resulting execution traces  $E' = \{e_1', e_2', \dots\}$  where  $e_i'$  is an  $n2$ -dimensional vector  $\langle a_{i1}, a_{i2}, \dots, a_{in2} \rangle$ , where  $a_{ik}$  is the execution count of  $s_k$  and  $s_k \in S$ .

Secondly, if an execution trace does not have any common statements with the program slice, we can assert that its corresponding test case is not fault-revealing. These test cases can be safely removed from the original test suite. We then generalize this property: if the size of the intersection set of an execution trace with the program slice is below a certain threshold, its corresponding test cases are not likely to be fault-revealing on one condition that the intersection set does not include any modified or deleted statements. So we remove those test cases that are remotely relevant to the code modification if they do not exercise the modified or deleted code. We intersect the program slice  $S$  computed previously with  $E'$  to form the feature set  $F' = \{f_1', f_2', \dots\}$ , where  $f_i'$  is the execution count of statement  $s_i \in S$ . For a test case  $t_i$ , the size of its feature is the number of statements it exercised in  $e_i$ . A small-sized feature indicates low level of relevance to the modification of  $P'$ . Since we are interested in test cases that have certain degrees of relevance, a test case is kept if: 1). its feature size is above a predefined threshold; 2) it exercises at least one modified or deleted statement. We use Filtering Rate,  $FR$  for short hereafter, to quantify the extension of filtering. If the threshold is  $M$  and the size of the program slice  $S_i$  is  $N$ , then the filtering rate  $FR_i$  is  $M/N * 100\%$ . When  $FR$  gets lower, the effect of filtering diminishes i.e. fewer features can be eliminated. The remaining features are then feed to the next step of filtering.

Since statements in a basic-block get executed together or not executed at all, only one of the statements in a basic-block can provide enough information. So statements of a basic-block should be reduced to a single statement. We use its first statement to represent this basic-block. So after this step of filtering, the resulting feature set  $F = \{f_1, f_2, \dots\}$ ,  $f_i = \langle a_{i1}, a_{i2}, \dots, a_{in3} \rangle$  is an  $n3$ -dimensional vector, where  $a_{ik}$  is the execution count of a basic block  $b_k$ . This is the final data and will be feed to the cluster analyzing step.  $n1 > n2 > n3$ , such that the dimension of cluster analysis is reduced.

Note that the final feature set  $F$  includes the features of all the modification traversing test cases. This is because we keep the features of all the test cases that exercise at least one modified or deleted statement during the second step of filtering. Given some assumptions [13], the set of modification traversing test cases is a superset of the fault-revealing test cases. So no omissions of fault-revealing test cases happen before clustering analysis.

#### 5. Clustering Analysis

In cluster analysis phase, a feature  $f_i$  is represented as an  $n$ -dimensional vector  $\langle a_{i1}, a_{i2}, \dots, a_{in} \rangle$ , where  $n$  is the number of basic-blocks in  $P$  and if the basic-block  $b_k$  does not get executed in execution trace  $e_i$ , then  $a_{ik} = 0$ . Euclidean distance is used as the distance function between to features  $f_i$  and  $f_j$  as following:

$$D(f_i, f_j) = \sqrt{\sum_{k=1}^m (a_{ik} - a_{jk})^2} \quad (1)$$

The simple  $k$ -means algorithm is implemented and employed as the cluster algorithm in our approach. The  $k$ -means algorithm requires the initial cluster number as a parameter. In our approach, this number is set according to the size of the final feature set. Let  $CN$  denote this initial cluster number, then  $CN = |F| * p$ , where  $|F|$  is the size of the final feature set and  $0 < p < 1$ .

#### 6. Sampling

After the cluster analysis, similar test cases will be partitioned into the same clusters. According to previous studies [5,6,10,23], if we use appropriate metrics to describe each pair of execution traces, then executions with unusual behaviors, etc. revealing a bug, tend to be isolated within the same clusters after the clustering process. And the next step is to sample out a subset of test cases which mostly consists of failed test cases.

We choose a popular strategy, namely adaptive sampling strategy [5,6], in our approach. We first sample a certain number of test cases with a pre-defined sampling rate, denoted as  $SR$  hereinafter. Then the test cases with those selected features are run to check if they are fault-revealing. If a fault-revealing test case is found, the entire cluster from which the test cases are sampled is selected. This strategy favors small clusters and has high probability to select fault-revealing test cases.

### III. EXPERIMENT AND EVALUATION

#### 1. Subject Program

We use a subject program, *space*, from *SIR*, Software-artifact Infrastructure Repository [15] as our subject program. *Space* has 5902 lines of code, 1533 basic-blocks. A total of 38 modified versions along with a base program. For each version, the base program is augmented with a real fault. A total of 13585 test cases are supplied.

## 2. Measures

We construct a measurement model to evaluate the effectiveness of a regression test selection approach. Our model consists of 3 measurements: precision, reduction, recall [23].

Precision measures the extent to which a selection method omits non-fault-revealing test cases in a run. We define precision as follows: if in a certain run the techniques selects a subset of  $N$  test cases, in which  $M$  test cases are fault-revealing. The precision of the technique is:  $M / N * 100\%$ . If a technique achieves higher precision, it means that this technique can omit more non-fault-revealing test cases.

Reduction measures the extent to which a technique can reduce the size of the original test suite. We define reduction as follows: if a selection technique selects  $M$  test cases out of all  $N$  existing test cases in a certain run, the reduction of the technique is:  $M / N * 100\%$ . A low value of reduction indicate that it reduce more cost of regression testing.

Recall measures the extent to which a selection technique can include fault-revealing test cases. We define recall as this: if a selection technique selects  $M$  fault-revealing test cases out of  $N$  existing fault-revealing test cases in a certain run, the recall of the technique is:  $M / N * 100\%$ . Recall indicates the fault detecting capability of a technique. A safe selection technique can achieve 100% recall.

Generally, a test selection technique makes a tradeoff between these 3 measurements. Safe techniques aim to achieve 100% recall, but they may falsely include many non-fault-revealing test cases. These 3 measures are computed for each version of space in our experiment.

## 3. Experimental Results

Rothermel et al. developed a safe regression selection tool Dejavu [14]. Dejavu is known as an effective algorithm in its high precision of test selection. A comparison is made between our approach and Dejavu. Note that version 1, 2, 32 and 34 of *space* do not have any fault-revealing test cases in the original test suites, hence they are excluded from the experiment.

We make 2 comparisons in the experiment: the first is between Dejavu and our approach with  $FR=0.3$ ; the second is between 2 different filtering rates,  $FR=0.3$  and  $FR=0.5$  of our approach. Other parameters are set as following: sampling rate  $SR = 0.1$ ; the initial cluster number  $CN = |F|*0.025$ , where  $|F|$  is the size of the final feature set after filtering. The sampling procedure is repeated 40 times for each version.

Figure 2-4 respectively give comparisons of precision, reduction and recall between Dejavu and our approach with  $FR$  set to 0.3. Our approach achieves approximately the same effectiveness in version 4, 5, 6, 20, 21, 22, 23, 30, 33 and is inferior to Dejavu in version 13, 25, 26, 35, 37, 38. We achieve certain improvement in the resting 19 versions.

Analysis on version 13, 25, 26, 35, 37, 38 shows that most of the program failures detected are caused by memory access violations. Those failures cause premature termination of the execution flows. However, static program analysis, like program slicing employed in our approach, cannot predict these drastic execution flow changes. In other words, the original execution traces and the program slices cannot provide enough information to differentiate and isolate those test cases. As discussed previously, the key to our approach is isolation of fault-revealing test cases into small clusters, otherwise our approach performs ineffectively.

On the other hand, if we raise  $FR$  to 0.5, certain improvement on precision, reduction and recall can be achieved. Figure 5-7 respectively compare the precision, reduction, recall of our approach with  $FR=0.3$  and  $FR=0.5$ .

We make the following observations in the experiment: (1) for most versions, our approach has higher precision and lower (lower is better) reduction than Dejavu. It means that we can select fault-revealing test cases from the original test suite and select relatively few non-fault-revealing test cases; (2) the effectiveness of our approach depends largely on the level of isolations of fault-revealing test cases. By choosing appropriate parameters such as filtering rate, sampling rate, initial cluster number etc., we can enhance the level of isolation.

## 4. Threats to Validity in the Experiment

Our primary concern of internal threats is the code for experiment. We use a well known tool *gcov* to collect coverage profilers. We also conduct a double check on other code. For external threats, the subject program *space* may not possess sufficient representativeness. More programs and more types of faults will be studied in our future work.

## IV. CONCLUSION AND FUTURE WORK

In this paper, we present a novel regression test selection technique. We show the effectiveness of program slicing techniques in filtering execution profile. A subject program, *space*, from *SIR* is used in our experiment. We construct three measurements, including precision, recall, and reduction to evaluate the effectiveness of a regression test selection approach. With the evaluation model, we compare our approach with a successful safe regression test selection technique proposed by Rothermel et al. [13, 14]. However, the following questions have not been answered in this paper:

- Can dynamic slicing techniques improve the filtering process further?
- How to determine a good filtering rate, sampling rate and cluster number for a given program?

These questions would be studied in our future work.

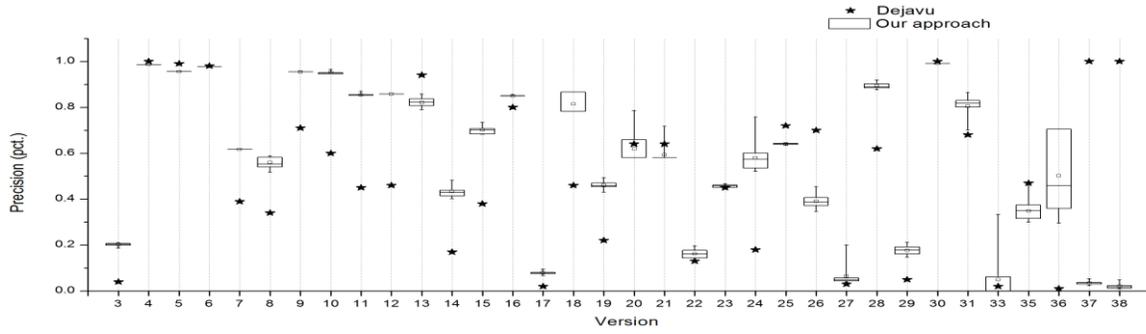


Figure 2. Comparison of precision between our approach when FR=0.3 and Dejavu

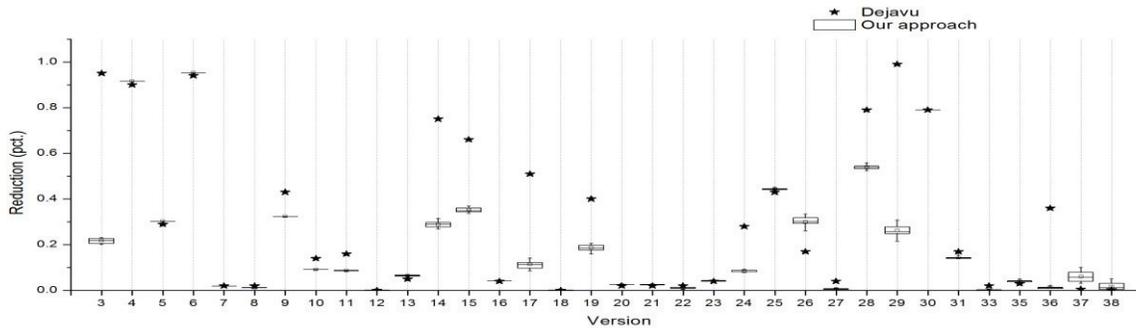


Figure 3. Comparison of reduction between our approach when FR=0.3 and Dejavu

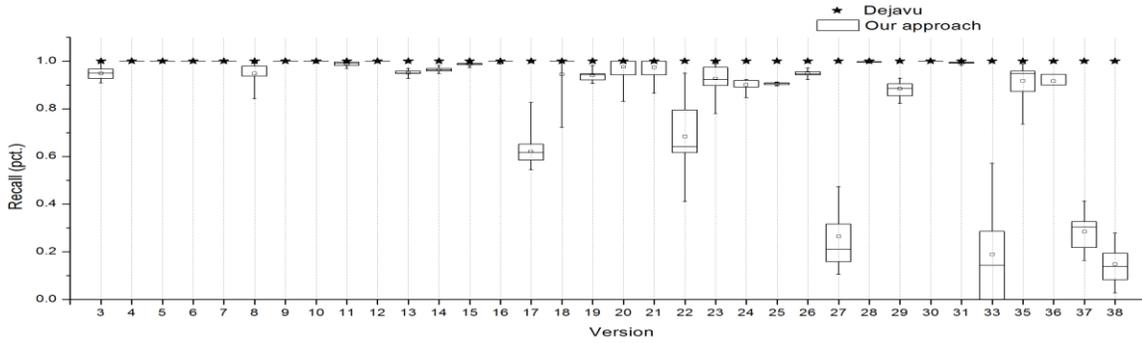


Figure 4. Comparison of recall between our approach when FR=0.3 and Dejavu

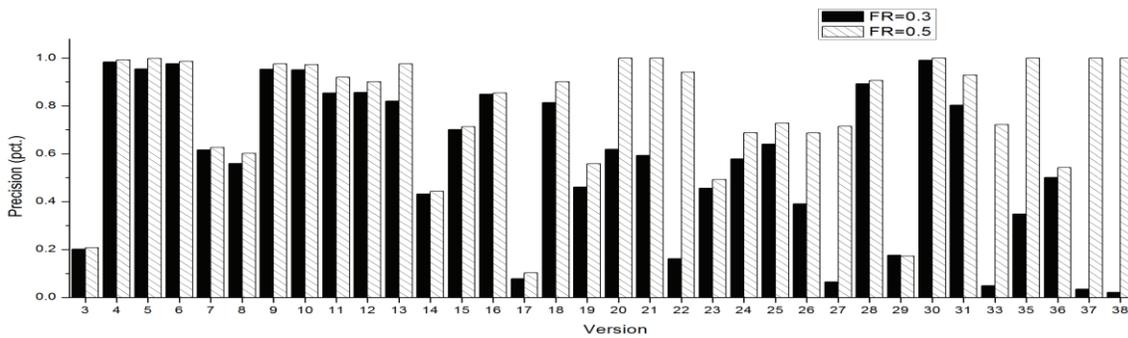


Figure 5. Comparison of precision between FR=0.3 and FR=0.5

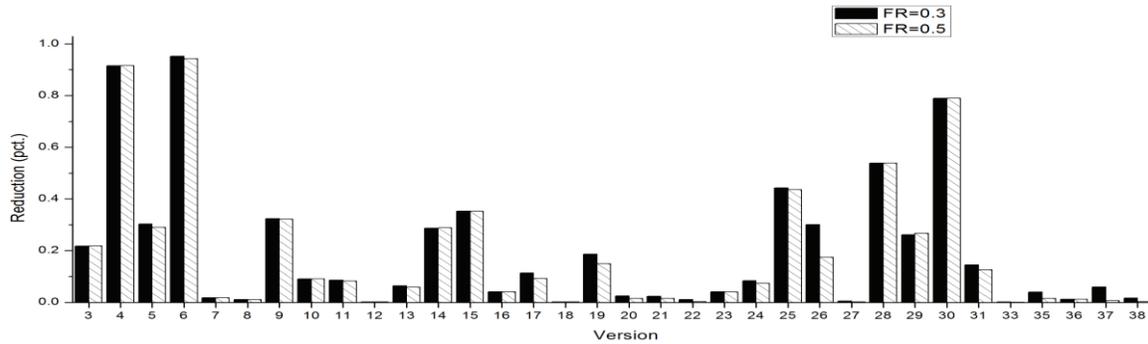


Figure 6. Comparison of reduction between FR=0.3 and FR=0.5

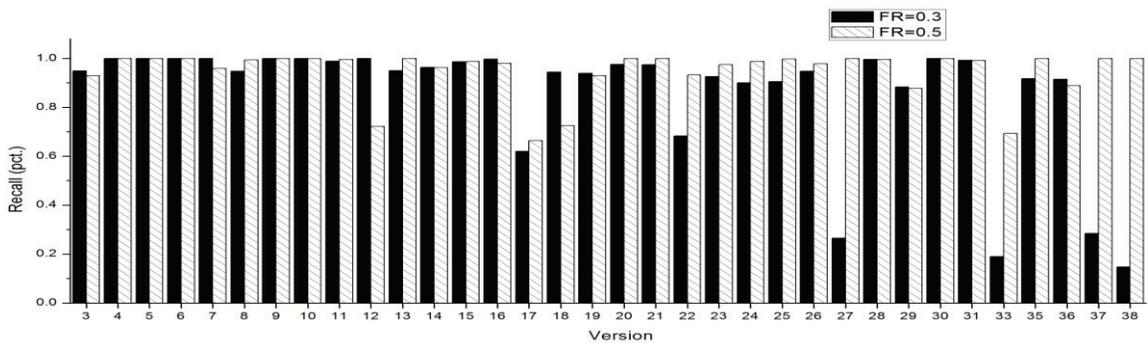


Figure 7. Comparison of recall between FR=0.3 and FR=0.5

## REFERENCES

- [1] H. Agrawal, J. Horgan, E. Krauser, and S. London, "Incremental Regression Testing," *ICSM 1993*: 348-357.
- [2] D. Binkley, K.B. Gallagher, "Program Slicing," *Advances in Computers*, 1996, 43: 1-50.
- [3] D. Binkley, "The Application of Program Slicing to Regression Testing," *Infor. Softw. Tech.*, 1998, 40(11-12): 583-594.
- [4] Y. Chen, D.S. Rosenblum, and K.P. Vo, "TestTube: A System for Selective Regression Testing," *ICSE 1994*: 211-220.
- [5] W. Dickinson, D. Leon, and A. Podgurski, "Finding Failures by Cluster Analysis of Execution Profiles," *ICSE 2001*: 339-348.
- [6] W. Dickinson, D. Leon, and A. Podgurski, "Pursing Failure: the distribution of program failures in aprofile space," *ESEC/SIGSOFT FSE 2001*: 246-255.
- [7] K.F. Fischer, "A Test Case Selection Method for the Validation of Software Maintenance Modifications," *COMPSAC 1977*:421-426.
- [8] C. Ji, Z.Chen, B. Xu, and Z. Zhao, "A Novel Method of Mutation Clustering Based on Domain Analysis," *SEKE 2009*: 422-425.
- [9] H.K.N. Leung, and L.J. White, "A Study of Integration Testing and Software Regression at the Integration Level," *CSE 1990*:290-300.
- [10] C. Liu, X. Zhang, J. Han: A Systematic Study of Failure Proximity. *IEEE Trans. Softw. Eng.*, 2008, 34(6): 826-84.
- [11] Gcov. The Gcov home page, <http://gcc.gnu.org/onlinedocs/gcov/Gcov.html>
- [12] G. Rothermel, and M.J. Harrold, "Analyzing Regression Test Selection Techniques." *IEEE Trans. Softw. Eng.*, 1996, 22(8): 529-551.
- [13] G. Rothermel, and M.J. Harrold, "Efficient, Effective Regression Testing Using Safe Test Selection Techniques," *Ph.D. dissertation, Clemson Univ.*, 1996.
- [14] G. Rothermel, and M.J. Harrold, "A Safe, Efficient Algorithm for Regression Test Selection," *ICSM 1993*: 358-367.
- [15] SIR, Software-artifact Infrastructure Repository, <http://sir.unl.edu/content/sir.html>
- [16] F.I. Vokolos, and P.G. Frankl, "Pythia: A Regression Test Selection Tool Based on Textual Differencing," *ENCRESS 1997*: 3-21.
- [17] Y. Wang, W. Tsai, X. Chen, S. Rayadurgram: "The Role of Program Slicing in Ripple Effect Analysis," *SEKE 1996*: 369-376
- [18] M. Weiser, "Program Slicing," *ICSE 1981*: 439-449.
- [19] S.Yan, Z.Chen, Z.Zhao, C.Zhang, and Y.Zhou, "A Dynamic Test Cluster Sampling Strategy by Leveraging Execution Spectra Information," *ICST 2010*: 147-154.
- [20] S.S. Yau, and Z. Kishimoto, "A Method for Revalidating Modified Programs in the Maintenance Phase," *COMPSAC 1987*: 272-27.
- [21] V. Vangala, and J. Czerwonka, P. Talluri. "Test Case Comparison and Clustering Using Program Profiles and Static Execution," *ESEC/SIGSOFT FSE 2009*: 293-294.
- [22] B. Xu, J. Qian, X. Zhang, Z. Wu, and L.Chen, "A Brief Survey of Program Slicing," *ACM SIGSOFT Software Eng.* 30(2): 1-36 (2005).
- [23] C. Zhang, Z. Chen, Z. Zhao, S. Yan and J. Zhang, "An Improved Regression Test Selection Technique by Clustering Execution Profiles," *To appear in QSIC 2010*.

# A Constrained Particle Swarm Optimization Approach for Test Case Selection

Luciano S. de Souza, Ricardo B. C. Prudêncio, Flavia de A. Barros  
Center of Informatics (CIn), Federal University of Pernambuco  
CEP 50732-970 - Recife (PE) - Brazil  
{lss2, rbc, fab}@cin.ufpe.br

## Abstract

*Automatic Test Case selection is an important task to improve the efficiency of Software Testing. This task is commonly treated as an optimization problem, whose aim is to find a subset of test cases (TC) which maximizes the coverage of the software requirements. In this work, we propose the use of Particle Swarm Optimization (PSO) to treat the problem of TC selection. PSO is a promising optimization approach which was not yet investigated for this problem. In our work, PSO was used not only to maximize coverage of requirements, but also to consider the cost (execution effort) of the selected TCs. For this, we developed a constrained PSO algorithm in which execution effort was treated as a constraint in the search, whereas the requirements coverage is used as the fitness function. We highlight that, although execution effort is an important aspect in the test process, it is generally neglected by the previous work that adopted search techniques for TC selection. In experiments performed in different test suites, PSO obtained good results when compared to other search techniques.*

## 1 Introduction

In the search for quality, Software Testing has become a central task in the software development process, aiming to assure quality and reliability [1]. However, this task can be very expensive, taking up to 50% of the total software development cost [2]. Several tools to automate testing tasks have been proposed, from test generation to its execution.

Existing tools for the generation of test cases can deliver good test suites concerning requirements' coverage (e.g., [3, 4]). However, these suites are usually too large to be executed within the available time [5]. Yet, an analysis of these suites reveals redundancies in the test cases, i.e., two or more test cases satisfying the same requirement. Hence, it is desirable to reduce the test suite, in order to respect time constraints, however without compromising coverage.

In this scenario, this work investigated mechanisms to

select an adequate subset of a given test suite considering time constraints but safeguarding requirements' coverage. Test Case (TC) selection is known to be a difficult task, and should not be performed at random. In the absence of automatic tools, this task is usually manually performed in an *ad-hoc* fashion. Besides being a slow process, manual selection does not always observe requirements' coverage [6].

Different authors have formulated the TC selection problem as an optimization problem, and tried to solve it by deploying search techniques (e.g., Greedy Search and Genetic Algorithms) [5]. Our work investigated the use of Particle Swarm Optimization (PSO) [7] for TC selection. PSO is a population-based search algorithm inspired in the social behavior of bird flocks. An increasing attention has been given to PSO in recent years, motivated by its success in different problems when compared to Genetic Algorithms and other search techniques [8]. Despite that, PSO was not yet investigated in the context of TC Selection.

Regarding selection criteria, the majority of the related previous works only deploy requirements' coverage; the cost of executing the TCs was rarely considered. We implemented a Binary Constrained PSO (BCPSO) [9, 10] considering both the requirements' coverage (quality) and the execution effort (cost) of the selected subset of TCs. We used requirements' coverage as the fitness function to be optimized, and used execution effort as a threshold constraint in the search. We also propose a hybrid PSO algorithm which integrates a greedy search mechanism to refine the solutions provided by BCPSO.

The proposed algorithms were evaluated using two different real-world test suites of functional TCs related to the mobile devices domain. The performed experiments obtained very satisfactory results. This work is being developed with the support of the CIn-Motorola Brazil Test Center (CIn-BTC) Research Group [3], a partnership between CIn/UFPE and the Motorola Industrial Ltda (Brazil).

Section 2 briefly discusses the Test Case Selection problem. Section 3 presents the proposed approach. Section 4 brings the experiments and obtained results. Finally, section 5 presents some conclusions and future work.

## 2 Test Case Selection

According to [6], Test Case selection is the task of selecting a subset of TCs from a test suite in order to achieve a given goal (e.g., maximum test suite size, minimum desired coverage etc). TC selection is commonly treated as an optimization problem by the literature [5]. In this context, a search technique explores the space of possible subsets of test cases aiming to find the subset which best matches the problem’s goal. However, the selected subset may not have the same coverage as the original test suite.

The TC selection problem is NP-Complete [5], in such a way that it is difficult to solve it in a deterministic way. Hence, heuristic search approaches are more feasible. Among the search algorithms used in this context, we can cite the HGS algorithm [11], the GRE algorithm [12], Greedy Search heuristics for set-covering [13] and Genetic Algorithms [14].

The criterion of coverage requirement to be optimized depends on the type of Software Testing. For structural test, requirements are “pieces” of program code (e.g., blocks, statements, decisions) that should be exercised by the test cases [15, 11]. For functional test, in turn, the requirements are reported in a specification of the software [6].

Although previous works have focused on optimizing the requirements coverage of the selected subset of TCs, the cost of executing these tests is also important. According to [14], the actual aim of TC selection is to achieve better efficiency in terms of cost. Nevertheless, few works were proposed in the context of test case selection considering cost [14, 16]. Yet, these works performed experiments only with structural test (although they aim to cover functional TC as well). In fact, it is harder to consider execution cost of functional tests, since it is necessary to estimate the cost of manually typing the TC before its execution [17].

The following sections presents details of our approach, which considers both coverage and cost when selecting TCs. Our experiments evaluated the proposed algorithms in two functional test suites.

## 3 PSO to Test Case Selection

Particle Swarm Optimization (PSO) [7] algorithm is a population-based search technique inspired on the birds flocks. The basic PSO algorithm initially defines a random population of *particles*, each one having a *position* in the search space and a *velocity*. The position codifies a candidate solution for the problem being solved and the velocity indicates the direction of the search performed by the particle. The particles are evaluated by a *fitness function* to be optimized. For a number of iterations, the particles fly through the search space, being influenced by each one’s own experience and by the experience of their neighbors.

Particles change position and velocity continuously, aiming to reach a better position.

PSO has shown to be a simple and efficient algorithm compared to other search techniques, including for instance the widespread Genetic Algorithms [8]. In the current work, we investigated the use of PSO to the problem of TC selection. We can point out some applications of PSO in Software Testing, particularly for test case generation (e.g., [18]). However to the best of our knowledge, there is no work investigating PSO for the TC selection problem.

In our proposal, we formulated the TC selection problem as a constrained optimization task in which requirements’ coverage is the fitness function to be optimized, and the execution effort of the selected TCs is used as a constraint in the search. As said before, previous works that deployed search techniques to TC selection rarely considered the execution cost of the selected TCs. Hence, the investigation of a constrained TC selection problem, and the use of constrained PSO to solve it are also specific contributions of our work.

We developed a Binary Constrained PSO (BCPSO) by merging two versions of PSO: (1) the binary version of PSO proposed in [9], since the TC selection problem under consideration has a binary search space; and (2) the PSO version which deals with constrained problems, proposed in [10]. The BCPSO is presented in section 3.1.

Furthermore, a hybrid implementation of BCPSO with a greedy-search algorithm, named as Forward Selection (FS), was developed aiming to verify whether some improvement in PSO performance could be obtained by using a local search mechanism for each particle (see section 3.2).

As it will be seen in section 4, the proposed algorithms were applied to TC Selection of functional test. We highlight that no previous work was found in the literature that performed cost-constrained TC Selection in the context of functional software test.

### 3.1 The Binary Constrained PSO

In this section, we provide a description of the BCPSO algorithm, developed to select test cases considering both coverage of software requirements and execution effort.

#### 3.1.1 Problem Formulation and Fitness Function

A particle position indicates a candidate subset of TCs to be used in the software testing process. Given a test suite  $T = \{T_1, \dots, T_n\}$  of  $n$  test cases, a particle position is represented as a binary vector  $\mathbf{t} = (t_1, \dots, t_n)$ , in which  $t_j \in \{0, 1\}$  indicates the presence (1) or absence (0) of the test case  $T_j$  among the subset of selected TCs.

The fitness of each particle is the percentage of requirements covered by the represented solution. Formally, let

$R = \{R_1, \dots, R_k\}$  be a given set of  $k$  requirements. Let  $F(T_j)$  be a function that returns the subset of requirements in  $R$  covered by the individual test case  $T_j$ . Then, the fitness function of a particle represented by  $\mathbf{t}$  is given as:

$$Fitness(\mathbf{t}) = 100 * \frac{|\bigcup_{t_j=1} \{F(T_j)\}|}{k} \quad (1)$$

In eq. (1),  $\bigcup_{t_j=1} \{F(T_j)\}$  is the union of requirements subsets covered by the selected test cases (i.e.,  $T_j$  for which  $t_j = 1$ ).

As said, the *execution effort* of the selected TCs is used as a constraint in the search process. Formally, each test case  $T_j \in T$  has a *cost score*  $c_j$ . The total cost of a solution  $\mathbf{t}$  is then defined as:

$$Cost(\mathbf{t}) = \sum_{t_j=1} c_j \quad (2)$$

In our work, the cost  $c_j$  was computed for each test case by using the execution effort estimation model developed by Aranha and Borba [17]. Finally, we formulated the optimization problem solved by PSO as:

$$maximize : Fitness(\mathbf{t}) \quad (3)$$

$$subject\ to : Cost(\mathbf{t}) < \theta \quad (4)$$

In the above equation,  $\theta$  is a threshold given by the user, which reflects the problem constraint: the amount of resources available to perform the Software Testing. In the constrained PSO proposed in [9], if a particle violates the constraint (i.e., if it is an infeasible solution), its fitness is penalized. The penalization of the fitness function was defined in our work as follows:

$$Fitness_{penalty}(\mathbf{t}) = Fitness(\mathbf{t}) - 100 \quad (5)$$

For particles violating the problem constraint, the fitness values initially computed using eq. (1) are replaced by the values computed by the penalized function in eq. (5), assuming non-positive values. This way, the infeasible particles do not influence the other particles of the population.

### 3.1.2 Search Operation

In PSO, each particle explores the search space by updating its position according to a *velocity vector*  $\mathbf{v} = (v_1, \dots, v_n)$  which indicates the direction of the search performed by the particle. The velocity vector is updated at each PSO iteration by the following equation:

$$\mathbf{v} \leftarrow \omega \mathbf{v} + C_1 r_1 (\hat{\mathbf{t}} - \mathbf{t}) + C_2 r_2 (\hat{\mathbf{g}} - \mathbf{t}) \quad (6)$$

In eq. (6),  $\hat{\mathbf{t}}$  indicates the best position achieved by the particle, and  $\hat{\mathbf{g}}$  is the best position achieved by its neighbors.

$r_1$  and  $r_2$  are random values in the interval  $[0,1]$ . The first part of eq. (6) represents the inertia factor, the second one represents the *cognitive* component of the search (own experience) and the third one represents the *social* component of the search (experience from the neighborhood).

The parameters  $\omega$ ,  $C_1$  and  $C_2$  control the trade-off between the cognitive and the social behavior of the particles. In our work,  $\omega$  linearly decreases from 0.9 to 0.4, and  $C_1 = C_2 = 2$  (as suggested by [19]). Finally, in order to define neighborhood, each particle is indexed, and its neighborhood consists of the particles which are its predecessor and successor.

In PSO, the particle position is updated in the direction of its velocity. In BCPSO, the update of positions followed the same operations originally proposed in the binary PSO [9]. First, the sigmoid function is used to normalize the values of velocity in the interval  $[0,1]$  as follows:

$$sig(v_j) = \frac{1}{1 + e^{-v_j}} \quad (7)$$

Finally, the new particle position is updated as follows:

$$t_j = \begin{cases} 1, & \text{if } r_j \leq sig(v_j) \\ 0, & \text{otherwise} \end{cases} \quad (8)$$

In the above equation,  $r_j$  is a random number sampled in the interval  $[0,1]$ . Equation (8) was proposed in [9] in order to certify that the new positions are still binary vectors. The position value  $t_j$  tends to 1 when the velocity assumes higher values (closer to 1). In its turn,  $t_j$  tends to 0 for lower values of velocity ( $v_j$  close to 0).

## 3.2 The Hybrid BCPSO-FS

A greedy search algorithm, named as Forward Selection (FS) inspired on [20], was adapted to consider the execution effort constraint. The search process starts with an empty set of TCs, and progressively selects a TC that, when added to the TCs set, does not violate the cost constraint. At each step, the algorithm chooses the non-selected test case  $T_j \in T$  with highest requirements coverage ( $F(T_j)$ ), and verifies whether the cost constraint is not violated. Whenever a tie is observed between the best remaining TCs, a random choice is made. This process is repeated until all remaining TCs will cause a violation of the cost constraint when individually added to the TCs subset being formed.

The FS algorithm is combined to the BCPSO in order to create a new hybrid algorithm (named here as BCPSO-FS). The hybrid algorithm is similar to BPSO, however, at each iteration of BCPSO, the FS algorithm is used as a local search mechanism in order to refine each particle's solution. Each particle produced in a BCPSO iteration is given as an initial solution to FS, which refines it until the stopping criterion of FS is reached. The solutions optimized by FS are

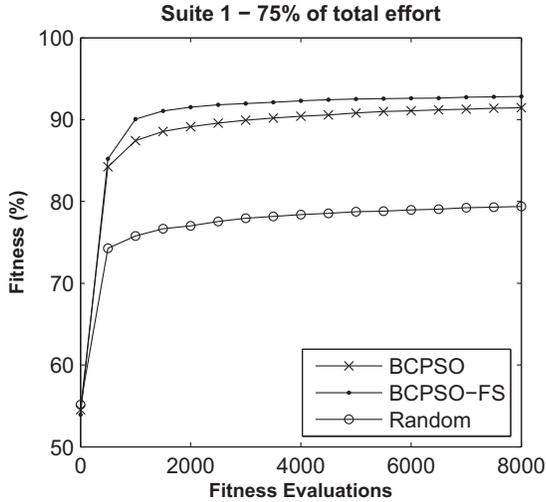


Figure 1. Search progress on fitness observed for Suite 1 and  $\theta=75\%$

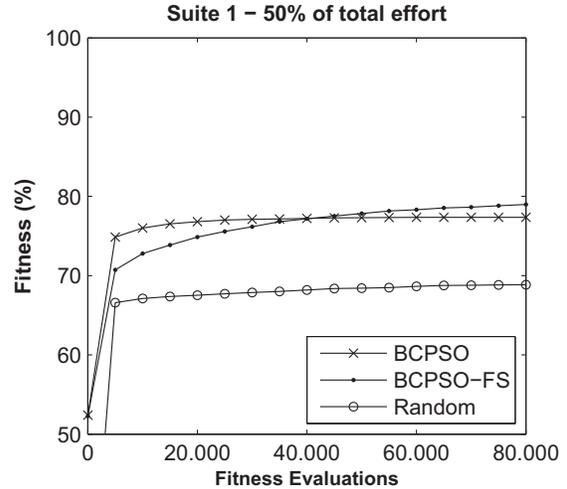


Figure 2. Search progress on fitness observed for Suite 1 and  $\theta=50\%$

then used as the particles population of the next iteration of BCPSO. The hybrid algorithm alternates the use of BCPSO and FS during the search: BCPSO performs a global exploration of the search space, whereas FS refines the solutions provided by BCPSO thus performing a local search.

Hybrid algorithms that combine global and local procedures have shown to be useful in other applications (e.g., [21]). We expect that the hybrid algorithm proposed in our work can also be useful for TC selection.

## 4 Experiments and Results

In this section, we present the experiments which evaluated the performance of the implemented search techniques. The experiments were performed to select TCs from two test suites used to test mobile devices<sup>1</sup>, referred here as Suite 1 and Suite 2. Both test suites have 80 TCs, however their characteristics are different, as shown in the table 1. Suite 1 is more complex, covering a higher number of requirements. The total effort needed to execute all test cases in Suite 1 is also higher when compared to Suite 2.

The TCs in Suite 1 are less redundant<sup>2</sup>, i.e. two distinct test cases rarely cover the same requirement. Hence, for Suite 1, it is expected to be more difficult to find a solution with good fitness evaluation. In Suite 2, in turn, each test case individually covers a higher number of requirements,

<sup>1</sup>These suites were obtained from the Motorola CIn-BTC (Brazil Test Center) research project

<sup>2</sup>Redundancy of a suite is measured by the average Jaccard similarity between the sets of requirements covered by each pair of TCs.

Table 1. Characteristics of the Test Suites

	Suite 1	Suite 2
Total Effort	1053.91 min	699.31 min
n. of Requirements	410	248
Redundancy	0.36%	14.09%
n. of Test Cases	80	80

with a higher level of redundancy. This characteristic makes it easier to select test cases in Suite 2.

In our experiments, we evaluated the search progress of BCPSO and BCPSO-FS for each suite in two different scenarios regarding the threshold ( $\theta$ ). In these scenarios, the  $\theta$  threshold was defined to reduce the execution effort to 75% and 50% of the total effort estimated for each suite. The search stopping criterion was set to the maximum number of 200,000 fitness evaluations (which showed to be big enough to allow the convergence of each tested algorithm).

As a basis of comparison, we also performed experiments using a *purely random* search, which, despite its simplicity, has the advantage of performing a uniform exploration of the search space, being very competitive in other contexts [21]. Each algorithm was executed 100 times for each suite and scenario considered.

Figures 1, 2, 3 and 4 present the average fitness (over the 100 runs) along the number of fitness evaluations. The proposed algorithms outperformed the Random search in all suites and scenarios, which indicates that the proposed algorithms were feasible to optimize the fitness function. By comparing BCPSO to BCPSO-FS, we observed that BCPSO obtained in general a better behavior in the begin-

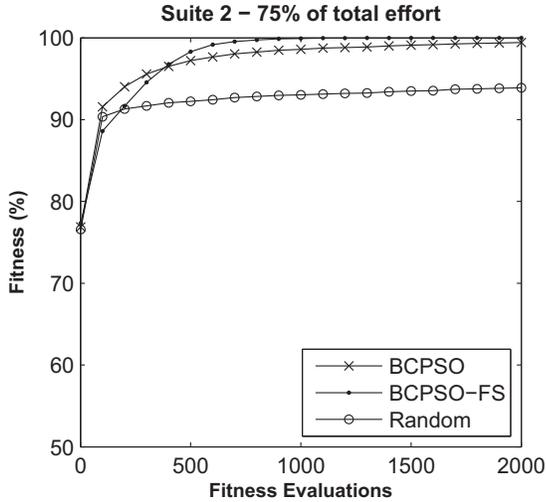


Figure 3. Search progress on fitness observed for Suite 2 and  $\theta=75\%$

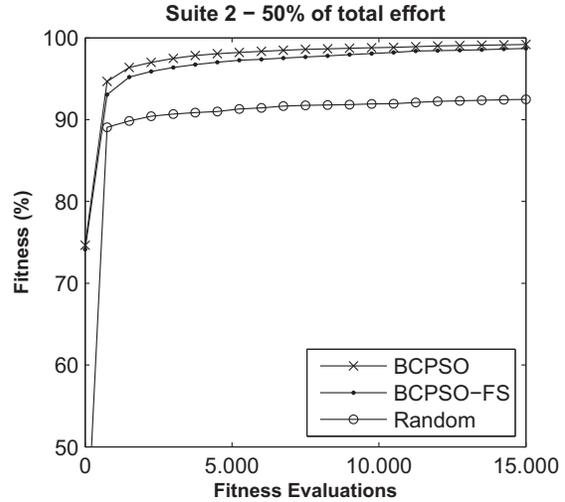


Figure 4. Search progress on fitness observed for Suite 2 and  $\theta=50\%$

ning of the search. In the initial iterations, a global exploration of the search space performed by BCPSO was more important than the refined search performed by BCPSO-FS. The BCPSO-FS, in turn, obtained better results when used to refine relatively good regions of the search space. In these regions, the FS local search could achieve positions which were difficult to be yielded by BCPSO.

As expected, better results were observed for Suite 2 (the less redundant suite) for which the search algorithm converged to adequate solutions within a lower number of fitness evaluations. We could also notice that the constraint imposed in the search had an impact in the results. For  $\theta=75\%$ , the search algorithms obtained in general better results, since this constraint is not so strong. In fact, the higher is the value of  $\theta$ , the higher is the number of subsets of TCs that do not violate the execution effort constraint (i.e. the higher is the number of feasible solutions). These results indicate that the performance obtained by the search algorithms would be dependent both on the characteristics of the test suite given as input, and on the resources constraints imposed by the context of application.

Finally, we compared the best solutions found by each algorithm BCPSO, BCPSO-FS, FS and Random (see table 2). Our results showed that the hybrid BCPSO-FS outperformed all other algorithms in all scenarios for Suite 1 (the most complex one). For Suite 2, all algorithms (apart from Random) achieved coverage values of 100%. Despite its simplicity, the FS algorithm has shown to be very competitive, particularly when compared to the BCPSO. The hybridization showed to improve the BCPSO algorithm by adding an explicit mechanism of local search. Other hy-

Table 2. Best average fitness results

Algorithm	Suite 1		Suite 2	
	$\theta=75\%$	$\theta=50\%$	$\theta=75\%$	$\theta=50\%$
BCPSO	93.50	77.39	100.00	100.00
BCPSO-FS	95.26	80.19	100.00	100.00
FS	95.00	78.43	100.00	100.00
Random	82.74	69.43	97.02	94.43

brid search strategies can be investigated in future work, for instance, by adopting other greedy search techniques.

## 5 Conclusion

In this work, we investigated the use of Particle Swarm Optimization for TC selection. We can point out some contributions of the current work. First, to the best of our knowledge, PSO was not investigated yet in the context of TC selection. Also, we considered the effort in executing the selected test cases by formulating TC selection as a constrained optimization task and by proposing specific versions of PSO to treat this task.

We implemented a Binary Constrained PSO (BCPSO) and a hybrid algorithm BCPSO-FS, which integrated the FS algorithm to improve the performance of the BCPSO. In the performed experiments, BCPSO and BCPSO-FS outperformed the results obtained by a Random search, thus verifying the viability of using PSO to TC selection. The BCPSO-FS obtained good results when compared to its individual components. The good performance of BCPSO-FS

was achieved by combining the global search performed by BCPSO and the local search performed by the FS algorithm.

Several extensions of the current work can be considered in the future. First, we intend to perform experiments on more test suites, and investigate the effect of the PSO's parameters in its performance. Following, we intend to adapt a higher number of algorithms for dealing with constrained TC selection and to perform a more complete comparison of techniques. Finally, we will investigate new strategies to combine search techniques, in order to provide more robust hybrid algorithms for TC selection.

*Acknowledgments:* The authors would like to thank CNPq, CAPES (Brazilian Agencies) and National Institute of Science and Technology for Software Engineering (INES) for their financial support.

## References

- [1] B. Beizer, *Software Testing Techniques*. International Thomson Computer Press, 1990.
- [2] R. Ramler and K. Wolfmaier, "Economic perspectives in test automation - balancing automated and manual testing with opportunity cost," in *Workshop on Automation of Software Test, ICSE 2006*, 2006.
- [3] P. Borba, D. Torres, R. Marques, and L. Wetzel, "Target - test and requirements generation tool," in *Motorola's 2007 Innovation Conference (IC'2007)*, 2007.
- [4] Y. Kissoum and Z. Sahnoun, "A formal approach for functional and structural test case generation in multi-agent systems," in *IEEE/ACS Intern. Conf. on Computer Systems and Applications*, 2007, pp. 76–83.
- [5] J.-W. Lin and C.-Y. Huang, "Analysis of test suite reduction with enhanced tie-breaking techniques," *Inf. Softw. Technol.*, vol. 51, no. 4, pp. 679–690, 2009.
- [6] E. Cartaxo, P. Machado, and F. Oliveira Neto, "On the use of a similarity function for test case selection in the context of model-based testing," *Software Testing, Verification and Reliability*, pp. 270–285, 2009.
- [7] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE Intern. Joint Conf. on Neural Networks*, 1995, pp. 1942–1948.
- [8] R. C. Eberhart and Y. Shi, "Comparison between genetic algorithms and particle swarm optimization," *LNCS*, vol. 1447, pp. 611–616, 1998.
- [9] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics*, 1997, pp. 4104–4109.
- [10] X. Hu and R. Eberhart, "Solving constrained nonlinear optimization problems with particle swarm optimization," in *6th World Multiconference on Systemics, Cybernetics and Informatics*, 2002, pp. 203–206.
- [11] M. J. Harold, R. Gupta, and M. L. Soffa, "A methodology for controlling the size of a test suite," *ACM Trans. Softw. Eng. Methodol.*, vol. 2, no. 3, pp. 270–285, 1993.
- [12] T. Y. Chen and M. F. Lau, "A new heuristic for test suite reduction," *Information & Software Technology*, vol. 40, no. 5-6, pp. 347–354, 1998.
- [13] V. Chvatal, "A greedy heuristic for the set covering problem," *Mathematics of Operations Research*, vol. 4, no. 3, pp. 233–235, 1979.
- [14] S. Yoo and M. Harman, "Pareto efficient multi-objective test case selection," in *Proceedings of the 2007 International Symposium on Software Testing and Analysis*, 2007, pp. 140–150.
- [15] L. M. G. Feijs, N. Goga, S. Mauw, and J. Tretmans, "Test selection, trace distance and heuristics," in *Proceedings of the IFIP 14th International Conference on Testing Communicating Systems*, 2002, pp. 267–282.
- [16] A. G. Malishevsky, J. R. Ruthruff, G. Rothermel, and S. Elbaum, "Cost-cognizant test case prioritization," Department of Computer Science and Engineering, University of Nebraska-Lincoln, Tech. Rep., 2006.
- [17] E. Aranha and P. Borba, "An estimation model for test execution effort," in *Proceedings of the 1st International Symposium on Empirical Software Engineering and Measurement*, 2007, pp. 107–116.
- [18] A. Windisch, S. Wappler, and J. Wegener, "Applying particle swarm optimization to software testing," in *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, 2007, pp. 1121–1128.
- [19] Y. Shi and R. Eberhart, "Parameter selection in particle swarm optimization," in *Proc. of the Intern. Conf. on Evolutionary Programming*, 1998, pp. 591–600.
- [20] R. Kohavi and G. John, "Wrappers for feature subset selection," *Artificial Intelligence*, vol. 97, no. 1-2, pp. 273–324, 1997.
- [21] M. Takaki, D. Cavalcanti, R. Gheyi, J. Iyoda, M. d'Amorim, and R. B. Prudêncio, "A comparative study of randomized constraint solvers for random-symbolic testing," in *Proceedings of the Nasa Formal Methods Symposium*, 2008, pp. 56–65.

# Software Defect Estimation using Support Vector Regression

Roberta A. A. Fagundes

Universidade Federal de Pernambuco

Centro de Informática

Av. Prof. Luiz Freire, s/n - Cidade Universitária

CEP 50740-540 - Recife (PE) - Brazil

raaf@cin.ufpe.br

Renata M.C.R. de Souza

Universidade Federal de Pernambuco

Centro de Informática

Av. Prof. Luiz Freire, s/n - Cidade Universitária

CEP 50740-540 - Recife (PE) - Brazil

rmcrs@cin.ufpe.br

## Abstract

*By using a kernel function, data that are not easily separable in the original space can be predicted data. This paper evaluates the performance of support vector regression for software defect estimation. In addition, this method is compared with the two statistical methods: kernel and linear regression. The performance of the methods is assessed by the mean magnitude of relative errors (MMRE). Experiments were carried out using a project data set from NASA and the results show that support vector regression methods gives better performance than regression statistical methods in this task.*

## 1. Introduction

Software development and maintenance constitutes a significant cost for organizations, and the manner of high quality software requires extensive testing for software defects [1]. Quality of a software system is relative to the number of defects reported in the final product. Early discovery of software errors is very important and may cause significant cost savings, especially for large and complex systems.

During the past years, many researchers have attempted to evaluate different methods and several defect prediction systems have been proposed. The results of these systems are given in terms of classification accuracy, precision, performance, etc. However, these factors do not really show the goodness of the model. Examples of software defect estimation using global learning methods include the regression method [3], classification [7] and examples of software effort estimation contain the neural networks [2][4].

Machine learning techniques use data from past projects to build a regression model that is subsequently employed to predict the defect of novel projects [3]. The linear regression [8] is widely used in statistical estimation. The benefits

of a linear model are its simplicity and ease of use, while its major drawback is its high model bias: if the underlying function is not well approximated by a linear function, then linear regression produces poor results. Support vector regression is a kernel method for regression based on the principle of structural risk minimization [5]. This method exploits the fact that, over a small enough subset of the domain, any sufficiently nice function can be well approximated by a linear function. Kernel methods have outperformed more traditional techniques in a number problems, including regression [3][4].

This paper discusses regression models over a statistical view of learning in an application with software defect estimation. A NASA software project data base [6] is considered. This dataset is related to one of the NASA projects and contains several modules. All discovered faults of the system are also registered in each dataset, together with the number of module containing the fault.

The idea is to present the usefulness of the SVR method as one alternative to the problem of software engineering estimation models, particularly defect estimation. In addition, the SVR method is compared with two regression statistical methods. The performance of the methods is measured by the prediction accuracy that is assessed based on the mean magnitude of relative errors (MMRE).

The structure of this paper is as follows: Section 2 presents regression methods. Section 3 describes the importance of estimation in software engineering and shows the NASA data base. Section 4 presents an experimental evaluation of the methods using the NASA data set. Finally, Section 5 gives the concluding remarks.

## 2. Kernel-Based Regression Methods

This section presents two regression methods. These methods construct a local approximation of a linear function that model the relation between a set predict variables

and a response variable at the query point. The goal of a regression method is to find the function  $f(x)$  that best models the training data. In the linear regression, this is done by finding the line that minimizes the sum of squares error on the training set.

Let  $\Omega = (x_i), y_i (i = 1, \dots, n)$  be a training data set. Each pattern  $i$  is described by a vector  $x_i = (x_{i1}, \dots, x_{ip})$  representing quantitative values of  $p$  predictor variables  $(X_1, \dots, X_p)$  and a quantitative response value  $y_i$  representing the value of a dependent quantitative variable  $Y$ .

In our case, we aim to build regression models with a training data set and to use these models to predict software project defects

### 2.1. Support Vector Regression (SVR)

Support vector machines (SVMs) are a set of related learning methods for classification and regression problems introduced by Vapnik and they have attracted much research attention in recent years due its demonstrated improved generalization performance over other techniques in many real world applications [4]. The main difference between SVM and these techniques including neural network is that it minimizes the structural risk instead of the empirical risk. The principal is based on the fact that minimizing an upper bound on the generalization error rather than minimizing the training error is expected to perform better.

*Support vector* regressions (SVRs) are state-of-art learning machines for solving linear and non-linear regression problems. They are able to construct spline approximations of given data independently from the number of input-dimensions regarding complexity during training and with only linear complexity is compared to exponential complexity in a conventional method.

In  $\epsilon$ -SVR [5], our goal is to find the  $\epsilon$ -insensitive loss function. This function defines a band around the true outputs sometimes referred to as a tube. The idea is that errors smaller than a certain threshold  $\epsilon > 0$  are ignored. On the other hand, we do not care about errors as long as they are less than  $\epsilon$ , but will not accept any deviation larger than this.

When using SVR for linear regression, the function is given by

$$f(x) = \langle \omega, x \rangle + b \text{ with } \omega \in X, b \in \mathfrak{R} \quad (1)$$

where  $\langle \dots \rangle$  denotes the dot product in  $X$ . For the case of nonlinear regression,  $f(x) = \langle (\omega, \phi(x)) \rangle + b$ , where  $\phi$  is some nonlinear regression function witch maps the input space to a higher (maybe infinite) dimensional feature space.

The SVR algorithm involves the use of Lagrangian multipliers, which rely solely on dot products of  $\phi(x)$ . This can be accomplished via kernel functions, defined as

$$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle \quad (2)$$

Thus, the method avoids computing the transformation  $\phi(x)$  explicitly. The common kernel types are linear, polynomial and Gaussian kernels. In this paper, we consider SVR with linear kernel that is computed as  $K(x_i, x_j) = \langle x_i, x_j \rangle$ .

### 2.2. Kernel Regression (KR)

The kernel regression [8] (here called KR) is a well-established non-parametric method for nonlinear regression in which the target value for a test point is estimated using a weighted average of the surrounding training samples. The weights are typically obtained by applying a distance-based kernel function to each of the samples, which presumes the existence of a well-defined distance metric.

The essential idea of this method is that in estimating  $f(x)$  it is desirable to give greater weight to observations that are close to the focal  $x_i$ . Considering  $d(x, x_i)$  the distance between an observation  $x$  of the data set  $D$  and the query  $x_i$ . To weight the neighborhood of  $x_i$  it is need a kernel function  $K(d(x, x_i))$  that attaches greatest weight to observations that are close to the focal  $x_i$ , and then falls off symmetrically and smoothly as  $|d|$  grows. Given these characteristics, the specific choice of a kernel function is not critical.

A popular kernel function is the Gaussian function (equation 5) given by

$$K(d(x, x_i)) = \frac{1}{\sqrt{2\pi}} e^{-\frac{d(x, x_i)}{2h}} \quad (3)$$

where  $d(x, x_i)$  is the square Euclidian distance between  $x$  and the location of interest  $x_i$ . Here, the bandwidth  $h$  is the standard deviation of a normal distribution centered at  $x_i$ .

## 3. Software Defect Estimation

Software estimation is responsive to the widespread problems the software industry has experienced in creating meaningful cost and schedule estimates. The estimations are the basis of planning and particularly they are useful in schedule and budget development. It is important to estimate the size of the software, the project costs and number of defects the whole project.

Software defect is probably one of the most analyzed response variables in recent years in the process of project management. The determination of the estimated value for this variable when initiating software projects allows us to quality and resources of productivity software. The cost of fixing a defect later can be several orders of magnitude higher than during development, yet a program must

be shipped by some deadline dictated by market considerations. This makes the estimation of the number of remaining defects a very important challenge.

In order to run an application with software defect estimation using support vector regression, experiments with a well known NASA data set presented in [6] is considered in this paper. This data set contains 18 projects. Each project has several examples that are described by two predictor variables and a response variable that is Error Count (EC). Here, two projects *KC1* and *KC4* of this data set are adopted.

The project *KC1* is a large ground system and made up of 43K*SLOC* of C++ code. The predict variables considered for this project are Design Complexity (DC) that is the degree of complexity of the module and Branch count (BC) that is the number of ramification in the module.

The project *KC4* is a ground-based subscription server and consisting of 25K*LLOC* of *Perl* code. The predictor variables are: Branch count (BC) and Node Count (NC) that is number of nodes found in a given module.

#### 4. Experimental Evaluation

This section presents an performance analysis of the regression methods based on kernel functions considered in this work regarding the projects *KC1* and *KC4*. In addition, these methods are compared with the classic linear regression one. The accuracy prediction of the methods is measured by the mean magnitude of relative error (MMRE) that is estimated by the hold-out method in the framework a Monte Carlo simulation. The experiments are performed using the Language R [3].

Lower values of MMRE mean good regression models, and a common criterion for accepting a model as good is  $MMRE \leq 0.25$  [3] and [4]. The mean magnitude of relative error (MMRE) is given by

$$MMRE = \frac{1}{n} \sum_{i=1}^n \frac{|y_i - \hat{y}_i|}{y_i} \quad (4)$$

where  $y_i$  and  $\hat{y}_i$  are the original and estimated value for Error Count (EC), respectively, and  $n$  is the size (number of examples) of a project.

Initially, an analysis to measure the linear correlation between the variables for each project is carried out. The correlation  $\rho$  is given by the *Pearson's Correlation*. Small values of  $\rho$  mean low degree of relationship between two variables.

The correlation between the NC and BC variables for the project *KC4* is given by  $\rho = 0.281688$ . This value points out low degree of linear dependence between these variables. Regarding the linear correlation between a predict variable and the response variable (EC) for the project

*KC4*, we have that the correlation between NC and EC variables is  $\rho = 0.6942585$  and correlation between BC and EC variables is  $\rho = 0.6792276$ . These values indicate that there is a moderate correlation between predictor and response variables for project *KC4*.

For the project *KC1*, the correlation between the BC and DC predictor variables is  $\rho = 0.2177881$ . Again, the predictors variables are fairly correlated. Regarding the linear correlation between a predict variable (EC) and the response variable for the project *KC1*, we have that the correlation between BC and EC variables is  $\rho = 0.3743979$  and the correlation between DC and EC variables is  $\rho = 0.6928196$ . These results show that there is a moderate correlation between DC and EC variables and a fair correlation between BC and EC variables.

The regression methods were applied to the projects *KC1* and *KC4* and the MMRE for each regression method was computed from a test data set. Test and learning sets are randomly selected from each project. The learning set corresponds to 75% of the original data set and the test data set corresponds to 25%. The experiments were organized in the framework of Monte Carlo simulation and the estimated MMRE of corresponds to the average of the MMREs found between the 100 replications of the test set. Statistical Student's t-test for paired samples at a significance level of 5% is then applied to compare the methods.

Regarding the application for the SVR method was used the value  $10^{-1}$  as the cost parameter and the linear kernel. For KR method was used the value 0,75 as span parameter.

Two regression models were adopted for estimating the response variable (ER): a model with one predictor variable (simple regression) and a model with two predictor variables (multiple regression).

Table 1 presents the average and the standard deviation for the MMRE considering the Linear, KR and SVR regression methods and multiple and simple model for project *KC4*. For the simple regression model was used the NC variable as predictor because this variable is the best option in terms of the correlation coefficient  $\rho$ . From the results in this table, we can observe that the SVR model outperforms the KR and linear methods.

**Table 1. MMRE for Project *KC4***

Measure	Simple Regression		
	SVR	KR	Linear
Average	0.00920101	0.0115975	0.0118506
Standard Deviation	0.00686635	0.010012	0.00797625
	Multiple Regression		
	SVR	KR	Linear
Average	0.01360363	0.01532860	0.0151362
Desviation Standard	0.00340079	0.00815224	0.00429567

Table 2 shows the average and the standard deviation for the MMRE considering the Linear, KR and SVR regression methods and multiple and simple model for project *KC1*. For the simple regression model was used the DC variable as predictor because this variable is the best option in terms of the correlation coefficient  $\rho$ . From the results in this table, again we can observe that the SVR model is better than the KR and linear methods.

**Table 2. MMRE for Project *KC1***

Measure	Simple Regression		
	SVR	KR	Linear
Average	0.00876021	0.0177154	0.0193363
Standard Deviation	0.0011359	0.00463945	0.0027578
	Multiple Regression		
	SVR	KR	Linear
Average	0.0193927	0.0216992	0.028415
Standard Deviation	0.0193927	0.0056143	0.00412899

Table 3 and Table 4 present the comparison between the regression methods for projects *KC4* and *KC1*, respectively, based on the observed values of statistic tests. Let  $\mu^1$  and  $\mu^2$  be the average of the MMRE for SVR method and a non-SVR method, respectively. Here, the null and alternative hypotheses of a statistic test are:  $H_0 : \mu_1 = \mu_2$  and  $H_1 : \mu_1 < \mu_2$ .

**Table 3. Methods for Project *KC4***

Model	SVR $\times$ Linear	
	p-value	Decision
Simple Regression	$4.557715e - 06$	Reject $H_0$
Multiple Regression	0.0004909987	Reject $H_0$
	SVR $\times$ Kernel	
	p-value	Decision
Simple Regression	0.03787513	Reject $H_0$
Multiple Regression	0.005610107	Reject $H_0$

**Table 4. Methods for Project *KC1***

Method	SVR $\times$ Linear	
	p-value	Decision
Simple Regression	$1.977956e - 18$	Reject $H_0$
Multiple Regression	$1.977956e - 18$	Reject $H_0$
	SVR $\times$ Kernel	
	p-value	Decision
Simple Regression	$1.977956e - 18$	Reject $H_0$
Multiple Regression	$2.491372e - 15$	Reject $H_0$

From the results in these tables, we can conclude that the performance of the SVR method in terms of the MMRE is

clearly superior to those of the kernel and linear regression methods.

## 5. Conclusions

In this paper, we have investigated the use of support vector regression for estimation of software project defect. We have carried out experiments using two projects of the well-known NASA data set. The support vector regression method was compared the both kernel and linear regression methods. We have considered the estimation of defect based solely on the simple regression, as well as, multiple regression for the two projects. The prediction quality is assessed by MMRE. This measure is estimated by hold-out method in the framework of a Monte Carlo simulation with 100 replications. The results provided by the SVR method are compared with the correspondence results provided by linear and kernel regression methods regarding the two projects. Indeed, the results showed that the SVR model is the best option in terms of prediction quality.

## References

- [1] S.M. Fakhrahmad, A.Sami, Effective Estimation of Modules' Metrics in Software Defect Prediction Proceedings of the World Congress on Engineering 2009 Vol I WCE 2009, July 1 - 3, 2009, London, U.K.
- [2] Zeng H., Rine D., Estimation of Software Defects Fix Effort Using Neural Networks, Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC04) 0730-3157/04, 2004, IEEE.
- [3] Leal Q. Luciana, Fagundes A. A. Roberta, Souza M.C.R. Renata, Gusmo M.G. Cristine and Moura P.Hermano, Nearest-Neighborhood Linear Regression in an Application with Software Effort Estimation, SMC 2009.
- [4] Oliveira L.I. Adriano, Estimation of software projects effort with support vector regression. Neurocomputing, vol 69, no. 13-15, pp. 1749-1753, August, 2006
- [5] A.J. Smola, B. Scholkopf, A tutorial on support vector regression, Stat. Comput. 14 (3) (2004) 199-222.
- [6] Metric Data Program, NASA IV & V Facility, <http://mdp.ivv.nasa.gov/>
- [7] J. Dem sar, (2006). Statistical Comparisons of Classifiers over Multiple Data Sets. Machine Learning Research, 7, pp.1-30.
- [8] D.C. Montgomery, and E.A. Peck: Introduction to Linear Regression Analysis. Wiley, New York, 1982.

# Analyzing the Relationship of Process Metrics And Classified Changes - A Pilot Study

Andreas Mauczka, Mario Bernhart, Thomas Grechenig  
Research Group for Industrial Software  
Vienna University of Technology  
1040 Vienna, Austria  
{andreas.mauczka, mario.bernhart, thomas.grechenig}@inso.tuwien.ac.at

**Abstract**—Knowing how a software project is likely to evolve is an essential problem for any software project manager. Finding efficient predictors for performance indicators (e.g., bugrates) has been the focus of many studies. Previous studies found that process metrics make likely candidates for this predictor role, for bug data in particular. We propose a methodology for in-depth analysis of process metrics to find out how they relate to changes. We use a lexical approach to classify changes into perfective, adaptive and corrective changes. The analysis consists of examining a set of hypotheses on the nature of the relationship of certain process metrics and the change categories, e.g., perfective changes and their impact on the consecutive bug appearance rate of a module. As this work is in progress, we present a pilot study on a module of the Ant Project to showcase and discuss our technique and point out early trends.

**Index Terms**—Data Mining; Process Metrics; Software Repositories; Change Management Data; Maintenance;

## I. INTRODUCTION

Efficient monitoring of a software project is a fundamental need of any project manager. With mailing lists, version control and problem tracking databases huge pools of historical project data are at the disposal to depict a detailed current state of the project and possible threats. A focus of current research is on building models for fault prediction (e.g., [9] or [15]), however there is also a need for a more fine-grained analysis of change and process metrics. Mockus et al. present a lexical approach to classify changes more differentiated in [7]. We want to expand on this approach by showing how specific maintenance tasks (adaptive, corrective and perfective changes) relate to process metrics, e.g. whether a perfective change temporarily decreases the fault rate in a module.

We want to show that there are significant differences between the various process metrics and the categories of changes - e.g., that fault data based on a bug tracker and bug fixing procedures through corrective changes might not be as closely related as generally is assumed. We use the Core Module of the Ant Project<sup>1</sup>, a subproject of the Apache Project<sup>2</sup> to showcase trends that warrant further research and to discuss our technique and tooling. In future work we will formulate a more generic methodology to perform analyses on heterogeneous projects, based on the experiences made with this study. The planned analysis consists of a regression

analysis and a time series analysis similar to the analysis in [5]. The knowledge gained by the pilot study will be used as a foundation for multiple project analysis.

## II. PRESENTING THE IDEA AND THE DATA

Predicting fault data as shown by Zimmermann et al. in [15] is a helpful source to learn more about the nature of software projects. However, from a project management point of view finding or predicting bugs is only one part of improving software quality. To prevent bugs and to keep the number of bugs low is as much of a priority as knowledge about possible bugs. Performance increase and keeping the system scalable, stable and secure is important as well. These problems are not addressed by analyzing or predicting bugs alone. To analyze most of these factors, we will examine the relationship of process metrics and corrective, adaptive and perfective changes. We use the definition by Mockus in [7] - corrective changes are changes to fix faults, adaptive changes are changes that add new features and perfective changes are changes that restructure the code to accommodate future changes.

We propose the following hypotheses:

- **Hypothesis 1:** Process metrics provide meaningful results for corrective changes as it has been proven in various studies (e.g., in [15]) that process metrics relate to fault data and that this bug rate is constant (i.e. corrective changes are constant too).
- **Hypothesis 2:** Process metrics provide meaningful results for adaptive changes as some process metrics measure developer activity on a module and an active module is likely subject to further adaptive changes. Some process metrics analyze numbers of faults and changes and as has been suggested and proven in literature, modules that are prone to change are also likely to contain faults (See [10] or [9]).
- **Hypothesis 3:** Process metrics are unlikely to relate to perfective changes in a significant fashion as process metrics measure activity and perfective changes are more likely subject to the design of modules than implementation activity.

As this is work in progress, the hypotheses are not verified or discarded based on the pilot study. There will be a finer set of more specific sub-hypotheses based on the data gathered

<sup>1</sup><http://ant.apache.org>

<sup>2</sup><http://www.apache.org>

in the pilot study for the final study (e.g., Perfective Changes reduce the consecutive fault rate of a module over a given period of time).

#### A. Classifying Change

We classify changes using an approach introduced by Mockus et al. in [7]. To test our approach we chose the Core Module of the Ant Project. We use scripts based on previous work of Mockus et al.<sup>3</sup> - a stand-alone tool is planned for the final study that integrates further process metrics and gathers the project data (bug database and version control repository data) by itself. The change data is gathered from a code repository, in this case it is Subversion (SVN)<sup>4</sup>. We gathered the changes over different time periods (2008 for Number of Deltas, 2003-2008 for Number of Faults and Developers) for a more diverse view on the data. We use the comments of each MR to classify the changes into corrective, adaptive and perfective changes. The WordNet<sup>5</sup> support to gather the stems of the words has not been implemented yet, but is planned for the categorization heuristic in the final study. Following the keyword clustering procedure, the classification rules had to be altered to provide meaningful results.

There are different approaches for classifying changes, e.g., the approach presented by Sliwerski et al. in [13]. However we were more interested in the maintenance tasks themselves than in the outcome (i.e., bug introduction). Additionally, we were interested in all changes not just the ones linked to the bug tracking system, as the data from the pilot study suggests that there are more corrective changes unrelated to bugs in the bug tracking system.

#### B. Gathering the Process Metrics

We will use the process metrics used by Graves et al. in [3] in the final study - a subset was chosen for the pilot study. To retrieve the data from Bugzilla<sup>6</sup> we employed scripts used by Mockus et al. in [6].

The following process metrics are included in the pilot study:

- **Number of past faults:** Graves et al. found that the number of faults in a module in the future are related to the faults found in the module in a past period of time; future faults are a constant multiple of past faults according to them. By classifying change we aim to prove that the consecutive fault rate decreases after a perfective change took place. We gather this metric by mining data from Bugzilla.
- **Number of deltas:** We will use the number of deltas to a module to find trends of increase and decrease depending on the kind of change that has been undertaken. According to Graves et al. a high number of deltas is an indicator for future faults in the module and thus corrective changes. The aim of this analysis is whether

there is something like the decay of a perfective change. The data required for this metric is the sum of all changes in a given time period.

- **Number of developers who have made deltas on the module:** We will analyze if the number of developers relates to changes undertaken in the module. A natural assumption here is that a high number of developers might lead to an increase in perfective changes later on in the module. This metric is gathered from the SVN by using StatSVN<sup>7</sup>.

Further planned process metrics for the final study:

- **Measure of the average age of the code:** This measure is calculated by using dates of changes weighted by the size of the changes.
- **Weighted time damp model:** Graves et al. use this model to compute a module's fault potential.

### III. ANALYZING THE RESULTS

We plan to analyze process metrics and their relationship to changes - one kind of analysis is done by correlating both measures with each other. We use the introduced set of process metrics on each kind of change and on all changes. In the second analysis we follow a different approach - we evolve the metric values and the changes in a time-series analysis, similar to Herraiz et al. in [5].

#### A. Number of Past Faults

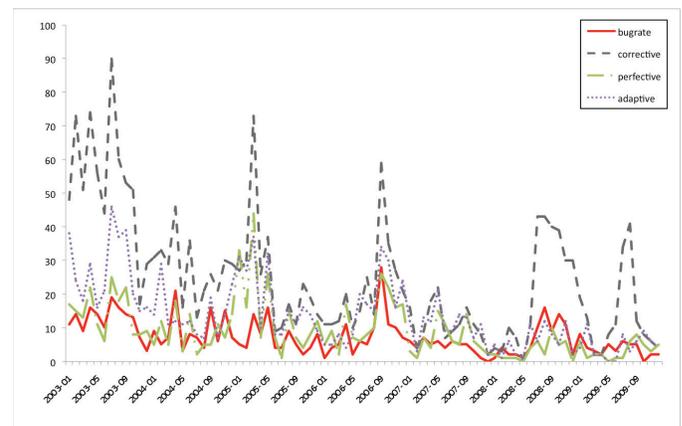


Fig. 1. Number of Faults and Classified Changes per Month (2003-2008)

Studies similar to the work presented by Sliwerski et al. in [13] show that there has been some effort to link corrective changes and bugs in a bug tracking system. However, as can be seen in Figure 1 corrective changes seem like multiples of the bug rate (a bug is not fixed by a single change and a bug might be reopened). If a large number of bugs are fixed without ever entering the bug tracking system, predicting fault data by using former fault rates needs to be reconsidered, even if one assumes that the behavior of not reported bugs is a constant factor (i.e., the number of bugs not reported does

<sup>3</sup><http://mockus.us/oss/>

<sup>4</sup><http://subversion.tigris.org>

<sup>5</sup><http://wordnet.princeton.edu/>

<sup>6</sup><https://issues.apache.org/bugzilla>

<sup>7</sup><http://www.statsvn.org/>

TABLE I  
NUMBER OF FAULTS AND CLASSIFIED CHANGES

Changetype	Correlation Values
Corrective	0,78
Perfective	0,59
Adaptive	0,61
All Commits	0,67

not change over time). While our pilot study seems to carry the current assumptions from bug data analysis, perfective changes and their impact on the fault rate provide a challenge for future analysis. When looking at Figure 1 we see two perfective spikes, which are followed by a decline in the bug rate - however adaptive changes and perfective changes spike there too, so a general change activity is more likely to have occurred than a suggested relationship of perfective changes and a temporarily lowered bug rate. The correlation values presented in Table I indicate that corrective changes correlate moderately with the fault rate. Perfective and adaptive changes correlate moderately as well with the fault rate, however are distinguishably lower than correlation of fault rate and corrective changes.

### B. Number of Deltas

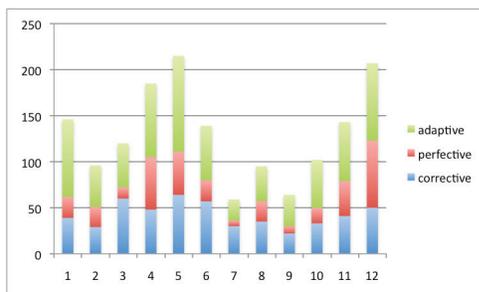


Fig. 2. Number of Deltas - Classified by Change Category (2008) per Month

In Figure 2 changes are split into the three categories, unspecified changes are not depicted, because the categorization technique is still being tuned. For the pilot study we assume the unspecified changes to be distributed according to their categorized appearance (adaptive and corrective accounting for the majority of the changes). About 25% of the changes could not be categorized by our solely automated categorization. We hope to decrease the amount by adding additional features to the mechanism. Figure 2 shows summed up changes (over the course of a year - 2008 - for better visibility). While corrective changes seem constant (in accordance to our hypotheses), perfective changes peak twice. One of the peaks happens before a peak of corrective and adaptive changes, however the sample size is too small to make any assumptions yet, since it can be seen in Figure 1 that corrective changes spike over a longer time period and this is in contrast to our hypotheses. An additional process metric to be taken into consideration for the final study is feature requests per month, to see whether adaptive changes are demand-driven, or just prone to activity

TABLE II  
NUMBER OF DELTAS AND CLASSIFIED CHANGES

Changetype	Correlation Values
Corrective	0,85
Perfective	0,89
Adaptive	0,9

schedules (see Figure 1 which shows corrective and adaptive changes peaking in similar timeframes). By normalizing the change numbers by activity, we might be able to smoothen the change curves. The correlations shown in Table II strongly indicate that all three types of changes are dependent on the number of all commits and that there is no "asynchronous" change type, i.e. a change that happens more frequently during low activity in the repository.

### C. Number of Developers Who Have Made Deltas on the Module

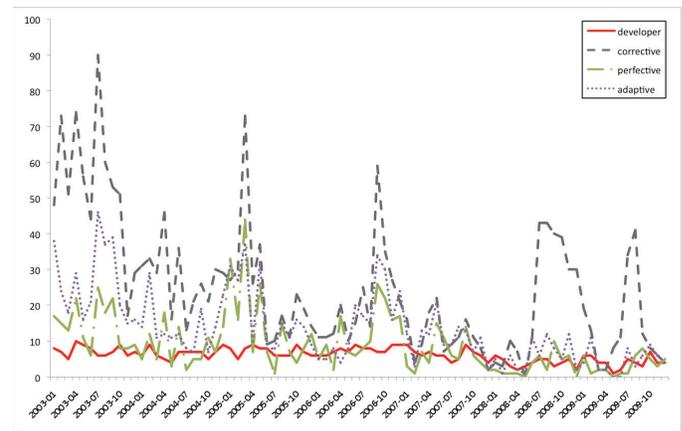


Fig. 3. Number of Developers and Classified Changes (2003-2008) per Month

As can be seen in Figure 3, the number of developers committing to the module seem constant, while the changes show peaks. It can be noted that the developer peak and the change peak fall within the same time frame, however further change peaks are not accompanied by developer peaks. The data gathered for the pilot study is not sufficient to make any assumptions about how the number of developers and changes of modules relate (Past research by Graves et al. show that fault data and number of developers do not relate significantly). This is further carried by the correlations shown in Table III. Curiously, adaptive changes correlate moderately with the developer number, indicating that feature adding procedures do increase with a higher developer number, while corrective and perfective tasks do not.

## IV. THREAT TO VALIDITY

As this is work in progress, the threats to validity of the presented study are many. We understand that for the final study, the change classification approach needs further validation and at least the WordNet support. As projects in the open source

TABLE III  
NUMBER OF DEVELOPERS AND CLASSIFIED CHANGES

Changetype	Correlation Values
Corrective	0,31
Perfective	0,44
Adaptive	0,53
All Commits	0,45

community are constantly evolving one has to keep in mind that statements about these projects are only temporary valid - there is no shipment and therefore a hypothesis like "Perfective changes reduce future bugs in the software" can only be proven temporarily (i.e., for a specific version). The values presented here are subject to change and present the values generated by our tools for the time being. The tools (e.g., the dictionary for the lexical approach) are still being tuned and might still show skewed results. Our lexical approach is not as successful yet as attempts in previous studies, however we did employ it on different Apache modules with similar performances (about half of the changes classified before tuning it).

#### A. Goals of the Pilot Study

The pilot study presented here should showcase an example of the data we will analyze. We present the pilot study to evaluate our approach and if the data can be gathered required can be gathered in a fashion that can be carried over to different projects with similar technological settings. Even though we can not discard or prove any hypotheses yet, the pilot study provides us with insights for more specific hypotheses.

### V. CONCLUSION

We showed that process metrics and classified changes raise interesting questions to expand on in future work. We explained the tools and techniques we plan to use in the final study and showed intermediate results on a single module of the Ant Project. We gave insight into the gathered data by showing the development of the metrics and dependent variables over time. One of the most interesting questions is on the relationship of bug tracking and corrective changes - for the field of bug prediction, findings here might have a considerable impact. Less evident from the data presented is the relationship of perfective changes and process metrics. Is the benefit of refactoring measurable? Are perfective changes capable predictors of temporary fault rate decreases? The pilot study itself has delivered trends and gave insight into the data - the results however have their drawbacks and potential pitfalls.

#### A. Future Work

We presented data in the pilot study that warrants a more thorough analysis, therefore we plan to do a regression analysis and a time-series analysis for the final study. It is planned to have a user-friendly tool for the analysis in the final study, so as many as possible projects can be analyzed. By enlarging the study we aim to derive a model that holds general validity, or if that is not feasible, a learning model that can be used

in any project after parameterization. We plan to augment our analysis by product metrics, which should provide interesting results especially for perfective changes. A further parameter that might be introduced into our analysis is the modification done in a change to further distinguish between categories of changes (additionally to integrating simple change size and interval into the classification mechanism). Furthermore we consider implementing an approach similar to Fluri et al. in [2] to deliver additional information about the change categories and adding thorough refactoring analysis to the categorization (as presented by Weissgerber et al. in [14]).

### REFERENCES

- [1] M. Cataldo, A. Mockus, J. A. Roberts, and J. D. Herbsleb. Software dependencies, work dependencies, and their impact on failures. *Software Engineering, IEEE Transactions on*, 35(6):864 – 878, Nov 2009.
- [2] B. Fluri and H. Gall. Classifying change types for qualifying change couplings. *Program Comprehension, 2006. ICPC 2006. 14th IEEE International Conference on*, pages 35 – 45, Jan.
- [3] T. Graves, A. Karr, J. Marron, and H. Siy. Predicting fault incidence using software change history. *Software Engineering, IEEE Transactions on*, 26(7):653 – 661, Jul 2000.
- [4] T. Gyimothy, R. Ferenc, and I. Siket. Empirical validation of object-oriented metrics on open source software for fault prediction. *IEEE Transactions on software engineering*, Jan 2005.
- [5] I. Herraiz, J. Gonzalez-Barahona, and G. Robles. Forecasting the number of changes in eclipse using time series analysis. *Mining Software Repositories, 2007. ICSE Workshops MSR '07. Fourth International Workshop on*, pages 32 – 32, Apr 2007.
- [6] A. Mockus, R. Fielding, and J. Herbsleb. Two case studies of open source software development: Apache and mozilla. *Transactions on Software Engineering and Methodology (TOSEM)*, 11(3), Jul 2002.
- [7] A. Mockus and L. Votta. Identifying reasons for software changes using historic databases. *Software Maintenance, 2000. Proceedings. International Conference on*, pages 120 – 130, Sep 2000.
- [8] A. Mockus, D. Weiss, and P. Zhang. Understanding and predicting effort in software projects. *ICSE '03: Proceedings of the 25th International Conference on Software Engineering*, May 2003.
- [9] N. Nagappan, T. Ball, and B. Murphy. Using historical in-process and product metrics for early estimation of software failures. *Proceedings of the 17th International Symposium on Software Reliability Engineering*, Jan 2006.
- [10] N. Nagappan, T. Ball, and A. Zeller. Mining metrics to predict component failures. *Proceedings of the 28th international conference on Software Engineering*, Jan 2006.
- [11] P. Rigby and A. Hassan. What can oss mailing lists tell us? a preliminary psychometric text analysis of the apache developer mailing list. *Mining Software Repositories, 2007. ICSE Workshops MSR '07. Fourth International Workshop on*, pages 23 – 23, Apr 2007.
- [12] H. Schackmann and H. Lichter. Evaluating process quality in gnome based on change request data. *Mining Software Repositories, 2009. MSR '09. 6th IEEE International Working Conference on*, pages 95 – 98, Apr 2009.
- [13] J. Śliwerski, T. Zimmermann, and A. Zeller. When do changes induce fixes? *MSR '05: Proceedings of the 2005 international workshop on Mining software repositories*, Jul 2005.
- [14] P. Weissgerber and S. Diehl. Identifying refactorings from source-code changes. *Automated Software Engineering, 2006. ASE '06. 21st IEEE/ACM International Conference on*, pages 231 – 240, Sep 2006.
- [15] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy. Cross-project defect prediction: a large scale experiment on data vs. domain vs. process. *ESEC/FSE '09: Proceedings of the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering on European software engineering conference and foundations of software engineering symposium*, Aug 2009.

# Cost-Effective Combinatorial Test Case Prioritization for Varying Combination Weights

Ziyuan Wang<sup>1,2</sup> Baowen Xu<sup>1,2\*</sup> Lin Chen<sup>1,2</sup> Zhenyu Chen<sup>2,3</sup>

<sup>1</sup>Department of Computer Science and Technology, Nanjing University, Nanjing, 210093, China

<sup>2</sup>State Key Laboratory of Novel Software Technology, Nanjing University, Nanjing, 210093, China

<sup>3</sup>Software Institute, Nanjing University, Nanjing, 210093, China

{wangziyuan, bwxu, lchen}@nju.edu.cn, zychen@software.nju.edu.cn

**Abstract**—Combinatorial testing has been widely used in practice. People usually assume all test cases in combinatorial test suite will be executed fairly. However, in many applications where combinatorial testing is needed, the entire test suite is not run as a result of test resource constraints. To improve the efficiency of testing, combinatorial test case prioritization technique is needed. This paper presents a novel cost-effective combinatorial test case prioritization technique, which takes combination weights and test costs into account. Firstly, a metric, which could incorporate varying combination weights and test costs, is proposed to assess the combinatorial coverage efficiency of combinatorial test suite. And secondly, two heuristic algorithms are proposed to prioritize combinatorial test cases. Finally, experimental results are given to illustrate the properties of proposed technique, and also prove its comparative advantage over existing methods.

**Keywords**—software testing, combinatorial testing, test case prioritization, metric, algorithm.

## I. INTRODUCTION

Combinatorial testing (CT) is a practical testing technique. For a system under test (SUT) that has  $n$  parameters (or factors), it is unacceptable to cover all the possible  $n$ -tuple combinations of parametric values, since the combinatorial explosion of parametric values. CT provides a tradeoff between the cost of testing and the degree of combinatorial coverage. For example, 2-way CT (also named as pair-wise testing) requires covering all the 2-tuple combinations rather than the  $n$ -tuples. CT has been studied and applied widely because of its efficiency and effectiveness. Especially, it is very effective for some highly-configurable systems [1].

In traditional testing process, all test cases in test suite will be run fairly and completely to achieve a given combinatorial coverage criteria. However, in many applications where CT is needed, for example regression testing, the entire combinatorial test suite is not run as a result of test resource constraints. To improve the efficiency of testing, the combinatorial test case prioritization technique, which prioritizes test cases to cover some important combinations of parametric values as soon as possible, is needed to extend traditional CT technique. Besides the importance of combinations of parametric values, it is also necessary to take the cost of test cases into account: the test cases that require less cost should be run as earlier as possible to save the testing resource. Without loss of generality, we assume the “cost of test case” means the consumed time of the running of such test case.

In this paper, a novel cost-effective combinatorial test case prioritization technique will be proposed. This technique takes both the importance of combination and the cost of test cases into consideration. The contributions of our work include: (1) proposing a novel metric to assess the combinatorial coverage efficiency of combinatorial test suite; (2) proposing two greedy algorithms, which prioritize test cases in existing combinatorial test suite; (3) show the properties of proposed technique and its advantage over some existing methods in experiments.

This paper is organized as follows. Section 2 presents some backgrounds. Section 3 proposes a metric for combinatorial test case prioritization. Section 4 describes two algorithms. Section 5 provides experimental results. And finally, section 6 gives a conclusion and future works.

## II. BACKGROUND

In this section, some backgrounds will be presented.

### A. Combinatorial Testing

In order to detect the faults that are triggered by interactions among parameters, combinatorial testing (CT) aims to cover some specific combinations of parametric values.

For a system under test (SUT) that has  $n$  parameters (or factors), we suppose each factor  $f_i$  has  $a_i$  discrete valid values ( $i=1, 2, \dots, n$ ). Without loss of generality, let  $F=\{f_1, f_2, \dots, f_n\}$  denote the set of factors, and  $V_i=\{1, 2, \dots, a_i\}$  denote the set of valid values for  $f_i$  ( $i=1, 2, \dots, n$ ).

**Definition 1.** The  $n$ -tuple  $test=(v_1, v_2, \dots, v_n)$  is a test case for SUT, where  $v_1 \in V_1, v_2 \in V_2, \dots, v_n \in V_n$ .

**Definition 2.** Given a 2-dimension array  $A=(a_{ij})_{m \times n}$ , where the  $j$ -th column is associated with the factor  $f_j \in F$  and there are  $a_{ij} \in V_j$  for  $j=1, 2, \dots, n$ . If each  $m \times \tau$  ( $1 < \tau < n$ ) sub-array contains all possible  $\tau$ -tuple combinations of parametric values of corresponding  $\tau$  parameters, then  $A$  is a  $\tau$ -way covering array (or a covering array with a fixed strength  $\tau$ ). It could be denoted as  $CA(m; \tau, F)$ .

For a given SUT, a  $\tau$ -way combinatorial test suite, which covers all  $\tau$ -tuple combinations of parametric values, could be obtained easily from a  $\tau$ -way covering array. Here, each row of covering array is a test case of test suite. Therefore, we say that combinatorial test suite and covering array are equivalent in this paper.

\* Corresponding Author

For example, we consider a system that is shown in Table I. Exhaustive testing requires  $3^4=81$  test cases. However, the 2-way combinatorial test suite (Table II) needs only 9 test cases to cover all the 2-tuple combinations of parametric values.

TABLE I. THE SUT WITH 4 PARAMETERS

ClassA	ClassB	ClassC	ClassD
A1	B1	C1	D1
A2	B2	C2	D2
A3	B3	C3	D3

TABLE II. PAIR-WISE TEST SUITE FOR SUT

ClassA	ClassB	ClassC	ClassD
A1	B1	C1	D1
A1	B2	C2	D2
A1	B3	C3	D3
A2	B1	C2	D3
A2	B2	C1	D2
A2	B3	C3	D1
A3	B1	C3	D2
A3	B2	C1	D3
A3	B3	C2	D1

One of the key challenges in CT is generating test suite with minimal number of test cases. But unfortunately, it has been proved that the problem of generating combinatorial test suite is NP-C [2]. To solve this problem, people develop many combinatorial test generation methods. Most of these methods could be categorized into algebraic methods, greedy methods, meta-heuristic search, and others [18].

### B. Test Case Prioritization

Test prioritization technique aims to schedule test cases in an order that increase their effectiveness at meeting some given performance goal. This problem could be defined as [4]:

**Definition 3.** For an initial test suite  $T_{init}$ ,  $PT$  is the set of permutations of  $T_{init}$ ,  $f$  is a function from  $PT$  to real numbers. Test prioritization aims to find a  $T \in PT$  such that:

$$(\forall T') (T' \in PT) (T \neq T') [f(T) \geq f(T')]$$

Here,  $PT$  is the set of all possible orderings of  $T_{init}$ . And  $f$  is an objective function that, applied to any such ordering, yields an award value for that ordering.

For different application scenarios, many test prioritization techniques and empirical studies have been reported in recent years. For example, there are techniques based on source code and test history [5][6], varying test costs and fault severities [7], basic block coverage [8], time budget [9], slicing result [10], and etc. To evaluate the prioritized test suite's efficiency of faults detection, the metrics APFD [6], APFD<sub>C</sub> [7], and NAPFD [11] are proposed. And there are also some metrics that consider other types of information, such as the metric APBC that evaluate the efficiency of basic block coverage [12].

### C. Combinatorial Test Case Prioritization

People began to study combinatorial test case prioritization technique. All these works can be addressed in two strategies: (1) *Pure Prioritization Strategy*: prioritize test cases in existing combinatorial test suite; and (2) *Re-Generation Strategy*: incorporate prioritization into combinatorial test generation.

For the situation that neither importance of combinations nor test costs are taken into consideration, R. C. Bryce et al proposed a metric for combinatorial test prioritization, and used the first strategy to prioritize combinatorial test cases [13].

The importance of combinations is also taken into account. The covering array that involved in the varying combination weights is named as the biased covering array. R. C. Bryce et al adopted the second strategy to generate biased covering array [14]. And X. Chen and Q. Gu used the ant colony algorithm to generate biased covering array [15].

To setup the weights of combinations that evaluate their importance, X. Qu and M.B. Cohen et al proposed the methods that based on code coverage and specification [11]. Especially, for the configurable systems, they proposed the weighting method that based on fault detection and configuration change [16]. And in their extending works, the cost of changing configurations is also taken into consideration [17].

## III. METRIC

Firstly, we will propose a metric that evaluate test suite's efficiency of combinatorial coverage.

### A. Limitation of Existing Metric

For the situation that neither importance of combinations nor test costs are taken into consideration, R. C. Bryce et al. defined a metric as follow [13]:

$$f(T) = \sum_{i=0}^m \left| \bigcup_{j=0}^i \text{tCov}(T[j]) \right|$$

Here, it is assumed that  $T = \{T[1], T[2], \dots, T[m]\}$ , and the  $T[i]$  denotes the  $i$ -th test cases in  $T$ . The  $\text{tCov}(T[i])$  computes the set of combinations that covered by the test case  $T[i]$ .

Besides the limitation that neither weights nor costs are taken into account, there are also some other limitations: (1) As the description of literature [13], there are  $m$  test cases in  $T$ . So the indexes of them should be  $0, 1, \dots, m-1$  or  $1, 2, \dots, m$ . It is impossible that both  $T[0]$  and  $T[m]$  appear in  $T$ . (2) In actual testing, the mission of test case is accomplished if and only if such test case has been run completely. It means that the  $\text{tCov}(T[i])$  should be counted after the execution of  $T[i]$ .

### B. Proposed Metric

In this sub-section, we will propose a novel metric for the situation that both combination weights and test cost are taken into consideration.

Considering the  $\tau$ -way test suite  $T = \{T[1], T[2], \dots, T[m]\}$ . The whole testing process should be considered as a discrete process including  $m$  steps. And in each step, a single test case will be run. So the metric, which measures the quality of the whole testing process, should have a form as follows:

$$f(T) = \sum_{i=1}^m h(i)$$

To define the  $h_{CT}(i)$  that measures the contribution of all executed test suite in the  $i$ -th step, we calculate the number of covered combinations. And in the  $i$ -th step, the  $T[1], T[2], \dots, T[i-1]$  have been run, while the  $T[i]$  is still running. So the set of combinations that covered by executed test cases should be:

$$\text{AddtlCov}(i) = \begin{cases} \emptyset, & i = 1 \\ \bigcup_{j=1}^{i-1} \text{tCov}(T[j]), & 2 \leq i \leq m \end{cases}$$

We take the combination weights into account. Assume all  $\tau$ -tuple combinations form a set  $CombSet$ , and the weight of  $\tau$ -tuple combination  $c_k \in CombSet$  is  $Weight(c_k) = cw_k$ . So the rate of total contributions of those executed test cases is:

$$\text{Contribute}(i) = \begin{cases} 0, & i = 1 \\ \frac{\sum_{c_k \in \text{AddtlCov}(i)} cw_k}{\sum_{c_k \in CombSet} cw_k}, & 2 \leq i \leq m \end{cases}$$

Then we take the test costs into account, and assume the cost of test case  $T[i]$  is  $Cost(T[i]) = tc_i$ . So in the  $i$ -th step, the total contributions of those executed test cases per unit cost is:

$$\text{ContributePerCost}(i) = \frac{tc_i}{\sum_{j=1}^m tc_j} \times \text{Contri}(i)$$

By summing  $\text{ContributePerCost}(i)$  for  $i=1, 2, \dots, m$ , we can get the a metric for the situation that both combination weights and test cost are taken into consideration:

$$f(T) = \frac{\sum_{i=2}^m (tc_i \times \sum_{c_k \in \text{AddtlCov}(i)} cw_k)}{(\sum_{j=1}^m tc_j) \times (\sum_{c_k \in CombSet} cw_k)}$$

To illustrate the APCC metric, we take a SUT with  $F=\{2^3\}$ , which means there are 3 parameters and each parameter has 2 values, as an example. We generate a 2-way combinatorial test suite (Figure 1(a)) that covers all the 2-tuple combinations of parametric values. Here, we assume all 2-tuple combinations have an equal weight. If the order of prioritized test cases is 1-2-3-4, the APCC metric value is 0.25 (Figure 1(b)); and if the order is 4-3-2-1, the APCC metric value is 0.5 (Figure 1(c)).

**Theorem 1.** Let the  $t$  is the time when the whole testing process is terminated exceptionally. If  $t \sim U[0, \sum_{j=1}^m tc_j]$ , then the physical sense of APCC metric is the mathematical expectation of the rate of total contributions of executed test cases when testing process is terminated.

**Proof:** When  $T[i]$  is still running, the rate of contributions of executed test cases is  $\text{Contribute}(i)$  ( $i=1, 2, \dots, m$ ). Note that  $t \sim U[0, \sum_{j=1}^m tc_j]$ , it means that the probability of the event that testing is terminated exceptionally when  $T[i]$  is still running is  $p(i) = tc_i / \sum_{j=1}^m tc_j$ . So the mathematical expectation ( $ME$ ) of the rate of total contributions of executed test cases when testing process is terminated could be described as:

$$\begin{aligned} ME &= \sum_{i=1}^m (p(i) \times \text{Contribute}(i)) \\ &= \sum_{i=1}^m \left( \frac{tc_i}{\sum_{j=1}^m tc_j} \times \frac{\sum_{c_k \in \text{AddtlCov}(i)} cw_k}{\sum_{c_k \in CombSet} cw_k} \right) \\ &= \frac{\sum_{i=2}^m (tc_i \times \sum_{c_k \in \text{AddtlCov}(i)} cw_k)}{(\sum_{j=1}^m tc_j) \times (\sum_{c_k \in CombSet} cw_k)} \end{aligned}$$

Therefore, we can conclude that the value of APCC metric is equal to the mathematical expectation of the rate of total contributions of executed test cases. ■

#### IV. ALGORITHM

After define the metrics, two algorithms, which prioritize test cases in an initial test suite  $T_{init}$  to increase their APCC metric value, will be proposed in this section.

##### A. Prioritize according to the total contribution per unit cost

Firstly, we propose an algorithm which prioritizes test cases according to the test case's total contribution per unit cost. Here, we define the total contribution per unit cost ( $TCPUC$ ) for each test case  $test \in T_{init}$ :

$$TCPUC(test) = \frac{\sum_{c_k \in \text{Cov}(test)} cw_k}{Cost(test)}$$

After calculate it for all  $m$  test cases in  $T_{init}$ , we sort test cases according to their  $TCPUC$  in descending order, and get a prioritized test suite  $T$ . And for the test cases that have the same priority, we select randomly. Algorithm 1, which adopts the selection sorting as an example, describes the details.

##### B. Prioritize according to the additional contribution per unit cost

Another opinion is that, all the contributions that have been made by previous test cases should be ignored when evaluate current test case. So it is reasonable to consider test case's additional contribution per unit cost ( $ACPUC$ ). We define the  $ACPUC$  for each  $test \in T_{init}$  that have not been selected:

$$ACPUC(test) = \frac{\sum_{c_k \in (\text{tCov}(test) - \text{tCov}(T))} cw_k}{Cost(test)}$$

Here,  $\text{tCov}(T)$  contains all the  $\tau$ -tuple combinations that have been covered by the test cases in prioritized test suite  $T$ . While  $\text{tCov}(test)$  contains all that covered by current  $test \in T_{init}$ .

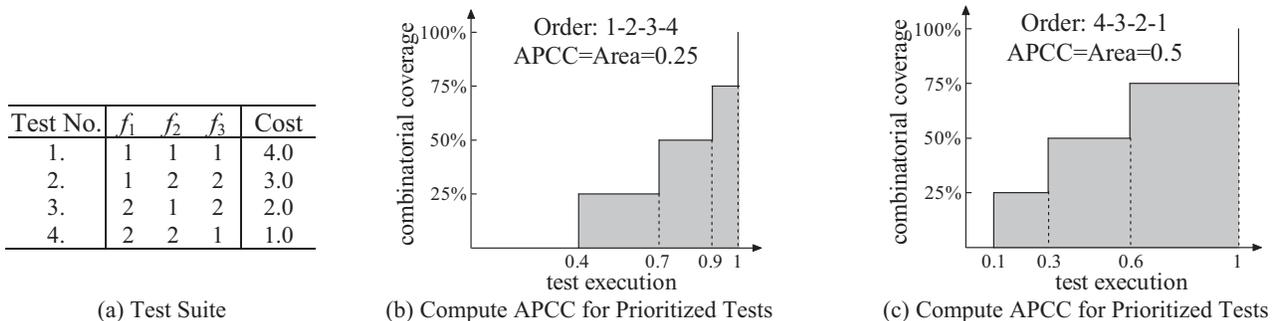


Figure 1. Example of Computing APCC Metric Values

**Algorithm 1.** Prioritize Test Cases According to the Total Contribution per Unit Cost (TotalCTPri)

**Input:**  $T_{init}$ , Weights of Combinations, Costs of Test Cases  
**Output:** Prioritized Test Suite  $T$

1. Initialize  $T[1..m]$  and  $Count[1..m]$
2. **for**  $i:=1$  **to**  $m$
3.    $Count[i]:=TCPUC(T_{init}[i])$
4. **end for**
5. **for**  $i:=1$  **to**  $m$  //Selection sort according to  $TCPUC$
6.    $Max:=0; Best:=0$
7.   **for**  $k:=i$  **to**  $m$
8.     **if** ( $T_{init}[k] \notin T$  **AND**  $Count[k]>Max$ ) **then**
9.        $\{Max:=Count[k]; Best:=k\}$
10.    **end if**
11.   **end for**
12.    $T[i]=T[Best]$
13. **end for**

At each time, we calculate the  $ACPUC$  for all test cases that have not been selected into the prioritized test suite  $T$ , and select one with the highest  $ACPUC$ . And for the test cases that have the same priority, we select randomly. This operation will repeat until all test cases in  $T_{init}$  have been selected into  $T$ . Algorithm 2 describes the details.

## V. EXPERIMENTS

To assess the efficiency of proposed combinatorial test case prioritization technique, we will design some experiments.

### A. Performance of proposed algorithms

In the firstly experiment, we will evaluate the performance of proposed algorithms TotalCTPri and AddtlCTPri.

#### 1) Experimental setup

Firstly, generate initial combinatorial test suite  $T_{init}$ . We select 4 systems under test (SUT) and 3 classic combinatorial test generation algorithms including Orthogonal Array (OA) [3], Recursion Construction (RC) [19], and DDA [20]. For each SUT, we only adopt the best one (“best” means generate the smallest test suite) of all available algorithms to generate 2-way combinatorial test suites. The details about SUTs and selected algorithms are shown in Table III.

TABLE III. TEST GENERATION ALGORITHMS AND SUTS

$S_1: 5^6$	Orthogonal Array
$S_2: 3^{13}$	Recursion Construct
$S_3: 2^3 \times 3^3 \times 4^3 \times 5$ $S_4: 2^4 \times 6^2 \times 7^2 \times 8^2$	DDA

Secondly, setup the weights of 2-tuple combinations. We adopt following 4 weight distribution styles, which have been described in [14], to assign weight for each parametric value. And then, the weight of combination could be calculated by multiplying the weights of all corresponding parametric values.

- *Equal* (E) : all values have the same weight 1;
- *Random* (R): weights are distributed randomly;

**Algorithm 2.** Prioritize Test Cases According to the Additional Contribution per Unit Cost (AddtlCTPri)

**Input:**  $T_{init}$ , Weights of Combinations, Costs of Test Cases  
**Output:** Prioritized Test Suite  $T$

1. Initialize  $T[1..m]$
2. **for**  $i:=1$  **to**  $m$
3.    $Max:=0; Best:=0$
4.   **for**  $k:=1$  **to**  $m$
5.     **if** ( $T_{init}[k] \notin T$ ) **then**
6.        $Count:=ACPUC(T_{init}[k])$
7.       **if** ( $Count>Max$ ) **then**
8.          $\{Max:=Count; Best:=k\}$
9.       **end if**
10.    **end for**
11.   **end for**
12.    $T[i]=T_{init}[Best]$
13. **end for**

- *Reciprocal* (P): all value weights for  $f_i$  are  $(1/a_i)^2$ ;
- *Half* (H): half of the weights for each parameter are set to 0.1, and other half to 0.9.

Thirdly, setup the costs for test cases. We use following 4 cost distribution styles that described in [7] to assign costs:

- *Equal* (E) : all test cases have the same cost 1;
- *Random* (R) : costs are distributed randomly;
- *Mozilla* (M): see Table IV;
- *QTB* (Q): see Table V.

TABLE IV. COST DISTRIBUTION IN MOZILLA

Name	Level	Description	Percent
HTML	1	least	87%
Printing	2	normal	1%
Smoke tests	3	more	2%
Buster	4	most	10%

TABLE V. COST DISTRIBUTION IN QTB

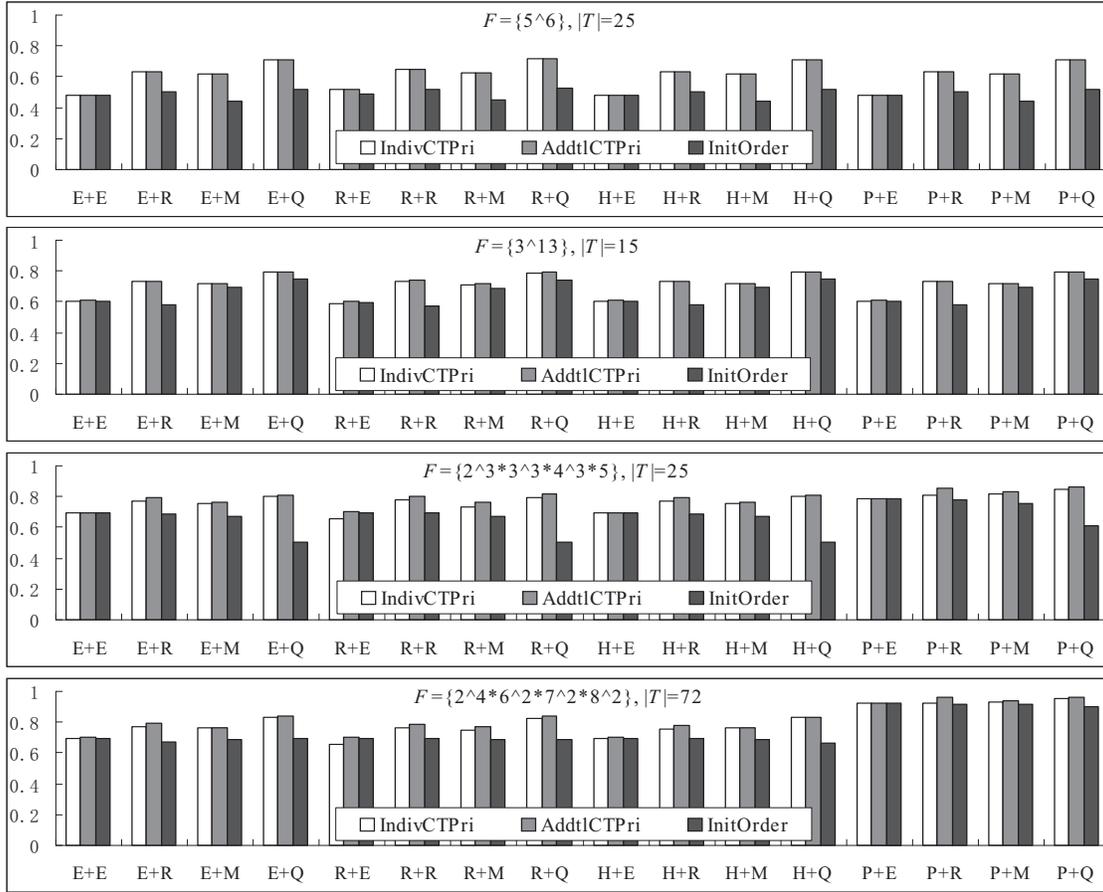
Level	Description	Percent
1	less	88%
10	more	12%

So there are totally 16 different combined distributions. For 4 SUTs and 16 combined distributions, we run proposed two algorithms and record the results in Figure 2. In which the horizontal axis denotes the APCC values, and the horizontal axis denotes the combined distributions.

#### 2) Result and discussion

As displayed in the figures 2, it can be concluded that the combinatorial test case prioritization technique can improve the performance, which is measure by the proposed APCC metric, of combinatorial test suite evidently.

And comparing two proposed algorithms, it is evident that the performance of TotalCTPri is a little worse than that of AddtlCTPri for most distributions. But the gap between the performances of two proposed algorithms is not very big.



**Figure 2.** APCC Metric Value for Different Prioritization Algorithms and Combined Distributions

### B. Compare to existing techniques

In the second experiment, we will compare our proposed technique to a related technique, which includes an algorithm WDDA [14]. The technique in literature [13] is not taken into consideration, since our technique will be equal to it if both the combination weights and the test costs are ignored.

#### 1) Experimental setup

Firstly, for 4 different SUTs that shown in Table III, we set weights for parametric values according to 4 different weight distributions: *Equal*, *Random*, *Half*, and *Reciprocal*. And compute weights of all 2-tuple combinations. Then, use WDDA algorithm to generate prioritized pair-wise test suites  $T_{WDDA}$  for different SUTs and weight distributions respectively, the sizes of generated test suites are displayed in Table VI.

TABLE VI. SIZE OF GENERATED TEST SUITES

Algorithm	Weight Distribution	S <sub>1</sub>	S <sub>2</sub>	S <sub>3</sub>	S <sub>4</sub>
WDDA	<i>Equal</i>	16	19	25	72
	<i>Random</i>	25	20	26	73
	<i>Half</i>	16	19	25	72
	<i>Reciprocal</i>	16	20	27	77
Normal Algo	-	16 <sup>(a)</sup>	15 <sup>(b)</sup>	25 <sup>(c)</sup>	72 <sup>(c)</sup>

S<sub>1</sub>: 4<sup>5</sup>, S<sub>2</sub>: 3<sup>13</sup>, S<sub>3</sub>: 2<sup>3</sup>×3<sup>3</sup>×4<sup>3</sup>×5, S<sub>4</sub>: 2<sup>4</sup>×6<sup>2</sup>×7<sup>2</sup>×8<sup>2</sup>

a: Orthogonal Array, b: Recursive Construct, c: DDA

For comparison, we consider initial combinatorial test suites of 4 SUTs, and list the sizes of them in Table IV. The same as before, initial test suites  $T_{init}$  are prioritized by the TotalCTPri and AddtlCTPri to get  $T_{Total}$  and  $T_{Addtl}$  respectively.

Note that for many SUTs and weight contributions, the sizes of  $T_{WDDA}$  are bigger than that of corresponding initial test suite (also the sizes of  $T_{Total}$  and  $T_{Addtl}$ ). That means the total costs of  $T_{WDDA}$ ,  $T_{Total}$  and  $T_{Addtl}$  are different. So it is necessary to modify them before comparing: if  $|T_{init}| < |T_{WDDA}|$ , then create  $|T_{WDDA}| - |T_{init}|$  test cases randomly, and insert these test cases into  $T_{Total}$  and  $T_{Addtl}$  as the  $|T_{init}|$ ,  $|T_{init}|+1$ , ...,  $|T_{WDDA}|$ -th test cases; or vice versa.

#### 2) Result and discussion

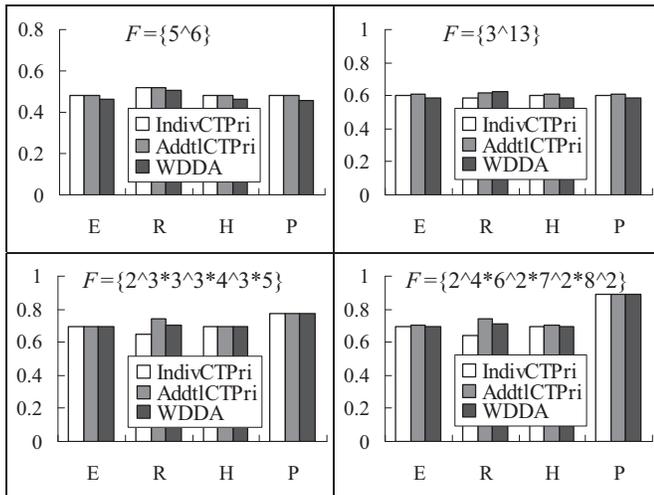
By analyzing the comparison results that shown in Figure 3, we can find following conclusions:

When algebraic methods (OA or RC) are available, the prioritized test suites that generated by WDDA are usually bigger than test suites that generated by the algebraic methods. And the APCC values of  $T_{WDDA}$  are less than that of  $T_{Total}$  and  $T_{Addtl}$  almost all the time. So it is evident that our proposed technique has advantage in this circumstance.

When algebraic methods are not available, we use DDA to generate initial test suites. (1) If the weight distribution is set as *Equal* or *Half*, the sizes of  $T_{WDDA}$  and  $T_{init}$  are approximate and

even equal sometimes. And the APCC values of  $T_{WDDA}$ ,  $T_{Total}$  and  $T_{Addtl}$  are also approximate and even equal. (2) If the weight distribution is *Random* or *Reciprocal*, the sizes of  $T_{WDDA}$  are always bigger than that of  $T_{init}$  (this phenomenon is also described in [14]). And APCC values of  $T_{WDDA}$  are less than that of  $T_{Total}$  and  $T_{Addtl}$ .

Above all, it is concluded that the combinatorial test case prioritization technique that proposed in this paper has some advantage over existing methods.



**Figure 3.** APCC Metric for Different Prioritization Techniques and Weight Distributions

## VI. CONCLUSION

Both combinatorial testing and test case prioritization are practical software testing techniques. Therefore, peoples have begun to study combinatorial test prioritization techniques. Considering the circumstance that both combination weights and test costs are varying, this paper described a novel cost-effective combinatorial test case prioritization technique, which includes a metric to assess the performance of test suites and two algorithms to prioritize existing test cases. Computational results suggest that our proposed algorithms can improve the performance of existing test suite. It is also indicated that the proposed technique has some advantage over some existing combinatorial test case prioritization techniques.

Although some methods that assign weights to parametric values and combinations have been reported [11][16], it still need to be studied in our future works to assist more widely application of combinatorial test case prioritization techniques. Besides, empirical study is also required to assess the fault detection efficiency of prioritized combinatorial test suite in different types of real software systems.

## ACKNOWLEDGMENT

The work described in this paper was partially supported by the National Natural Science Foundation of China (60773104, 60803007, and 90818027); and the National High Technology Research and Development Program of China (863 Program: 2008AA01Z143, 2009AA01Z147).

## REFERENCES

- [1] C. Yilmaz, M. B. Cohen, A. A. Porter. Covering Arrays for Efficient Fault Characterization in Complex Configuration Spaces. IEEE Transaction on Software Engineering, 2006, 32(1): 20-34.
- [2] G. Seroussi, N. H. Bshouty. Vector Sets for Exhaustive Testing of Logic Circuits. IEEE Transaction on Information Theory, 1988, 34(3): 513-522.
- [3] R. Mandl. Orthogonal Latin Squares: An Application of Experimental Design to Compiler Testing. In Communications of the ACM, 1985, 28(10): 1054-1058.
- [4] S. Elbaum, A. G. Malishevsky, G. Rothermel. Prioritizing Test Cases for Regression Testing. In Proceedings of the ACM International Symposium on Software Testing and Analysis (ISSTA2000): 102-112.
- [5] W. E. Wong, J. R. Horgan, S. London, H. Agrawal. A Study of Effective Regression Testing in Practice. In Proceedings of 8th IEEE International Symposium on Software Reliability Engineering (ISSRE1997): 264-274.
- [6] G. Rothermel, R. H. Untch, C. Y. Chu, M. J. Harrold. Prioritizing Test Cases for Regression Testing. IEEE Transactions on Software Engineering, 2001, 27(10): 929-948.
- [7] S. Elbaum, A. Malishevsky, G. Rothermel. Incorporating Varying Test Costs and Fault Severities into Test Case Prioritization. In Proceedings of the 23rd International Conference on Software Engineering (ICSE2001): 329-338.
- [8] A. Srivastava, J. Thiagrajan. Effectively Prioritizing Tests in Development Environment. In Proceedings of the ACM International Symposium on Software Testing and Analysis (ISSTA2002): 97-106.
- [9] K. R. Walcott, M. L. Soffa, G. M. Kapfhammer, R. S. Roos. Time-Aware Test Suite Prioritization. In Proceedings of the ACM International Symposium on Software Testing and Analysis (ISSTA2006): 1-12.
- [10] D. Jeffrey, N. Gupta. Test Case Prioritization Using Relevant Slices. In Proceedings of Computer Software and Applications Conference (COMPSAC2006): 411-420.
- [11] X. Qu, M. B. Cohen, K. M. Woolf. Combinatorial Interaction Regression Testing: A Study of Test Case Generation and Prioritization. In Proceedings of 23th IEEE International Conference on Software Maintenance (ICSM2007): 255-264.
- [12] Z. Li, M. Harman, R.M. Hierons. Search Algorithms for Regression Test Case Prioritization. IEEE Transactions on Software Engineering, 2007, 33(4): 225-237.
- [13] R. C. Bryce, A. M. Memon. Test Suite Prioritization by Interaction Coverage. In Proceedings of Workshop on Domain-Specific Approaches to Software Test Automation (DoSTA2007): 1-7.
- [14] R. C. Bryce, C. J. Colbourn. Prioritized Interaction Testing for Pair-wise Coverage with Seeding and Constraints. Information and Software Technology, 2006, 48(10): 960-970.
- [15] Xiang Chen, Qing Gu, Xin Zhang, Daoxu Chen. Building Prioritized Pairwise Interaction Test Suites with Ant Colony. In Proceedings of the 9th International Conference on Quality Software (QSIC2009): 347-352.
- [16] X. Qu, M. B. Cohen, G. Rothermel. Configuration-Aware Regression Testing: An Empirical Study of Sampling and Prioritization. In Proceedings of International Symposium on Software Testing and Analysis (ISSTA2008): 75-85.
- [17] H. Srikanth, M. B. Cohen, X. Qu. Reducing Field Failures In System Configurable software: Cost-based Prioritization. In Proceedings of International Symposium on Software Reliability Engineering (ISSRE2009): 61-70.
- [18] Changhai Nie, Hareton Leung, Baowen Xu. A Survey of Combinatorial Testing. Computing Surveys, 2010, to appear.
- [19] C. J. Colbourn, S. S. Martirosyan, T. V. Trung, R. A. Walker II. Roux-Type Constructions for Covering Arrays of Strengths Three and Four. Designs, Codes and Cryptography, 2006, 41(1): 33-57.
- [20] R. C. Bryce, C. J. Colbourn. A Density-Based Greedy Algorithm for Higher Strength Covering Arrays. Software Testing, Verification and Reliability, 2009, 19(1): 37-53.

# A Multi-State Bayesian Network for Shill Verification in Online Auctions\*

Ankit Goel<sup>1</sup>, Haiping Xu<sup>1</sup> and Sol M. Shatz<sup>2</sup>

<sup>1</sup>Computer and Information Science Department

University of Massachusetts Dartmouth, North Dartmouth, MA 02747, USA

{agoel, hxu}@umassd.edu

<sup>2</sup>Computer Science Department

University of Illinois at Chicago, Chicago, IL 60607, USA

shatz@uic.edu

**Abstract.** *Online auction systems have made remarkable progress in recent years. However, one of the most severe and persistent problems in such systems is shilling behavior, which is a type of auction fraud where a bidder artificially drives up the bidding price so that the winner of the auction has to pay more than he otherwise would pay. Verification of shill bidders in an online auction is difficult due to incomplete knowledge about suspicious bidders. In this paper, we introduce a novel approach for verifying shill bidders using a multi-state Bayesian network, which supports reasoning under uncertainty. We describe how to construct the multi-state Bayesian network and present formulas for calculating the probabilities of a bidder being a shill and being a normal bidder. To illustrate the effectiveness of our approach, we provide a case study for shill verification, and demonstrate that a multi-state Bayesian network performs better than a bi-state Bayesian network.*

## 1. Introduction

Online auctions have become an integral part of e-commerce. An online auction system provides a platform for people from different walks of life and geographic locations to come together for the purpose of exchange of services, goods or money. In the spirit of the auction, the highest bidder becomes the winner of the auctioned item. As the number of people participating in online auctions increases, online auction systems are experiencing an immense volume of auction-based trading, as well as the problems associated with this volume of traffic.

Despite the popularity of online auctions in recent years, one of the biggest problems in online auctions, namely *shill bidding* [1-3], remains unchecked. Shill bidding is a bidding activity that artificially increases an auctioned item's price or apparent desirability. To engage

in shill bidding, a seller might recruit a fake bidder, or create a pseudo-bidder account, to place bids that are solely intended to raise the price of an auctioned item. In this case, an honest bidder may become a victim of shill bidding and not even be aware that such activity has occurred. As a concrete example, consider auction  $A$  held by seller  $S$ . If the current price of the auctioned item is below the expectation of the seller,  $S$  can employ bidder  $B$  to bid and raise the price. Once the auction price has reached a certain satisfactory value,  $B$  stops bidding to avoid accidentally winning the auction.

Shill bidding is not a new phenomenon in the domain of online auctions; however, the development of effective ways to detect and verify shill bidding in online auctions is still an open problem [3]. Since both a seller and a buyer can be anonymous and managed by a single person with multiple accounts, it is very hard to discover the actual relationship between sellers and bidders. Furthermore, the actions of a shill bidder are seemingly close to normal bidding, which make it difficult to differentiate between such a shill bidder and a normal bidder. To safeguard the interests of legitimate users, researchers have attempted to identify various shill bidding patterns [4-6]. Meanwhile, many popular auction systems have implemented techniques aimed at curbing shilling activity. For example, eBay has a reputation point system where each buyer or seller has a reputation score called a *feedback score*. This score reflects a user's trustworthiness as evaluated by other users with whom this user has traded. However, most of the methods deployed so far cannot establish a shill bidder due to a lack of complete information about the bidder, thus allowing guilty parties to go unpunished.

In this paper, we propose a Bayesian network based system that can handle incomplete knowledge and help verify whether a shill suspect is an actual shill. Figure 1 shows a framework for shill detection and verification. Based on online auction data and shill bidding patterns, we can detect shill suspects using existing approaches, such as the data mining method [7] and real-time model

---

\* This material is based upon work supported by the U.S. National Science Foundation under grant numbers CNS-0715648 and CNS-0715657.

checking mechanism [8]. However, there is no guarantee that a skill suspect is an actual skill because it is possible for a normal bidder to coincidentally demonstrate some shilling behaviors, although the bidder has no intention at all to be a skill. In order to verify if a skill suspect is an actual skill, we must use additional evidence to reason about the skill suspect's possibility for being a skill or a normal bidder. Our approach employs a multi-state Bayesian network as a verification engine. Once an actual skill is confirmed, the involved auction would likely be cancelled in order to protect other users' interests. Note that although skill detection is an important component of our proposed framework, details about detection mechanisms are beyond the scope of this paper.

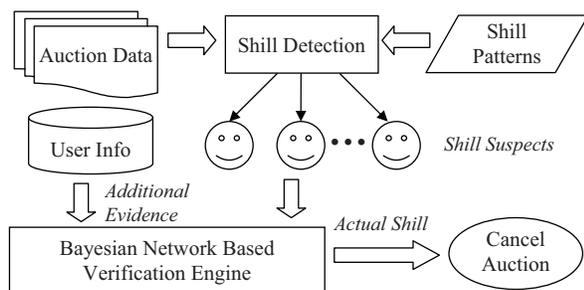


Figure 1. Shill detection and verification framework

## 2. Related Work

Most of the previous work related to auction fraud focuses on detection of in-auction fraud, which happens while transactions are in progress. Such auction fraud occurs disguisedly, leaving behind no obvious evidence; thus the victims typically do not even realize that the fraud has taken place. In order to effectively detect shill bidding, which is a major type of in-auction fraud, researchers have summarized various shill and normal bidding patterns [4-6]. Such bidding patterns are important knowledge for detection of shill suspects; however, they are not sufficient for justifying the presence of shills because some shill bidding patterns can be considered as normal behaviors under certain circumstance. For example, when an auctioned item is a rare and very hard to find item, a bidder may place very high bids in order to win the auction since the opportunity to bid on such an item might not present itself in the future. In this case, the bidder's bidding behavior may match with some shill bidding patterns; however, the bidder has no intention at all in being a shill.

Other researchers have utilized statistical and data mining techniques to detect abnormal bidding behaviors. Kauffman and Wood used a statistical approach to examining reserve price shilling behaviors and presented the factors that lead to this behavior [9]. They also showed how to use an empirical model to test for questionable and opportunistic bidding behaviors. Chau

and Faloutsos proposed a data mining method to detect auction fraud by extracting characteristic features from exposed fraudsters [10]. They determined the significant features related to auction fraud by analyzing the fraudsters' transaction histories, which exist as graphs. More recently, Xu, et. al proposed a real-time model checking approach to detect shill suspects in auctions that are in progress, or "live" [8]. This approach introduced a dynamic auction model (DAM) that can be used to detect shilling behaviors formally specified using linear temporal logic (LTL). Although the above approaches can be effective in detecting shill suspects in online auctions, they are not sufficient for determining whether a shill suspect is an actual skill. In contrast, the focus of this paper is to introduce a verification engine that supports reasoning under uncertainty for shill verification. As such, this work is complementary to other research efforts for detection of shill suspects.

Previous work on shill verification is rare. Dong, et. al used Dempster-Shafer (D-S) theory to verify whether a user is a shill using additional evidence from both auction data and user information [11]. This is different from our approach because in our approach, the evidence set is based solely on user information. Note that using our shill detection and verification framework, the auction data has already been used in the first stage – for detection of shill suspects, we do not use the same information again in the second stage – for shill verification. Therefore, our approach requires less computation, and thus it is more efficient for shill verification than the D-S based method.

## 3. Bayesian Network Based Verification Engine

### 3.1 Bayesian Network for Shill Verification

Bayesian network (BN) or belief network is a probabilistic graph model that can be used to capture uncertain knowledge in a natural and efficient way [12]. BN models the dependencies among variables and gives a concise specification of full joint probability distribution. BN is a directed acyclic graph (DAG) where each variable is denoted by a node, and the probability of a node is conditionally dependent on its parent node(s). The nodes are selected from the same domain, such that they help in the decision making process. It is vital that these nodes represent major features in the domain, which contribute factors in decision making or influencing the variables that affect the major calculation.

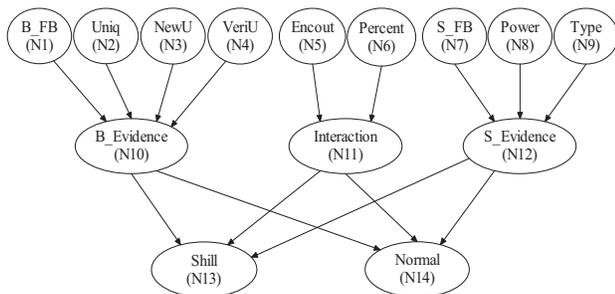
The basic task for a probabilistic inference system using BN is to calculate the posterior probability for a query variable  $X$ , given a set of observed evidence  $\mathbf{e}$  for a set of evidence variables  $\mathbf{E}$ . This is done by summing terms of the full joint distribution as follows [12]:

$$P(X|\mathbf{e}) = \alpha P(X, \mathbf{e}) = \alpha \sum_y P(X, \mathbf{e}, y) \quad (1)$$

where  $X$  is the query variable,  $\mathbf{e}$  is the observed values

for a set of evidence variables  $\mathbf{E}$ ,  $\mathbf{y}$  is the values for a set of unobserved (or hidden) variables  $\mathbf{Y}$ , and  $\alpha$  is the normalization factor. Note that  $\mathbf{P}(X, \mathbf{e}, \mathbf{y})$  is the full joint probability of  $X$ ,  $\mathbf{e}$  and  $\mathbf{y}$ .

Figure 2 shows a BN with three layers that we designed for skill verification. The first layer defines three groups of evidence nodes, namely evidence nodes for determining the strength of bidder's evidence (N1-N4), evidence nodes for determining the strength of seller's evidence (N7-N9), and evidence nodes for determining the interaction strength of the involved bidder and seller (N5, N6). Nodes N1-N9 represent evidence sources that are observable in the context of eBay. To simplify the network, we define three unobservable nodes (N10-N12) in the second layer, whose parents are the aforementioned three groups of nodes, respectively. The values of the three layer-two nodes (N10-N12) can be used to determine the conditional probabilities of being a skill or a normal bidder, which are defined as nodes *Skill* (N13) and *Normal* (N14) in the third layer.



**Figure 2.** Bayesian network for skill verification

We now explain each of the nodes (N1-N14) defined in Figure 2.

**Bidder Feedback (N1)** represents a range of feedback score values for a particular bidder. In eBay, feedback scores for bidders are represented by different colored stars. For example, a *yellow* star represents that the bidder has a feedback score in the range of 10 to 49. A reputation mechanism, such as feedback stars, serves as a deterrent for bad user service and illegal actions; it can also be used to gauge the trustworthiness of a user. Obviously, a bidder with a high score has been in the system longer and is trusted by many people. Since it can take a long time and serious effort for a user to obtain a high feedback score, a legitimate bidder is not likely to risk sacrificing this score lightly by engaging in skill bidding.

**Number of Unique Sellers (N2)** is the number of different sellers whose auctions the bidder has participated in over the past 30 days. Most skill bidders frequently trade with the same seller or a set of sellers. Knowing the number of different sellers with whom the

bidder has interacted, helps us understand the bidder's general activity with the auction community.

**New User (N3)** is a person who registered recently, possibly within the last 30 days. eBay assigns a special icon to such users. A new user has little or no associated history and has the least to lose if banned for fraud. Thus, new users are more likely to appear on a suspect list than experienced users. Note that since a user can have multiple accounts, it is easy for a user to create a new account for skill bidding.

**Verified User (N4)** is a registered user who has provided eBay with further proof of identity. To become a verified user with eBay, the user must provide personal information (e.g., home address and birth date), which is processed through a third party company such as Equifax. Since a verified user's personal information is recorded, such a person typically would not risk skill bidding and accidentally being caught.

**Encounters (N5)** is the number of auctions held by seller  $S$ , in which the bidder placed a bid in the past 30 days. Although a high number of encounters may imply that the bidder simply prefers seller  $S$ , it is also possible that the bidder is serving as a skill bidder for  $S$ .

**Bid Percentage (N6)** is the percentage of bids placed by the bidder in auctions held by seller  $S$  over the total number of bids placed by the bidder in the past 30 days. Although a high bid percentage may be due to the competitiveness of the auctions held by seller  $S$ , it is also an indicator of skill bidding because it implies a very close relationship between the bidder and seller  $S$ .

**Seller Feedback (N7)** is the feedback assigned to a seller. Again, eBay uses different colored stars for different feedback score ranges. Similar to the bidder feedback star, a seller with a good feedback score typically does not want to risk losing that score by being involved in skill bidding. Note that a high feedback score is critical to a seller because most bidders prefer to buy items from a trustworthy seller, even at the cost of paying more money for the auctioned item, as it guarantees them proper and timely delivery.

**Power Seller (N8)** is a status assigned by eBay to some sellers who consistently sell a significant volume of items, maintain a 98% positive feedback rating, and provide a high level of service to their buyers. There are multiple levels of power seller like *Gold*, *Silver* and *Bronze*. Obviously, being a power seller is a positive indicator that that seller is not involved in skill bidding.

**Seller Type (N9)** of a seller can either be *private* or *store* at eBay. To be an eBay store, the seller must have an eBay seller account with credit card information on file. The seller must also be a verified user or have a feedback score of at least 20. So a "store seller" tends to be trusted by bidders, and is typically not likely to be involved in skill bidding.

**Bidder Evidence Level (N10)** is evaluated based on the bidder’s evidence information from its parent nodes. A high bidder-evidence (strength) level implies the bidder is more likely an actual skill.

**Interaction Level (N11)** represents the strength of interaction between the bidder and seller in question. A high level of interaction implies a preferred seller or an unhealthy relationship between the two parties.

**Seller Evidence Level (N12)** considers the probability that the seller is involved in shill bidding. Note that if we implicate a bidder in an auction, the seller of the same auction must also be implicated. Similar to bidder evidence level, a seller is judged based on the seller’s evidence information from its parent nodes.

**Shill Bidder (N13)** indicates whether a bidder suspect is an actual shill given additional evidence. This node represents the prior probability that the bidder is an actual shill based on the states and probabilities of its parent nodes, namely N10-N12.

**Normal Bidder (N14)** indicates whether a bidder suspect is actually a normal bidder given additional evidence. This node represents the prior probability that the bidder is actually a normal bidder based on the states and probabilities of its parent nodes, namely N10-N12.

Note that we consider the two nodes N13 and N14 as conditionally independent. Thus given a set of evidence  $\mathbf{e}$  (i.e., values of evidence variables N1-N9), the summation of probabilities of being a shill and a normal bidder,  $P(\text{shill}|\mathbf{e}) + P(\text{normal}|\mathbf{e})$ , does not necessarily equal to 1.

### 3.2 Multi-State Bayesian Network

In a BN, each node typically takes only one of two values, such as *true* and *false*. This type of BN is called a *bi-state* BN. The second column in Table 1 shows the two states for each of the nodes we are considering. For example, node N1 in a bi-state BN can be in a state of *low* or *high*. N1 has the value of *low* when the bidder feedback star is *Turquoise* or below; otherwise, N1 has the value of *high*. Note that the values corresponding to each state are determined based on the actual data distribution of the original information from eBay. In other words, the states are assigned based on how shills and normal bidders are clustered, but only through approximation and observation.

In order to make the BN reflect the dependencies among different nodes more precisely, we consider a *multi-state* BN, where the nodes are not limited to two states. The third column of Table 1 shows the use of multiple states for some nodes. For example, node N1 in a multi-state BN can be in one of the following states *None*, *Yellow*, *Blue*, *Turquoise*, or *Other*, which correspond to different levels of a bidder feedback star. In our case study, we demonstrate that a multi-state BN performs better than a bi-state BN for shill verification.

Table 1. State values for nodes N1-N9

Node	Bi State	Multi State
N1	Low/High	None/Yellow/Blue/Turquoise/Other
N2	Low/High	1 / 2-5 / 6-15 /Other
N3	True/False	True/False
N4	True/False	True/False
N5	Low/High	Low/High
N6	Low/High	No more than 30 / 31-80 / Other
N7	Low/High	None/Yellow/Blue/Turquoise/Other
N8	Low/High	None/Bronze/Other
N9	Store/Private	Store/Private
N10	Low/High	Low/High
N11	Weak/Strong	Weak/Strong
N12	Low/High	Low/High
N13	Yes/No	Yes/No
N14	Yes/No	Yes/No

### 3.3 Conditional Probability Table

The prior knowledge of the BN can be derived from eBay auction data as well as user information. The auction data and user information were collected from eBay using the Trading APIs available for developers [13], which allows a developer to retrieve various types of data via different web service invocations. It is important to note that information retrieved about a bidder is limited; one can only obtain information that is available for the last 30 days before the time of data collection. This is in accordance with eBay’s privacy policies, which prevents other users from acquiring unlimited information regarding a bidder in an auction. Furthermore, all bidders are named anonymously (e.g.,  $x^{**y}$ ) – only a seller of an auction may see the real user identifications of the bidders participating in the auction.

We represent the prior knowledge of the BN as conditional probability tables (CPT) associated with each node in the network. A CPT is similar to a truth table and shows the conditional probability of the node with respect to every state of its parent nodes. In order to complete the CPT for each node, we retrieved auction data for the query “laptop.” Auction data for a total of 109 auctions with 1127 bidders was accumulated. With the help of an existing shill detection tool [8], we created a training data set by identifying shill suspects (bidders or sellers) and verifying them manually through investigation of their profiles as well as their shill patterns. In the following, we provide a few examples of shill bidding patterns that were useful for creating the training data set.

**High and Irregular Incremental Bids:** Shill bids have a tendency to be relatively high as compared to other bids placed by normal bidders. A normal bid usually has an increment of \$1 to \$5, or \$10 to \$50 for high-end auctions. If a bidder’s bid increment is very high and irregular, the bidder is likely a shill.

**Early and Middle Stage Bidding:** Most shills only bid in the initial and middle stages of an auction, but stop bidding in the final stage to avoid winning the auction. On the other hand, an active bidder in the final stage of an auction is not likely a shill.

**Successive Bidding:** A normal bidder typically does not outbid himself. Therefore, any successive bidding activities that outbid oneself can be considered as a good indicator of shilling behavior.

**Total Increase in Price:** An interesting observation is the total increase in the price of an item due to a single bidder or multiple bidders who collude to raise the auction price. Thus, it is a good clue to shilling.

Once the training data set is ready, we can create a CPT for each node in both a bi-state BN and a multi-state BN. Note that to create CPTs for node N10 and N12, we manually identified bidders and sellers with high evidence level based on their suspiciousness. We also manually identified bidders and sellers with strong interaction strength for creating CPT for node N11. Table 2 shows part of the CPT for node N10 in the bi-state BN. For example, the second row of Table 2 tells that when the bidder’s feedback score is *low* (i.e., *Turquoise* or below, for N1), and the number of unique sellers is also *low* (i.e., less than 10, for N2), and the bidder is a new user (N3), and the bidder is not a verified user (N4), then the conditional probability for N10 being *high* (i.e., high bidder evidence level) is 0.83. This is reasonable because an unverified new user with a low number of unique sellers and low feedback score is likely a shill (if the bidder has already been detected as a shill suspect). As another example, since there are no training examples in the category “N1 = H, N2 = L, N3 = T, N4 = F” (i.e., the third row in Table 2), the conditional probability for N10 being *high* is set as 0.00. This setting will have no impacts on shill verification results because a new user is not likely to have a high feedback score.

Table 2. Part of the CPT for node N10 in bi-state BN

VeriU (N4)	NewU (N3)	Uniq (N2)	B_FB (N1)	P(N10 = H)
F	T	L	L	0.83
F	T	L	H	0.00
F	T	H	L	0.15
F	T	H	H	0.00
F	F	L	L	0.86
F	F	L	H	0.15
F	F	H	L	0.30
F	F	H	H	0.10

Similarly, Table 3 shows part of the CPT for the node N10 in the multi-state BN. The CPT for node N10 in the multi-state BN provides a conditional probability for more specific state values of its parent nodes. For example, the second row in Table 3 indicates that when the bidder’s feedback score is *None* (i.e., no feedback star

for N1), and the number of unique sellers is 1 (N2), and the bidder is not a new user (N3), and the bidder is not a verified user (N4), then the conditional probability for N10 being *high* is 0.93. This is also reasonable because an unverified experienced user with very low feedback score, who only traded with the same seller of the auction in the past 30 days, is very likely a shill (again, if the bidder has already been detected as a shill suspect).

Table 3. Part of the CPT for node N10 in multi-state BN

VeriU (N4)	NewU (N3)	Uniq (N2)	B_FB (N1)	P(N10 = H)
F	F	1	None	0.93
F	F	1	Yellow	0.75
F	F	1	Blue & Torq	0.72
F	F	1	Other	0.10
F	F	<=5	None	0.89
F	F	<=5	Yellow	0.87
F	F	<=5	Blue & Torq	0.87
F	F	<=5	Other	0.10
F	F	<=15	None	0.30
F	F	<=15	Yellow	0.25
F	F	<=15	Blue & Torq	0.10
F	F	<=15	Other	0.10
F	F	Other	None	0.00
F	F	Other	Yellow	0.00
F	F	Other	Blue & Torq	0.00
F	F	Other	Other	0.00

#### 4. Case Study

To illustrate the effectiveness of our proposed approach, we developed a Bayesian network toolkit (a snapshot of the toolkit for a multi-state BN is shown in Figure 3). The inputs of the toolkit are the BN for a particular auctioned item, and an auction with the same type of auctioned item, along with the user information for all involved bidders and seller. In this study, the auctioned item is of type “laptop,” and the auction under investigation is “HP HDX 16t notebook” held from Oct-09-09 to Oct-16-09, as shown in Table 4.

The conditional probabilities for query variables *Shill* and *Normal*, given evidence  $\mathbf{e}$ , can be calculated using Equations (2) and (3), respectively, which are derived from Equation (1) defined in Section 3.1.

$$\mathbf{P}(Shill|\mathbf{e}) = \alpha \mathbf{P}(Shill, \mathbf{e}) = \alpha \sum_y \mathbf{P}(Shill, \mathbf{e}, y) \quad (2)$$

$$\mathbf{P}(Normal|\mathbf{e}) = \alpha \mathbf{P}(Normal, \mathbf{e}) = \alpha \sum_y \mathbf{P}(Normal, \mathbf{e}, y) \quad (3)$$

In the above equations, *Shill* is the query variable N13, *Normal* is the query variable N14,  $\mathbf{e}$  is the observed values for the set of evidence variables  $\mathbf{E} = \{N1, N2, \dots, N9\}$ , and  $\mathbf{y}$  is the values for the set of unobserved variables  $\mathbf{Y} = \{N10, N11, N12, N14\}$  in Equation (2), and  $\mathbf{Y} = \{N10, N11, N12, N13\}$  in Equation (3), respectively.

Table 4. HP HDX 16t notebook bidding history

Bidder (FB)	Bid Amount	Bid Time	Bidder (FB)	Bid Amount	Bid Time
3***3 (27)	US \$630.00	Oct-16-09 08:19:12 PDT	9***a (1)	US \$200.00	Oct-10-09 12:40:10 PDT
p***t (299)	US \$621.00	Oct-15-09 18:34:38 PDT	g***e (245)	US \$112.50 <sup>§</sup>	Oct-09-09 15:49:21 PDT
p***t (299)	US \$611.00 <sup>§</sup>	Oct-15-09 18:34:38 PDT	9***a (1)	US \$110.00	Oct-10-09 12:39:56 PDT
g***e (245)	US \$601.00	Oct-14-09 16:59:59 PDT	g***e (245)	US \$102.50 <sup>§</sup>	Oct-09-09 15:49:21 PDT
p***t (299)	US \$601.00	Oct-15-09 17:54:10 PDT	9***a (1)	US \$100.00	Oct-10-09 12:39:43 PDT
g***e (245)	US \$561.00 <sup>§</sup>	Oct-14-09 16:59:59 PDT	g***e (245)	US \$61.00 <sup>§</sup>	Oct-09-09 15:49:21 PDT
p***t (299)	US \$551.00	Oct-12-09 17:09:44 PDT	9***a (1)	US \$60.00	Oct-10-09 12:39:27 PDT
8***I (29)	US \$550.00	Oct-14-09 12:31:54 PDT	g***e (245)	US \$51.00 <sup>§</sup>	Oct-09-09 15:49:21 PDT
p***t (299)	US \$510.00 <sup>§</sup>	Oct-12-09 17:09:44 PDT	9***a (1)	US \$50.00	Oct-10-09 12:39:04 PDT
g***e (245)	US \$500.00	Oct-09-09 15:49:21 PDT	g***e (245)	US \$31.00 <sup>§</sup>	Oct-09-09 15:49:21 PDT
m***s (4)	US \$500.00	Oct-10-09 16:49:22 PDT	a***a (21)	US \$30.00	Oct-09-09 13:58:29 PDT
g***e (245)	US \$405.00 <sup>§</sup>	Oct-09-09 15:49:21 PDT	g***e (245)	US \$26.00 <sup>§</sup>	Oct-09-09 15:49:21 PDT
m***s (4)	US \$400.00	Oct-10-09 16:49:00 PDT	a***a (21)	US \$25.00	Oct-09-09 13:58:02 PDT
g***e (245)	US \$202.50 <sup>§</sup>	Oct-09-09 15:49:21 PDT	a***a (21)	US \$0.99 <sup>§</sup>	Oct-09-09 13:58:02 PDT
<sup>§</sup> Automatic bid using eBay proxy bidding system.			Starting Price	US \$0.99	Oct-09-09 08:20:59 PDT

In order to determine whether a shill suspect is an actual shill, we first define thresholds based on our experience for probabilities of being a shill and a normal bidder. Although the thresholds are subjectively defined, they are sufficient for our case study and can be improved later as we gain more experience with shill verification. The current thresholds are defined as follows:

$$P(\text{shill}|\mathbf{e}) \geq 0.8 \text{ and } P(\text{normal}|\mathbf{e}) < 0.30 \Rightarrow \text{Shill}$$

Now from the bidding history listed in Table 4, we detected three shill suspects, namely  $g^{***e}$ ,  $m^{***s}$ , and  $9^{***a}$ , all of which demonstrated some shill patterns. For example, bidder  $m^{***s}$  has very high and irregular incremental bids, and only placed bids in the middle stage of the auction. The evidence we collected for the three shill suspects is listed in Table 5.

Table 5. Evidence of three shill suspects

Node	Bidder		
	$g^{***e}$	$m^{***s}$	$9^{***a}$
N1	Turquoise	None	None
N2	14	1	2
N3	False	False	False
N4	False	False	False
N5	2	1	4
N6	6	100	45
N7	Yellow	Yellow	Yellow
N8	None	None	None
N9	Private	Private	Private

Based on the evidence data and the BN we developed for auctioned items of type “laptop,” we used our Bayesian network toolkit to calculate  $P(\text{shill}|\mathbf{e})$  and  $P(\text{normal}|\mathbf{e})$  for the three shill suspects. The results for both the bi-state BN and the multi-state BN are listed in Table 6. From Table 6, we can see that using the bi-state BN, no shill suspects can be confirmed as actual shills, although bidder  $m^{***s}$  had demonstrated very obvious shilling behaviors. In contrast, when using the multi-state BN, bidder  $m^{***s}$  can be successfully identified as an actual shill. Thus, the experimental results of the case study conform to our expectation that a multi-state BN performs better than a bi-state BN.

Table 6. Probability of being a shill or a normal bidder

Probability of Being Shill / Normal Bidder		Bidder		
		$g^{***e}$	$m^{***s}$	$9^{***a}$
Bi-State	$P(\text{shill} \mathbf{e})$	0.4008	0.6115	0.6391
	$P(\text{normal} \mathbf{e})$	0.5687	0.3052	0.2590
Multi-State	$P(\text{shill} \mathbf{e})$	0.3449	<u>0.8086</u>	0.7234
	$P(\text{normal} \mathbf{e})$	0.7456	<u>0.2607</u>	0.2454

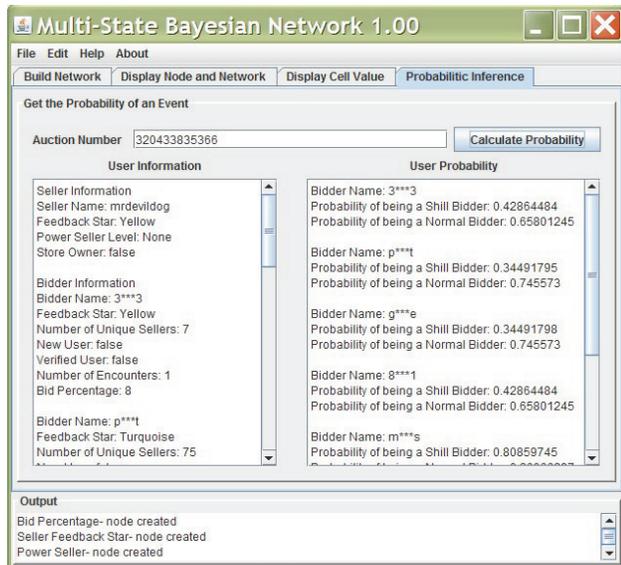


Figure 3. Tool support for multi-state BN

In our case study, we also calculated  $P(\text{shill}|\mathbf{e})$  and  $P(\text{normal}|\mathbf{e})$  for other bidders in the auction (as shown in Figure 3). Our experiments show that no other bidders satisfy our defined requirements to be shill, as is expected. For example, bidder 3\*\*\*3 is the winner of the auction, and is thus not likely to be a shill. This fact is consistent with the results demonstrated in Figure 3, where  $P(\text{shill}|\mathbf{e}) = 0.4286$  and  $P(\text{normal}|\mathbf{e}) = 0.6580$ .

## 5. Conclusions and Future Work

Shill detection and verification are an imprecise art riddled with uncertainties. However, using a probabilistic inference system such as a BN, we can account for these uncertainties with some degree of belief and reach a decision. The BN gives us results for decision making based on prior knowledge about the domain and thus provides some level of quantification. The biggest dilemma we have when implicating a bidder is whether the bidder was simply “in the wrong place at the wrong time.” With additional evidence for suspected bidders, our multi-state BN helps us resolve this uncertainty when implicating a bidder and the seller of the auction.

We also see from the results that a multi-state BN gives more accurate results as compared to a bi-state BN. This follows the fact that using multi-state BN, we can classify shills in fine-grained categories, allowing us to draw more precise conclusions.

For future work, we will implement a clustering algorithm like  $k$ -means clustering [14], which can help cluster large chunks of data and provide a clearer picture of the knowledge domain. This will help in the discovery of any hidden node states that can give a better result for identifying shills in online auctions. Furthermore, we also plan to incorporate our Bayesian network toolkit with an agent-based trustworthy online auction system we previously developed [15]. We believe once the above features are implemented, we will have a full-fledged, reliable verification engine, which can form the core of a trustworthy agent-based online auction system.

## References

- [1] B. K. Bhargava, M. Jenamani, and Y. H. Zhong, “Counteracting Shill Bidding in Online English Auction,” *International Journal of Cooperative Information Systems*, Vol. 14, No. 2-3, 2005, pp. 245-263.
- [2] D. H. Chau, S. Pandit, and C. Faloutsos, “Detecting Fraudulent Personalities in Networks of Online Auctioneers,” *Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Database (PKDD 2006)*, Berlin, Germany, September 2006, pp. 103-114.
- [3] F. Dong, S. M. Shatz, and H. Xu, “Combating Online In-Auction Fraud: Clues, Techniques and Challenges,” *Computer Science Review (CSR)*, Vol. 3, No. 4, November 2009, pp. 245-258.
- [4] H. Xu and Y.-T. Cheng, “Model Checking Bidding Behaviors in Internet Concurrent Auctions,” *International Journal of Computer Systems Science & Engineering*, 2007, Vol. 22, No. 4, pp. 179-191.
- [5] H. S. Shah, N. R. Joshi, A. Sureka, and P. R. Wurman, “Mining eBay: Bidding Strategies and Shill Detection,” *Proceedings of the 4th International Workshop on Mining Web Data for Discovering Usage Patterns and Profiles*, July 2003.
- [6] J. Trevathan and W. Read, “Detecting Shill Bidding in Online English Auctions,” *Handbook of Research on Social and Organizational Liabilities in Information Security*, M. Gupta and R. Sharman (eds.), Information Science Reference, 2009, pp. 446-470.
- [7] S. Pandit, D.H. Chau, S. Wang, and C. Faloutsos, “Netprobe: a Fast and Scalable System for Fraud Detection in Online Auction Networks,” *Proceedings of World Wide Web*, 2007, pp. 201-210.
- [8] H. Xu, C. K. Bates, and S. M. Shatz, “Real-Time Model Checking for Shill Detection in Live Online Auctions,” *Proceedings of the International Conference on Software Engineering Research and Practice (SERP'09)*, July 13-16, 2009, Las Vegas, Nevada, USA, pp. 134-140.
- [9] R. J. Kauffman and C. A. Wood, “Running up the Bid: Detecting, Predicting, and Preventing Reserve Price Shilling in Online Auctions,” *Proceedings of the 5th International Conference on Electronic Commerce*, 2003, Pittsburgh, PA, pp. 259-265.
- [10] D. Chau and C. Faloutsos, “Fraud Detection in Electronic Auction,” *Proceedings of European Web Mining Forum (EWMF 2005)*, ECML/PKDD 2005, October 2005, Porto, Portugal.
- [11] F. Dong, S. M. Shatz, and H. Xu, “Reasoning Under Uncertainty for Shill Detection in Online Auctions Using Dempster-Shafer Theory,” To appear in *International Journal of Software Engineering and Knowledge Engineering (IJSEKE)*, Vol. 20, No. 7, November 2010.
- [12] S. Russel and P. Norvig, *Artificial Intelligence: a Modern Approach*, Second Edition, Prentice Hall, 2005.
- [13] eBay, *Trading Web Services – eBay Developers Program*, eBay Inc., Retrieved on September 18, 2009 from <http://developer.ebay.com/products/trading/>
- [14] D. J. Hand, H. Mannila, and P. Smyth, *Principles of Data Mining*, The MIT Press, Cambridge, Massachusetts, 2001.
- [15] R. Patel, H. Xu, and A. Goel, “Real-Time Trust Management in Agent Based Online Auction Systems,” *Proceedings of the 19th International Conference on Software Engineering and Knowledge Engineering (SEKE 2007)*, July 9-11, 2007, Boston, USA, pp. 244-250.

# An Empirical Evaluation on the Relationship Between Final Auction Price and Shilling Activity in Online Auctions\*

<sup>1</sup>Fei Dong, <sup>2</sup>Sol M. Shatz and <sup>3</sup>Haiping Xu

<sup>1,2</sup>Computer Science Department

University of Illinois at Chicago, Chicago, IL 60607, USA

{fdong, shatz}@cs.uic.edu

<sup>3</sup>Computer and Information Science Department

University of Massachusetts Dartmouth, North Dartmouth, MA 02747, USA

h xu@umassd.edu

**Abstract.** *In this paper, we are interested in the relationship between final prices of online auctions and possible shill activities during those auctions. We conduct experiments on real auction data from eBay to exam the hypotheses that (1) A lower-than-expected final auction price indicates that shill bidding was less likely to occur in that auction; and (2) A higher-than-expected auction final price indicates possible shill bidding. In the experiments, a neural network approach is used to learn the expected auction price. In particular, we trained the Large Memory Storage and Retrieval (LAMSTAR) Neural Network based on features extracted from item descriptions, listings and other auction features. The likelihood of shill bidding is determined by a previously proposed Dempster-Shafer theory based shill certification technique. The experimental results imply that both a lower-than-expected final auction price and a higher-than-expected final auction price might be used as direct evidence to distinguish trustworthy auctions from likely shill-infected auctions, allowing for more focused evaluation of those shill-suspected auctions.*

## 1. Introduction

Online auction platforms, such as eBay, have become ideal places for people to purchase bargain-priced and hard-to-find items. Besides, they also allow people to become instant businessmen at a low cost. However, frauds develop in online auctions as online auction platforms expand in use. Shill bidding, which is one type of auction fraud, aims to inflate the final price of an auction. Such bidding activity has been found to have a severe impact on the fairness of auction markets, and in the worst scenario it can result in insufficient market or even market failure [1].

Many online consumers do not realize that shill bidding is a serious illegal behavior. In fact, if convicted, a shill bidder can serve several years in prison and pay a significant fine [2]. In 2004, an eBay store owner who conducted shill bidding pleaded guilty to Combination in Restraint of Trade, a violation of the New York antitrust law. It is a felony punishable by a maximum of four years in prison [2].

To protect online auction bidders from shill bidding, shills, who intentionally drive up an auction's final price, should be detected as they are placing the shill bids so that the auctions can be cancelled and bidders are protected from fraudulent activities. However, it is not easy to detect shills due to their concealment efforts and the lack of non-deniable evidence. Yet, if we can easily and quickly distinguish auctions that are suspected of involving shills from auctions without shills, the process of shill investigation may become somewhat less difficult since efforts can then be focused on just those auctions with potential shills. In this paper, we propose a method to classify auctions into two such categories – those that are likely to involve shills and those that are not likely to involve shills. The primary classification feature we consider is the actual final price of the auction (in comparison to an expected final price).

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 describes the hypotheses we evaluate and why we formed these hypotheses. Section 4 introduces how the experimental data is collected and how the desired features of the data can be defined. Section 5 presents our experimental procedures and reports the results. Finally, Section 6 provides conclusions and mentions some future work.

## 2. Related Work

Generally, a substantial amount of work has been done in the study of auction data. Heijst et al., combined text mining and boosting algorithms to predict auction final prices [3]. Ghani et al., compared a regression model,

---

\* This material is based upon work supported by the U.S. National Science Foundation under grant numbers CNS-0715648 and CNS-0715657.

a neural network and a decision tree, and they achieved the best result using the neural network when treating the price prediction problem as a series of binary classification problems [4]. In [5], Lim et al., employed grey system theory to predict auction closing prices in a simulated auction environment. In this paper, we use a neural network approach, but for the special purpose of predicting shilling activities. In particular, the expected auction price is learned from the LARge Memory Storage and Retrieval (LAMSTAR) Neural Network, where price prediction is based on features extracted from item descriptions, listings and bids features.

The topics of shill detection and verification also have attracted significant attention from researchers. Patel et al., proposed a real-time shill monitor for agent based online auction systems using role-based access control mechanisms [6]. Xu, et al., introduced a formal model checking approach for detecting shilling behaviors, especially the competitive shilling behaviors [7]. Dong et al., proposed a decision support system based on Dempster-Shafer (D-S) theory [8] to compute the likelihood of shill bidding activities [9]. Kauffman and Wood discovered that the existence of shill bids in an online auction can drive up the final selling price of the auction [10]. In contrast, in this paper we are interested in some different questions: If an auction finishes at a price that exceeds the expected closing price range, can we consider it to be likely that the auction involves shill bidding? And similarly, if an auction finishes at a price that is much lower than the expected closing range, is it likely that the auction does not involve any shill bidding?

### 3. The Hypotheses

Since a shill bidder's primary goal is to drive up the final price, it seems reasonable that for a shill-infected auction, the final auction price should be significantly higher than it would have been if no shill bids were placed in the auction. Therefore, if the final price of an auction can be predicted accurately (assuming there are no shill bids), the actual final price of the auction would be very useful in deciding if this auction is likely to involve shills. This would provide direct evidence to both shill investigators and other bidders.

Kauffman and Wood found that shill bidding acts as a signal for other bidders to place higher bids and thus increase the auction's winning bid [10]. In other words, if shill bidding occurs in an auction, the auction will very likely have a higher final auction price. Equivalently, we suggest the following linguistic, fuzzy-logic type expression:

*Shill bidding is highly likely to result in a higher-than-expected final auction price.*

According to the *modus tollens* rule in classic logic, we obtain our first actual hypothesis.

- **Hypothesis 1 (The non-shill hypothesis):** A lower-than-expected final auction price indicates unlikely shill bidding.

Auction prices can reflect the current market of the auctioned item. If a final auction price falls in the normal price range – for example within +/-20% of the average price – the price of the auction conforms to market discipline. In contrast, if the market for a type of item is depressed, the final auction prices for these items are not expected to be high. Under this circumstance, if the final prices of a particular seller's auctions are significantly and consistently higher than those of the same-item auctions, the seller is suspected of employing shill bidding or other types of fraudulent bidding activities.

Therefore, the final price of an auction is believed to be an indicator of trustworthiness.

*An auction that does not involve shill bidding is likely to have a lower-than-expected, or as-expected, final auction price.*

Again, according to the *modus tollens* rule, we obtain our second hypothesis:

- **Hypothesis 2 (The shill hypothesis):** A higher-than-expected auction final price indicates possible shill bidding.

### 4. Data Collection

EBay provides listings for a broad range of auctioned items. It also makes available detailed information of each auction and some limited information on sellers and bidders. Because there is no available public auction database, we designed a software agent to collect auction data from eBay. Given some search criteria, the agent is able to retrieve specified data for completed auctions, and store the data on local disks.

We designed the data collecting agent as shown in Figure 1. A central server first obtains the completed auctions' URLs from auction listing pages according to the search criteria. Then the server establishes and maintains a global queue that can be accessed by crawlers to keep track of the URLs. Following the given URLs, the crawlers sequentially scan HTML tags to extract the desired data. The collected data are then stored in a database. An advantage of this design is that multiple crawlers work collaboratively to create an efficient data collecting agent. In addition, the task queue can be used to avoid duplicated data collection.

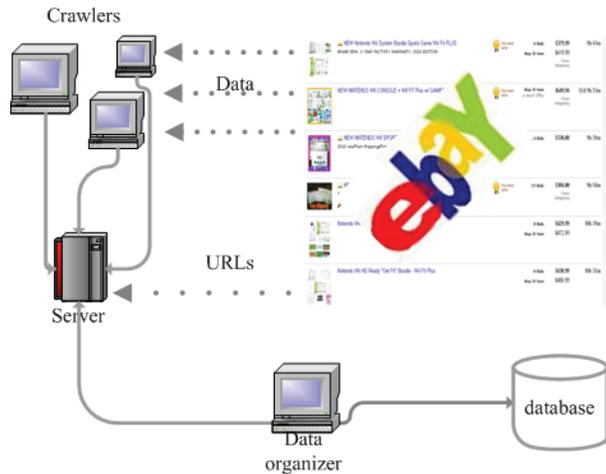


Figure 1. Data collecting agent

The data used in this project is under the category of Nintendo Wii game console and systems. Note that although the broad categorization of the data is Wii game console system, the items bundled with the game systems vary from auction to auction. For example, the items for sale in one of the auctions include a Wii game console and a new Wii FIT, which is an accessory of Wii game system; while in another auction the auctioned item is a bundle of Nintendo Wii System Console, Steering Wheel and 13 Games. Therefore, the category of Wii console system is still a broad category that contains many types of items. This partially explains why the prices of this category cover a wide range. The final price distribution for the data used in this study is shown in Figure 2.

For each auction, we collected data that is filled in by the seller when listing an item for auction, including information about the seller, details of the item (name, specifications, description, photos, etc.) and attributes about the auction (length, starting bid, reserve price, shipping charges, etc.). The data is processed to extract attributes and create new attributes that are then used to predict the final price for that auction. The data features classified in 4 different groups are listed in Table 1.

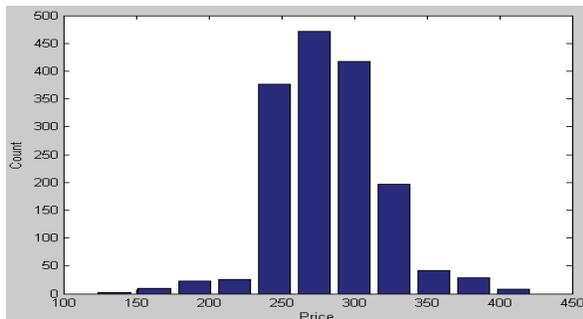


Figure 2. Final price distribution in the study

Table 1. Data features in 4 different groups

Group Name	Features	
Item	Condition (new, used, refurbished, unspecified)	Shipping cost (\$)
	Number of pictures	Description
Seller	Reputation percentage (%)	Reputation score
	Has web-store (Yes/No)	TopRatedseller (Yes/No)
Bid Details	Number of bids	Number of bidders
	Duration	Starting time
	Starting bid (\$)	End Time
	Month	Day
Category-Specific	Fit (Yes/No)	Game (number)
	Bundle (Yes/No)	Wheel (Yes/No)

## 5. Experiments

In this section, we report experiments we conducted to study the two hypotheses proposed in Section 3.

### 5.1 Overview

First, we built and trained a neural network to predict auction prices (the neural network based model is introduced in Section 5.2). Once the neural network based price predictor achieved good performance, we employed the price predictor to predict the final prices of new auctions that were not used in the training and testing phases. Since the price predictor can achieve a relatively high accuracy, we consider the predicted prices as “expected” prices. We selected 30 auctions whose predicted prices were higher than their actual prices and another 30 auctions whose predicted prices were lower than their actual final prices. The two groups of data were used to test Hypothesis 1 and Hypothesis 2, respectively. Next, we computed a skill score, as described in [9], for each of the 60 auctions. The skill score is the highest belief of shilling behavior among all bidders in an auction. The skill score is a number between 0 and 1 (inclusive) that indicates the likelihood of an auction involving shills. The skill score and belief of shill are defined in Section 5.3. In brief, the predicted price is compared with the actual final auction price. If the actual price is higher than the expected price, and the skill score for the auction is higher than or equal to 0.9, the shill hypothesis is verified. Otherwise, if the skill score is lower than 0.9, the shill hypothesis is not verified. Similarly, if the expected price is higher than the actual price, and if the skill score for the auction is less than 0.5, the non-shill hypothesis is verified. Otherwise if the skill score is higher than 0.5, the non-shill hypothesis is not

verified. Note that the thresholds of 0.9 and 0.5 are subjective values defined in [9]; but may be adjusted later based on our further experience. In the following sections, we explain further the process for obtaining expected prices and the method for computing skill scores.

## 5.2 Price Prediction

We built a price predictor based on the Large Scale Memory Storage and Retrieval (LAMSTAR) network [11]. The LAMSTAR, which combines Self Organization Map (SOM) and statistical decision tools, has been successfully applied to diagnosis, prediction and detection type of applications [12]. The trained network for predicting auction final prices is shown in Figure 3.

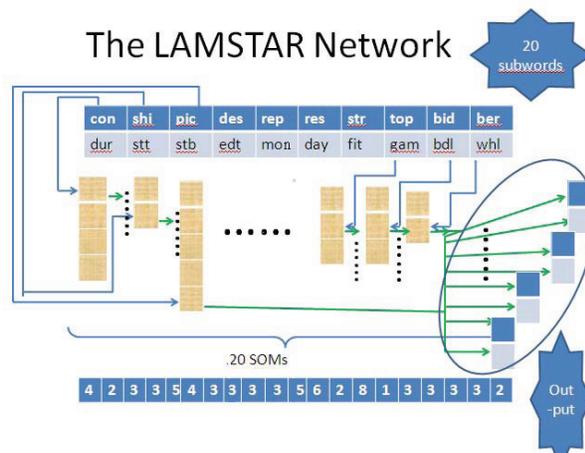


Figure 3. The network for predicting auction price

The grids on the top of the figure are the subwords, representing features from Table 1. Each of them is abbreviated by three letters. These features are preprocessed and then provided as inputs to the neural network. For every subword there is an associated SOM module (in the middle of the figure) that is used to store and retrieve information in the training process. For each subword, a winning neuron in the associated SOM module is determined based on the similarity between the input and a storage-weight vector (stored information).

In the middle of the figure, the arrows between SOM modules encode the correlations between them. The link-weights between different SOM modules and the link-weights from the SOM modules to the output decision layer are continuously trained during normal operational runs. They are adjusted on a reward/punishment principle. Specifically, for the weights of links to the output layer, if the output of the particular output neuron is desired, the link weight for that neuron is rewarded by a small non-zero increment, while if the output is not desired, the link weight is punished by a small non-zero decrement. The link-weights between SOMs can be trained in a similar way.

The grids on the rightmost part of the figure (pointed to by arrows and circled in an oval) are the output decision layer of the network. The network is designed with multiple output layers and each layer consists of two neurons, so each layer represents a binary classifier: whether the final auction price is within certain  $\$X$  range or not. The value of  $X$  in this study is set as 50. In other words, the price predictor is trained to predict if an auction’s final price is in a  $\$50$  range or not, such as  $(\$185, \$235]$ , rather than a specific numeric value. Since the minimum price in the collected data set is  $\$135$  and the maximum price is around  $\$410$ , there are 6 output layers in the neural network.

Because the actual final price is a specific number, while the expected price is defined using a range, to compare these prices, the actual price is compared to the average price of the expected range. For example, if a predicted price falls in the range  $(\$185, \$235]$ ,  $\$210$  is used as the comparator. In this study, we define “higher” as at least  $\$50$  more and “lower” as at least  $\$50$  less.

After training the neural network on 1000 auctions and testing it on another 600 auctions, the neural network achieved a precision as high as 95%. Thus, given an auction, the price predictor can determine with a small chance of error if the final auction price will fall in a range span of  $\$50$ . Note that in this experimental study, we only predict the final price of one specific category of items. We leave the work of predicting the final price of general items as future work.

## 5.3 Skill Analysis

In [9], a skill certification method based on the mathematical theory of evidence, Dempster-Shafer (D-S) theory was introduced. Six bid-level properties and two auction-level properties are quantified to compute the *belief of skill* for every bidder in an auction. The bid-level properties include the time of a bidder’s last bid in an auction, the bidder’s concurrent bidding activities, the bidder’s reputation score, the bidder’s average bid increment, the bidder’s winning ratio, and a bidder’s affinity for the seller. The auction-level properties include the number of bids and the starting price of the auction.

The D-S theory considers a universe of discourse  $\Theta$  (also called frame of discernment) that consists of a finite set of mutually exclusive atomic states in a problem domain [8]. For example, in the auction skill detection domain, the frame of discernment for a bidder is  $\Theta = \{\text{skill}, \sim\text{skill}\}$ . The power set  $2^\Theta$ , which is the set of all possible subsets of  $\Theta$  including the empty set, can be denoted as  $2^\Theta = \{\emptyset, \{\text{skill}\}, \{\sim\text{skill}\}, \Theta\}$ . The D-S theory assigns a belief mass to each subset of the power set by function  $m: 2^\Theta \rightarrow [0,1]$ . The function is called basic mass assignment (BMA) if it satisfies the following two equations:

$$\sum_{A \in 2^\Theta} m(A) = 1 \quad (1)$$

$$m(\emptyset) = 0 \quad (2)$$

Given a certain piece of evidence,  $m(A)$  represents one's belief exactly on state  $A$ , not any subset of  $A$ . The empty set  $\emptyset$  represents a contradiction, which cannot be true in any state. Therefore, the BMA for  $\emptyset$  is assigned 0. The basic mass assignment  $m(\Theta)$  can be interpreted as the total ignorance of the problem domain, where one feels uncertain about the truth because every state is present. For the skill detection problem, Eq. (1) and Eq. (2) imply that  $m(\text{skill}) + m(\sim\text{skill}) + m(\Theta) = 1$ .

To obtain the overall belief on state  $A$ , one must take the sum of beliefs on all subsets of  $A$ . As defined in Eq. (3), a belief function is defined as the mass sum of all  $B$ s, which are subsets of  $A$ . Thus, in D-S theory, a degree of belief is represented as a belief function rather than a Bayesian probability function, and mass values are assigned to sets of elements rather than singletons.

$$\text{bel}(A) = \sum_{B \subset A} m(B) \quad (3)$$

Based on Dempster's rule of combination, the formula for computing the belief of skill is as follows:

$$\text{belief}(\text{skill}_i) = m(\text{skill}_i) \quad (4)$$

$$m(\text{skill}_i) = m_1(\text{skill}_i) \oplus m_2(\text{skill}_i) \oplus \dots \oplus m_n(\text{skill}_i) \quad (5)$$

The skill certification approach is demonstrated to be effective and accurate. In the experiments, we compute the skill score for every bidder in an auction. The skill score for an auction is defined as the highest skill score among the bidders. The auction is suspected to involve skill bidding when the skill score for the auction is higher than 0.9; while the auction is considered to be free of skills when the skills score for the auction is less than 0.5.

## 5.4 Results and Discussion

After building the price predictor, we ran the price predictor on a distinct collection of data. As mentioned in Section 5.1, we selected 30 auctions that have predicted prices higher than actual prices; we name this group of data as Higher Group. We also selected another 30 auctions that have predicted prices lower than actual prices, and name this as Lower Group. According to the hypotheses proposed in Section 3, the skill scores of most of the auctions in Higher Group should be under 0.5 and the skill scores of most of the auctions in Lower Group should be above 0.9.

The actual prices, predicted prices, and the skill scores for Higher Group and Lower Group are shown in Figure 4 and Figure 5, respectively. In order to make skill scores visible in the figures with a Y-axis scaled from [0, 400], the skill scores are multiplied by 400 in both figures. To enhance visibility, points are connected by lines.

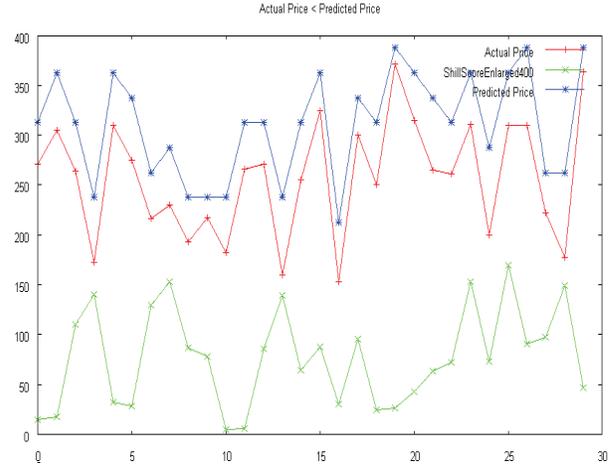


Figure 4. Analysis results of Higher Group auctions

From Figure 4, we can see that when the actual prices are less than the predicted prices, the skill scores for all 30 auctions are under 200, which means that each of them is smaller than 0.5 before being scaled by 400. Hence in accordance with the skill certification rules, no auctions in this group involved skill bidding activities. Therefore, the experimental results support the non-skill hypothesis (Hypothesis 1).

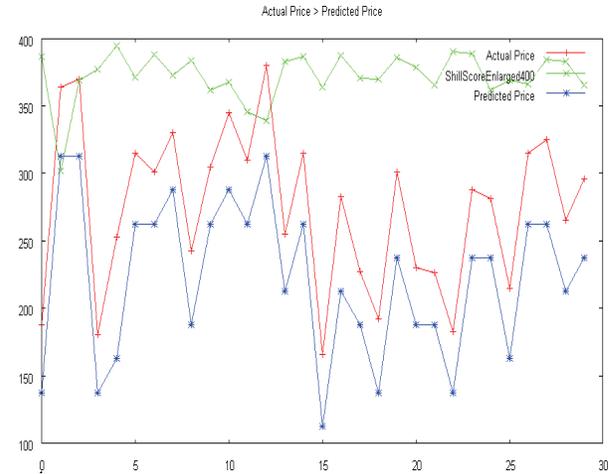


Figure 5. Analysis results of Lower Group auctions

From Figure 5 we can see that when the actual prices are higher than the predicted prices, most skill scores are beyond 360 (i.e., bigger than 0.9 before being scaled). However, there are several exceptions: 3 out of 30 skill scores are smaller than 0.9 but still much higher than the non-skill threshold, 0.5. As discussed in [9], a bidder with a skill score between 0.5 and 0.9 implies the bidder

is suspicious rather than innocent. Thus, the shill hypothesis (Hypothesis 2) is also supported by the data.

The experimental results provide us two very important pieces of evidence for shill bidding investigation: 1) if an auction has a higher-than-expected final price, the auction is likely to involve shills, and 2) if an auction has a lower-than-expected final price, the auction is unlikely to involve shills. Note that we did not explicitly consider “normal” (or as-expected) priced auctions, but we believe shilling activities are not likely in such cases as in the auctions with low-than-expected final auction prices.

The auctions with lower-than-expected prices can be eliminated from the set of auctions that need to be investigated. Moreover, the two pieces of evidence can be easily obtained. Both experienced investigators and ordinary bidders can use the price-based evidence as signals for determining the trustworthiness of a seller.

Note that in order to make our approach more practical, we need to consider the following three issues: First, since the market of an item may change with time, the price predictor should be trained periodically in order to make the expected price as accurate as possible. Second, in this paper, we only analyzed a small number of auction data in one category to verify the hypotheses; using more auction data from other categories may strengthen our results. Third, we should determine if the price of a general category of items can be predicted as accurately as that of the Wii gaming system studied in this paper.

## 6. Conclusions and Future Work

Shill bidding has been a serious issue faced by innocent bidders in online auctions for a long time. However, due to the characteristic of concealment of shill bidding activities, both bidders and investigators lack an easy yet effective way to evaluate the trustworthiness of a seller. Thus, there is a pressing need to explore a new simple method to investigate auction shills. In this paper, we first propose two hypotheses: (1) A lower-than-expected final auction price indicates that shill bidding is less likely to occur in the auction, and (2) A higher-than-expected auction final price indicates possible shill bidding. We then present experiments to test these hypotheses. The experimental results indicate that both hypotheses can be used to provide direct evidence in determining auctions with shill bids. The evidence can help people distinguish auctions with shills from auctions without shills, so effort can be focused on auctions with potential shills and thus save precious investigation time.

In our future work, we will look to improve the precision of the price predictor as well as the shill certification techniques. We will also strengthen the empirical study in terms of considering prediction-price intervals that reflect prediction errors, and performing

analysis aimed at uncovering threats to the validity of the findings. In addition, we will also consider studying other types of auctions with unexpected prices in order to widen the scope of our results.

**Acknowledgement.** We thank D. Graupe for his advice on designing and improving the precision of the LAMSTAR neural network.

## References

- [1] F. Dong, S. Shatz, and H. Xu, “Combating Online In-auction Fraud: Clues, Techniques and Challenges,” *Computer Science Review*, Vol. 3, No.4, 2009, pp. 245-258.
- [2] Consumer Affairs, “Shill Bidding Exposed in Online Auctions,” Retrieved on January 31, 2010 from <http://www.consumeraffairs.com/news04/shills.html>
- [3] D. V. Heijst, R. Potharst, and M. V. Wezel, “A Support System for Predicting eBay End Prices,” *Decision Support Systems*, Vol. 44, No. 4, 2007, pp. 970-982.
- [4] R. Ghani and H. Simmons, “Predicting the End-price of Online Auctions,” *Proc. of the International Workshop on Data Mining and Adaptive Modelling Methods for Economics and Management*, Pisa, Italy, 2004.
- [5] D. Lim, P. Anthony, and C. Ho, “Agent for Predicting Online Auction Closing Price in a Simulated Auction Environment,” *Proc. of Pacific Rim International Conference on Artificial Intelligence - PRICAI*, 2008, pp. 223-234.
- [6] R. Patel, H. Xu, and A. Goel, “Real-Time Trust Management in Agent Based Online Auction Systems,” *Proc. of the 19th International Conference on Software Engineering and Knowledge Engineering*, July 9-11, 2007, Boston, USA, pp. 244-250.
- [7] H. Xu and Y.-T. Cheng, “Model Checking Bidding Behaviors in Internet Concurrent Auctions,” *International Journal of Computer Systems Science & Engineering*, Vol. 22, No. 4, 2007, pp. 179-191.
- [8] G. Shafer, *A Mathematical Theory of Evidence*, Princeton University Press, 1976.
- [9] F. Dong, S. M. Shatz, and H. Xu, “Reasoning Under Uncertainty for Shill Detection in Online Auctions Using Dempster-Shafer Theory,” *To appear in Int. Journal of Software Engineering and Knowledge Engineering (IJSEKE)*, Vol. 20, No. 7, Nov. 2010.
- [10] R.J. Kauffman and C.A. Wood, “The Effects of Shilling on Final Bid Prices in Online Auctions,” *Electronic Commerce Research and Applications*, Vol. 4, No. 1, 2005, pp. 21-34.
- [11] D. Graupe and H. Kordylewski, “A Large-Memory Storage and Retrieval Neural Network for Adaptive Retrieval and Diagnosis,” *International Journal of Software Engineering and Knowledge Engineering*, Vol. 6, No.1, March 1998, pp. 115-138.
- [12] D. Graupe, *Principles of Artificial Neural Networks (2nd Edition)*, World Scientific Publishing Co. Pte. Ltd., Singapore, 2007.

# A Framework for Solar Energy Applications – Photovoltaic Systems

Papatella, F., Carvalho, T., Zarate, L., Pereira, E., Song, M.

Computer Science Department

Pontifícia Universidade Católica de Minas Gerais

Belo Horizonte – Brazil

{papatella, carvalho, zarate, song}@pucminas.br<sup>1</sup>

elizabeth.pereira@green.pucminas.br<sup>2</sup>

*Abstract*— The energy generation became an important aspect of modern life once the population is growing rapidly in the metropolitan regions. Pessimistic forecasts are growing in the global energy scenario demanding the use of renewable sources of energy, such as the solar one. But, in order to follow such demand new computer techniques have to be developed. This paper presents a framework to assist the developer in the project of new elements and simulation of solar applications. By applying the framework concepts, such as source code reuse, one can create a complete environment to evaluate solar energy data. This work focuses in the software development and tools to be used in solar energy generation process, such as the photovoltaic one.

## I. INTRODUCTION

The Brazilian institute (INPE) estimates an annual increase of 4% in electric energy demand for the next 20 years in Brazil [1]. This fact, and also the concern about environmental consequences due to the consumption of non-renewable fossil fuel, impels the researchers to develop alternative technologies to generate electric energy. The photovoltaic energy is an example.

But, in order to accomplish photovoltaic experiments, to report statistical results and to evaluate the efficiency of the solar energy conversion into electric energy new evaluative processes must be used.

There are many software devoted to solar energy application such as blast, doe (doeplus, visual doe), ener-win, energy plus, hot2000, passport and solar5. However, the costs for implantation, management and control are still far too expensive [2, 3].

Most software and tools do not attend the requirements demanded for different simulations of solar energy processes, even though the Software Engineering discipline provides paradigms to develop more flexible and reusable systems based on, for example, frameworks [4].

Frameworks are used in very different contexts such as graphical interfaces (Java Swing, AWT), net services (Java

RMI, Jeeves) and commercial works (IBM's San Francisco) [5].

In this work, a framework is developed in order to measure, simulate and test photovoltaic processes. The solution provides the reuse of software projects, analysis, and tests based on solar energy applications. It is up to the developer the reuse of the source code, which can be specialized according to the application.

It also provides many benefits such as the updating of climatic data, the creation and use of new elements, and the incorporation of new functionalities into existing applications. It is possible to manipulate data and formulas related to solar energy domain especially those ones already tested and validated.

This paper is divided into five sections. The Section 2 describes the framework. The Section 3 presents the photovoltaic model. In Section 4 a case study is presented. The Section 5 concludes this paper.

## II. THE FRAMEWORK

The main objective of any framework is to simplify the development of similar applications [6]. It can be used in order to encapsulate the business rules involved in the description of any new element some of which based on detailed descriptions of the previous ones.

The framework can carry dynamic simulations for renewable energies such as the solar one. In this work it has been generated an abstraction in order to describe photovoltaic systems - a method to convert solar energy into electric energy.

Any framework must have the capability to add new elements to its structure. It has to incorporate new functionalities or new elements to the set already developed.

In terms of solar systems it must have the ability, for example, to update climatic data. The framework is divided into four basic packages as depicted in Figure 1:

- the Form package which is responsible for supplying the interfaces,
- the Controller package to control the simulations,

- the DataBase package to provide access to the database, and
- the Energy package which initiates the framework and provides the programmer all the functionalities.

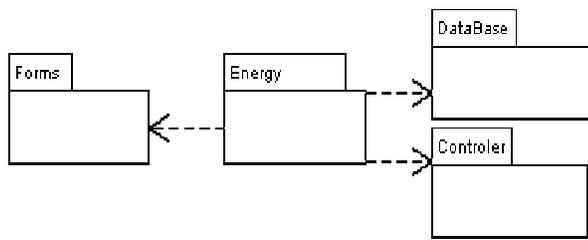


Fig. 1. The framework packages.

All solar energy system [7] simulations can be modeled according to some basic elements: the radiation, the collector and the container. All of them are encapsulated in one of the following modules:

- IRadiation – a module responsible for supplying the solar radiation data;
- ICollector – a module responsible for the conversion of solar energy into the electric one;
- IStore – a module responsible for storing energy;
- IManipulate – a module responsible for carrying all kinds of measurement.

The framework calls the solar radiation element to calculate the solar radiation data. The solar collector supplies information to the electric conversion process based on the radiation data.

Users do not need to be worried if the solar radiation is supplied monthly or hourly. The framework simply simulates the behavior of each element accordingly.

The framework is divided into three hierarchical layers (Figure 2). The first one corresponds to the interfaces and classes which are responsible for drawing elements (EMCObject, EMCPoint, and EMCPipeVertices classes). The second layer is responsible for guaranteeing that the simulation will run correctly. The last one adds to the simulation the elements already developed.

Each class has its own functionalities, e.g, the EMCPipe class deals with communications between each connection point.

All elements have their connection points and each connection point is defined in terms of EMCJoinPoint.

The framework organizes the elements as a directed graph where the main node is the solar radiation element and the others are interfaces such as ICollector, IManipulate or ICapture (Figure 3). All dependency is solved before running the simulation. The data flow is depicted in Figure 4:

- class Execute receives parameters from the elements involved in the process;

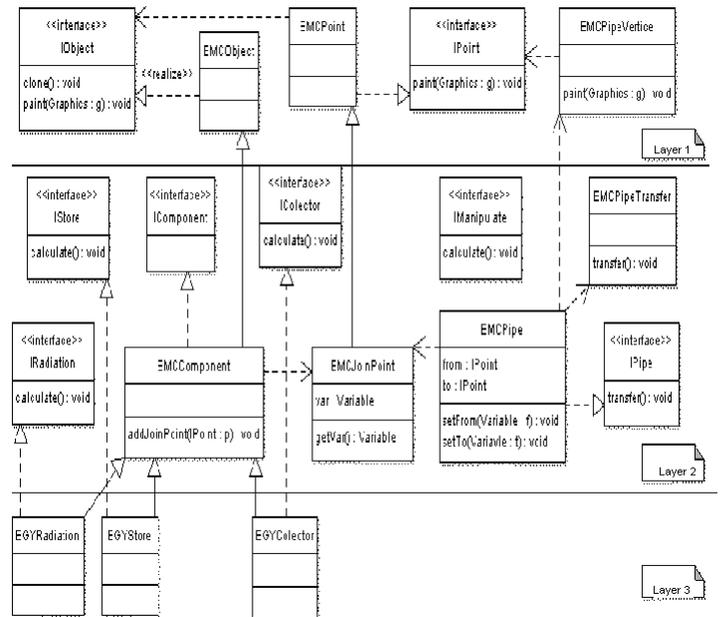


Fig 2. Main classes of the framework.

- the mountTree() method constructs the execution graph and the transfer() method links all the elements together.
- The calculate() method is called to run the simulation.

Basically every solar simulation process, thermal or photovoltaic, is based on the solar radiation. Sensors capture the energy, which is processed and distributed to the elements of the system.

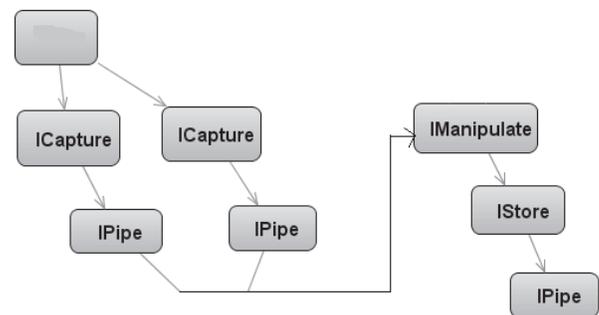


Fig. 3. Main classes used in simulations.

### III. PHOTOVOLTAIC SYSTEMS

This section presents the concepts and the mathematical models related to the solar energy conversion process into electric energy [8, 10]. The Solar Energy process identifies a year, named reference year, to estimate radiation on photovoltaic systems. The devices presented in the photovoltaic system are described as follows.

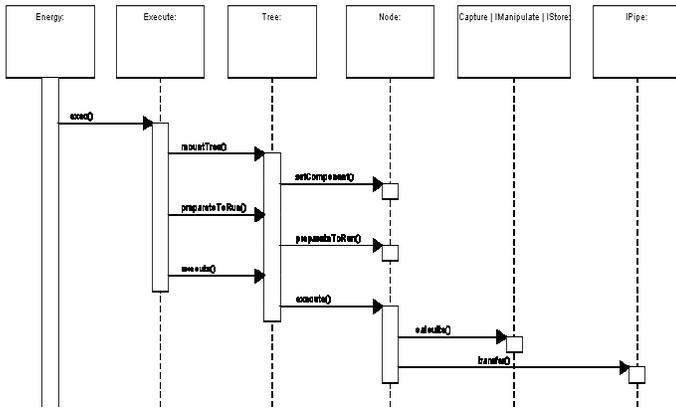


Fig. 4. Runtime scenario.

### A. The reference Year

The solar energy application performance is heavily dependent on climatic aspects. However, meteorological data analysis is computational intensive.

An alternative is to define meteorological variables representing a period of time in order to lower computer costs. The reference year represents a set of meteorological variables for a specific region during a year.

There are many methodologies to identify the reference year: the TRY (Test Reference Year), TMY (Typical Meteorological Year), DRY (Design Reference Year), and the SRY (Short Reference Year) [8, 9].

### B. Methods to Estimate Radiation

The equations to be presented describe models to estimate the radiation values on the terrestrial surface based on hourly, daily, and monthly climatic variations.

The Figure 5 shows the method based on historical measured data and on the identification of the presumable reference year. It is important to emphasize that the global radiation is calculated based on solar insolation.

### C. The Photovoltaic Panel

The panel is the photovoltaic system element responsible to convert the solar energy into electricity. This process is the result of energy being absorbed by the semiconductors elements of the solar module.

The curve that relates the voltage and the current, which is generated by photovoltaic modulus, is given by  $(I \times V)$ .

Information about voltage and current for the maximum power, the open-circuit voltage, the short-circuit current, the temperature coefficient for the short-circuit current, the referential solar irradiance and the temperature coefficient for the short-circuit voltage in the referential solar irradiance is also given.

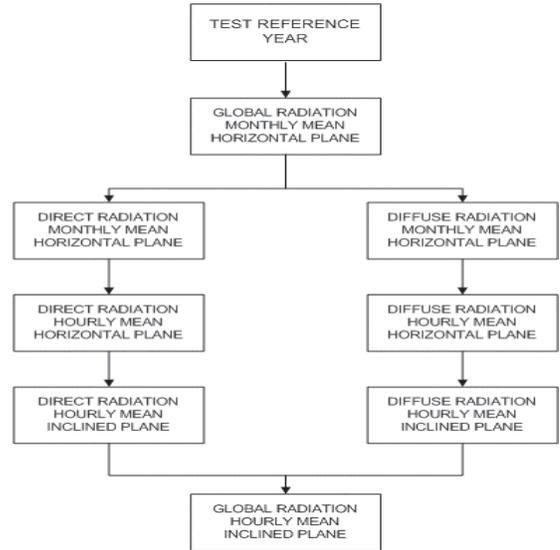


Fig. 5. The solar radiation based on the reference year.

The solar irradiance of 1000 W/m<sup>2</sup> and the temperature of 25°C are the pattern conditions for the photovoltaic arrangement [8].

The maximum value for a fixed solar intensity is found in the inflexion point of the modulus standard curve. The luminous intensity and the cells temperature mainly influence the photovoltaic modulus performance.

An increase in the isolation level also increases the cell temperature and, consequently, tends to reduce the modulus efficiency [8]. Because of that, crystalline solar cells reach their greatest efficiency at low temperatures - the coefficients depend on the type of material.

The elevation of temperature decreases significantly the value of the voltage generated by the modulus and causes a little rise in the current value.

The equation describing the photovoltaic generator performance relating voltage, solar intensity and temperature is given by equations (1), (2) e (3):

$$I = I_{sc} \left\{ 1 - C_1 \left[ \exp \left( \frac{V - \Delta V}{C_2 V_{oc}} \right) - 1 \right] \right\} + \Delta I \quad (1)$$

where  $C_1$  and  $C_2$  are coefficients described by the following formulas:

$$C_2 = \frac{V_{MP}/V_{OC} - 1}{\ln(1 - I_{MP}/I_{SC})} \quad (2)$$

$$C_1 = \left( 1 - \frac{I_{MP}}{I_{SC}} \right) \exp \left( - \frac{V_{MP}}{C_2 V_{OC}} \right) \quad (3)$$

where:

- $V$ : output circuit voltage;
- $VOC$ : open circuit voltage;
- $VMP$ : maximum power voltage;
- $I$ : output current;
- $SC$ : Short circuit current;
- $IMP$ : maximum power current.

#### D. Battery

The battery is a convenient device to store electric energy generated by the photovoltaic modulus. The initial charge, the discharge value, and the efficiency are represented as follows:

- $EB$ : storage energy in the batteries;
- $EBmin$ : minimum discharge level;
- $EBmax$ : maximum charge level;
- $\eta_{bat}$ : battery efficiency;

During the first hour, the charging and discharging process in the batteries respect the following restriction:

$$E_{Bmin} \leq E_B \leq E_{Bmax}.$$

This restriction is regulated by the controller element. Its characteristics are described in the next subsection.

#### E. The Charge Controller

This element disconnects the photovoltaic modulus when the battery reaches its maximum charge level. Also it activates the supplying element when the minimum charge level is reached.

The mathematical model describing its operation is given below. It takes into account two situations: the loading and unloading battery level [10].

After applying the charge and discharge restriction for the battery, the controller will charge up to the maximum limit if the energy generated by the photovoltaic arrangement exceeds the energy demanded (Equation 4).

Otherwise, if the energy demanded by the charge is higher than the one generated by the modulus, the batteries will be discharged to supply the deficit according to Equation 5.

$$E_B(t) = E_B(t-1) + \left( E_G(t) - \frac{E_L(t)}{\eta_{inv}} \right) \eta_{bat} \quad (4)$$

$$E_B(t) = E_B(t-1) - \left( \frac{E_L(t)}{\eta_{inv}} - E_G(t) \right) \quad (5)$$

where:

- $EG$ : Energy generated by the photovoltaic module;
- $EL$ : Energy wasted in the loading process;
- $t$ : Time to be considered (Hour);
- $\eta_{inv}$ : Efficiency of the Inverter;

#### F. The Inverter

Also known as DC-AC converter (direct current - alternating current), it establishes a connection between the photovoltaic generator and the DC source.

Its efficiency is given by the ratio power input and the power output varying within a band of 50% to 90% [10].

The efficiency value percentage, commonly specified by the manufacturer, is the maximum value that can be reached by this element. It is represented by  $\eta_{inv}$  (Equations 4 and 5).

### IV. CASE STUDY: PHOTOVOLTAIC SOLBRASIL TOOL

This section presents a case study based on the framework developed. It is important to emphasize that the hot spots are implemented both by the specialization of the abstract classes and the composition of elements, such as the ones related to the devices SBFVSolarPanel, BFVController, SBFVBattery and SBFVInverter. The Figure 6 shows an instance of the framework - the SolBrasil photovoltaic system.

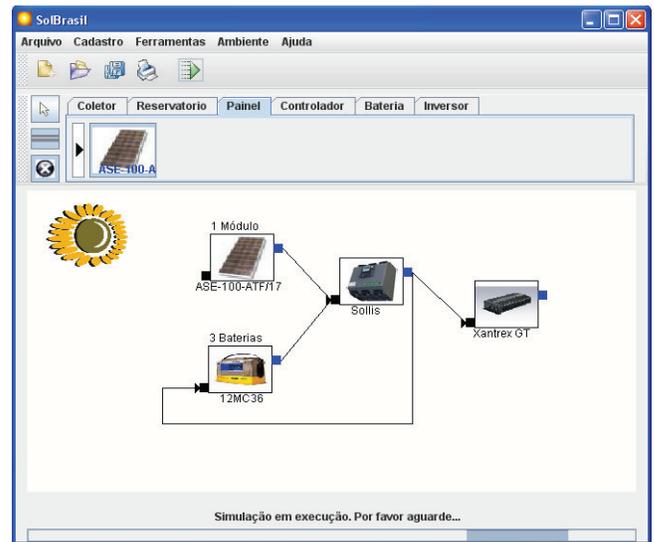


Fig. 6. SolBrasil - A framework instance.

#### A. The Execution Diagram

The design of photovoltaic systems depends on where it will be installed. The latitude and altitude data are important to the calculation methods.

The inclination angle of the photovoltaic generator, which maximizes its own energy production, varies through the year. It means that the same photovoltaic generator can be calculated from one region to another with different estimated values of solar radiation.

The global radiation on the panel surface considered as reference is the azimuth surface angle = 180° and the nearby reflectivity = 0,22 of Belo Horizonte city, Minas Gerais, Brazil.

The execution diagram was designed for a photovoltaic home system, isolated from the power supply system. The

devices were specified according to the manufacturers specifications: ASE-100-ATF/17 from ASE Americas manufacturer; the 12MC36 battery from Moura manufacturer; the Sollis controller element from Varixx manufacturer; and the XantrexGT conversor from Xantrex manufacturer.

### B. The Demand Definition

The operation of the photovoltaic system depends on the electric demand to be informed. The frozen spot FrmConsumo registers the consumption outline and its data are stored in the corresponding database structure.

The identification of the reference year allows the photovoltaic calculation to be done daily along a year. The Figure 7 shows the registered demand for the simulation.

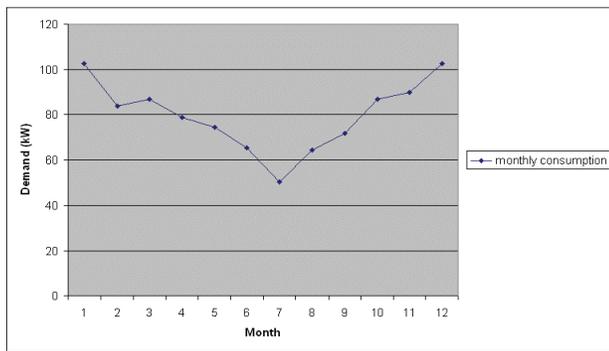


Fig. 7. Energy consumption graph.

The framework operates based on the registered energy consumption. It is also possible to inform the demand through the element energy consumption (given in w/h).

### C. The Results

Starting from the reference year it is possible to generate a synthetic sequence of the light index of the rainy days. The output is related to the solar irradiance and consequently is extremely important to the simulating process.

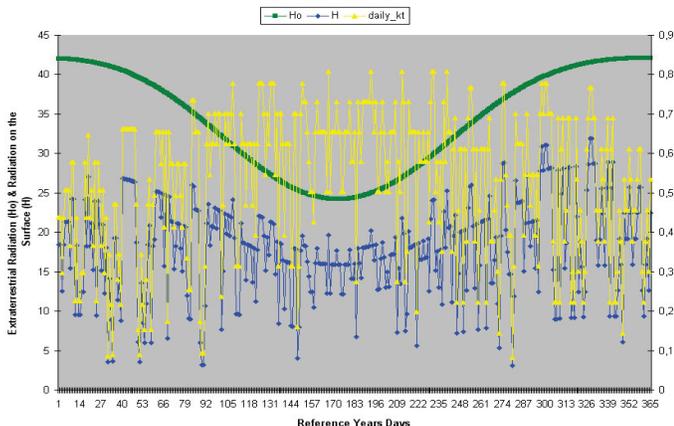


Fig. 8. The relation between the Extraterrestrial Radiation and the surface one, in the function of the light index along the year.

The Figure 8 shows the relation between the light index along the reference year and the extraterrestrial radiation reflected on the surface.

It is represented by the  $H_o$  curve and it does not have any influence on the deviation caused by the layers at the atmosphere. So, its behavior is regular and its intensity is high in almost every moment.

The radiation on the earth surface  $H$ , is shaped mainly in the function of the light index  $K_t$ , which represents the influence of clouds and atmospheric gases in the radiation brightness, that is, as cloudier it is, lower the radiance will be.

The Figure 9 shows the monthly average hourly radiation based on the reference year. It can be notice an increase in the irradiation (center of the graphic) showing that the inclination

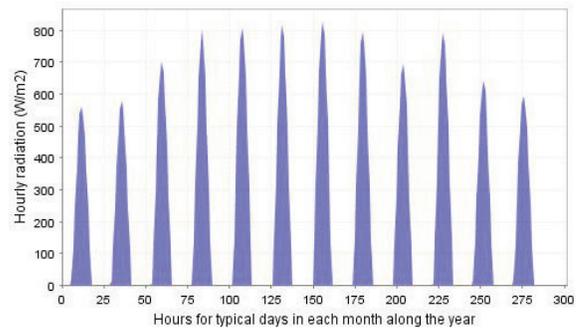


Fig. 9. The monthly average hourly radiation based on the reference year.

has an influence in the photovoltaic modulus during the months with lower indexes of irradiation and with elevated indexes of clarity.

The Figures 10 and 11 represent the graphics for the power supplied by one photovoltaic modulus and for the stored energy in a bank with three batteries along a year. After the computation of the irradiation data, the calculus is done to estimate the maximum power point of the photovoltaic generator. It is given by the ratio curve between the current  $I$  and the voltage  $V$  for each - the maximum value in the curve inflection. The graphic of the modulus-generated power is also in along-the-year hourly average.

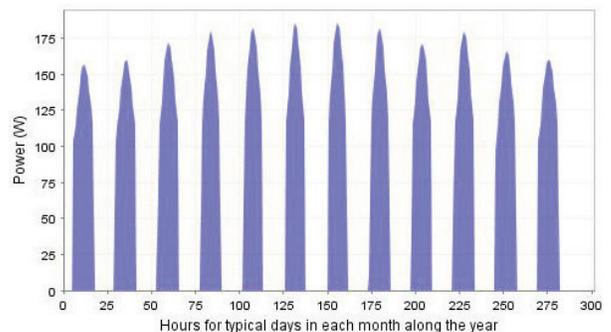


Fig. 10. Monthly Mean Hourly Power generated by the photovoltaic modulus along the year.

In the graphic for the stored energy in the battery bank (Figure 11), the top line delimitates the maximum capacity of storage, over which the bank would be overcharged; the bottom line delimitates the maximum depth of charge - below it the batteries would be exposed to a deep discharge that would damage them.

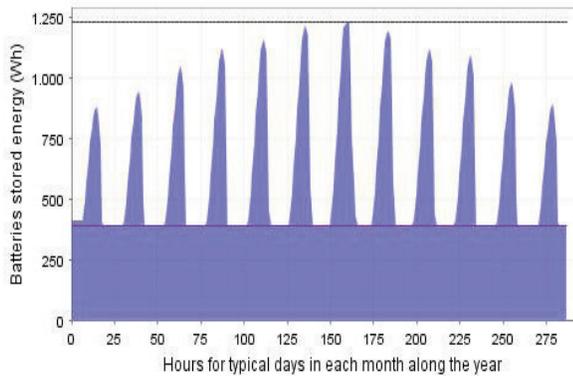


Fig. 11. Batteries Stored Energy along the year.

It can be noted that the available capacity is placed between two limits and that the charge controller interferes in such a way that it does not allow the bank to be overloaded or to receive a deep discharge.

It also shows that the energy, which is stored along the year, suffers a directly proportional variation to the medium hourly power generated by the modulus monthly - it avoids the batteries charging to its maximum initial value which certainly reduces the batteries useful lifetime.

## V. CONCLUSION AND FUTURE WORK

The framework approach allows a great flexibility in the development of new applications due to the facility in customizing users requirements. It also enables the reuse of the analysis, project and source code. It increases software quality, reduces the effort and time to develop new applications.

In this work it was developed a system, to simulate real systems, especially those ones related to the conversion of solar into electric energy through photovoltaic cells.

The proposed solution has simplified the procedures taken by solar energy researchers to develop control systems. Besides, it has allowed the reuse of a consolidated and tested solution.

This work can be considered an innovative proposal for the solar energy area as it has developed a methodology based on framework capable of calculating the behavior of photovoltaic systems isolated from electric network by means of analyzing the solar radiation, the energy balance, the statistical results and the identification of failures.

A future work intends to investigate the application of the proposed solution in systems related to other renewable energy sources such as wind systems.

## ACKNOWLEDGEMENT

This research has received financial support of FAPEMIG through the PPMIII-67/09 Project and CNPq through the 471089/2008 Project.

## REFERENCES

- [1] F. R. Martins, E. B. Pereira, S. L. Abreu, and S. Colle. Mapas de irradiação solar para o Brasil. resultados do projeto SWERA. XII Simpósio Brasileiro de Sensoriamento Remoto, pages: 3137–3145, 2005.
- [2] Silva, M. R. A. da, Oliveira, M. C. de and Nogueira, R. F. P. Photo-Fenton applicability in the treatment of ink wastewater industry. *Eclat. Quím.*, 2004, vol.29, no.2, p.19-26. ISSN 0100-4670.
- [3] Al-Rabghi O.M. and Akyurt M.M. A survey of energy efficient strategies for effective air conditioning, *Energy Convers Manage* 45 (2004), pp. 1643–1654.
- [4] Fontoura, M. Pree, W. and Rumpe, B. UML-F: A Modeling Language for Object-Oriented Frameworks. Proceedings of the ECOOP, pages 63–82, July 2000.
- [5] Hamu, D. and Fayad, M. Achieving Bottom-Line Improvements with Enterprise Frameworks, *Communications of ACM*, 41(8), 110-113, 1998.
- [6] Fach, P.W. Design Reuse through Frameworks and Patterns. *Software, IEEE*, 18(5):71-76, 2001.
- [7] Martins, Fernando Ramos, Pereira, Enio Bueno and Echer, Mariza Pereira de Souza. Solar energy resources assesment using geostationary satellites in Brazil: Swera Project. *Rev. Bras. Ens. Fis.*, 2004, vol.26, no.2, p.145-159. ISSN 0102-4744.
- [8] R. Festa and C. F. Ratto. Proposal of a Numerical Procedure to Select Reference Years. *Solar Energy*, 50(1):9–17, January 1993.
- [9] A. Argiriou, S. LykoudiS, S. KontoyiannidiS, and C. A. BALARAS. Comparison of Methodologies for TMY Generation Using 20 Years Data for Athens, Greece. *Solar Energy*, 66(1):33–45, May 1999.
- [10] Oliveira, and A. S. A. Diniz. Modelagem e simulação de gerador fotovoltaico. *Energia Solar*, pages: 15–19, 2004.

# Data manipulation API in ERP systems

Vadym Borovskiy, Wolfgang Koch, and Alexander Zeier  
Hasso Plattner Institute for Software Systems Engineering  
P.O. Box 900460, D-14440 Potsdam, Germany

{vadym.borovskiy, alexander.zeier}@hpi.uni-potsdam.de, wolfgang.koch@sap.com

## Abstract

*Efficiency in data manipulation is of vital importance to ERP systems. Flexible data manipulation API helps to address a number of acute needs of end users and application developers. To be efficient the API must meet a number of requirements: (i) allow data operations at any granularity level, (ii) guarantee data integrity, (iii) be scalable, (iv) allow cross-platform invocations, (v) be convenient to use. None of the ERP systems exposes an API like that.*

*Building on the notions of business objects and Web services, this paper contributes with the concept of a business object query language (BOQL). Essentially, BOQL is a query-like service invocation, which provides on-the-fly orchestration of CRUD-operations of business objects. BOQL allows to achieve the desired level of data manipulation efficiency at a reasonable price. To demonstrate the efficiency of the suggested approach the paper presents a use case showing how BOQL enables the development of composite applications.*

## 1 Introduction

Data manipulation is critical to enterprise resource planning (ERP) systems. An ERP system is a data-driven application, meaning that accessing and manipulating data represent the biggest share of its workload. The more efficient these operations, the greater the value of the system. The current paper elaborates on two questions: what makes ERP data manipulation efficient and how the efficiency can be achieved. The main contribution of the paper is the concept of a business object query language (BOQL). Essentially, BOQL is a query-like service invocation, which provides on-the-fly orchestration of CRUD-operations of business objects. BOQL allows to achieve the desired level of data manipulation efficiency at a reasonable price.

The rest of the paper is structured as follows. Section 2 discusses the requirements that an efficient data API must satisfy. Section 3 gives an overview of related work and

analyzes the state of the art in ERP data access. Section 4 presents the business object query language (BOQL), which is the main contribution of the paper, and shows how it satisfies the requirements discussed in Section 2. Section 5 presents a prototype demonstrating how an ERP system can expose BOQL as an interface to its data. Section 6 presents a use case, where BOQL is used to develop a composite application. Section 7 concludes the paper.

## 2 Accessing and manipulating ERP data

To be efficient, data API in an ERP system must satisfy a number of requirements: (i) allow data operations at any granularity level, (ii) guarantee data integrity, (iii) be scalable, (iv) allow cross-platform invocations, (v) be convenient to use. Each requirement is explained below.

The first requirement, interprets efficiency as the ability to access any combination of attributes with as little effort as possible. An ERP system stores in its database hundreds of gigabytes of data. Typically, these data are fragmented. For example, a sales order header is stored separately from sales order items. The fragmentation is the direct consequence of data normalization, the process of ensuring the maximum cohesion of entity types and eliminating redundancy in data model. The main argument in favor of normalization is that the management of normalized data is simpler than that of denormalized. The disadvantage of normalization is that it disjoints semantically coupled attributes into a number of subsets and stores them separately (in different database tables). Data API in such systems (e.g. SAP R/3 and SAP Business Suite) is often organized around these subsets, meaning that the system exposes operations that are able to manipulate only the attributes that belong to a single table. This is very inefficient, because the semantics of data operations in most applications assumes manipulation on application domain objects (purchase order, customer invoice and etc.) rather than on database tables. In this case the API's granularity does not match the granularity of the applications' needs. Whenever such a mismatch happens, application developers have to take the burden of express-

ing the required operation in terms of available ones. This complicates the code and leads to situations, when data manipulation part of an application outweighs the application's value adding part.

The second requirement, data integrity, ensures that any usage of the API must not compromise the integrity of ERP data. Data integrity means that the manipulation of data must be performed in accordance with business logic rules. For example, when creating a sales order, availability check must be performed. Only after a successful confirmation a sales order entry can be created in the system. Without such a check, creating a sales order by simply inserting data into corresponding database tables with SQL statements can result in a sale of not existing products.

The third requirement, scalability, ensures the ability of the target API to handle increasing number of requests. One of the current trends in ERP systems development is switching to software as a service (SaaS) model. No maintenance and a better pricing model based on actual usage drive organizations to favor SaaS ERP solutions over traditional on-premise ones [7]. Because the service provider rather than the consumer has to bear the cost of operating the system, the provider is strongly motivated to reduce the system's total cost of ownership. A well-designed hosted service reduces total cost of ownership by leveraging economy of scale. Sharing all aspects of the IT infrastructure among a number of consumers allows the provider to reduce operating expenses and thus increase profit. Consolidation of consumers onto a single operational system, however, comes at the price of increased system complexity due to multi-tenancy at the database layer. Multi-tenancy is basically mapping a number of single-tenant logical schemas in one multi-tenant physical schema in the database [4]. The inevitable consequence of multi-tenancy is increased workload: the system has to handle requests from much more consumers in comparison to a single-tenant system [8]. Therefore, scalability becomes a crucial characteristic of an ERP system, if its provider plans to offer it as a service.

The fourth requirement, cross-platform support, is necessary in the contemporary heterogeneous world. The variety of platforms, from which applications access ERP systems, has dramatically increased over the last decade. Desktop computers running different operating systems, mobile devices, enterprise-class servers, all may run applications interacting with ERP systems. Designing specific API for every platform configuration is costly in terms of development and maintenance. Therefore, to support such a heterogeneous environment, the target API must be based on industry standards and avoid the usage of proprietary technologies. Standards compliance, however, must not be abused. It is a trade-off: by improving interoperability designers often sacrifice performance and simplicity of code.

The last requirement is the ease of use. Although obvious, the requirement often gets overlooked. As consequence, powerful functionality can become not usable. In the context of ERP data manipulation the main usability factor in our opinion is the transparency of data model. Therefore, the target API must not only expose data of an ERP system, but also the system's data model or metadata.

### 3 Related work

A straightforward approach to data manipulation can be to use SQL. Since an ERP system relies on relational database, SQL statements could be issued directly against the database to operate on data. Although feasible, this approach is unlikely to satisfy the requirements stated above. The problem with SQL is that it violates the data encapsulation principle. It exposes too much control over the underlying database and increases the risk of corrupting data in the system. An ERP system is not only data, but also a set of business rules that apply to the data. Generally, these rules are not a part of the system's database. Direct access to the database circumvents the rules and violates data integrity. Therefore, accessing directly the data by any means must be prohibited.

An alternative to SQL can be data as a service approach. In this case a system exposes a number of Web services with strongly-typed interfaces operating on data. This approach has an advantage of hiding internal organization of data. Instead of a data schema and a query interface an ERP system exposes a set of operations that manipulate its data. By choosing operations and calling them in an appropriate sequence required actions can be performed. Because of using Web services this approach is platform independent. In fact, the data-as-a-service approach has been very popular. SAP, for instance, has defined hundreds of Web service operations that access data in SAP Business Suite [2]. Amazon Electronic Commerce service is another example of such approach [1]. However, this method has a serious disadvantage - lack of flexibility. Although an ERP system can expose many data manipulation operations, they will unlikely cover all combinations of attributes that applications might need to operate on. Therefore, granularity mismatches are very likely to occur. As discussed earlier, this will require application developers to manually construct a sequence of calls on existing operations to perform a desired manipulation. An example of such a case is presented in [6]. To partially overcome the mismatch, the interfaces of Web services can be relaxed [5]. This, however, will blur the semantics of the operations.

Service data objects (SDO) [9] enhance the data-as-a-service approach by rigidly specifying many aspects of data manipulation API. SDO is a specification for a programming model that unifies data programming across hetero-

geneous data sources, provides robust support for common application patterns, and enables applications, tools, and frameworks to more easily query, view, bind, update, and introspect data [3]. SDO has a composable (as opposed to monolithic) architecture and is based upon the concept of disconnected data graphs. Under the disconnected data graphs pattern, a client application retrieves a data graph from a data source, mutates the it, and can then apply the data graph changes back to the data source. Access to data sources is provided by a class of components called data access services. A data access service (DAS) is responsible for querying data sources, creating graphs of data containing data objects, and applying changes to data graphs back to data sources. SDO essentially wrap data sources and fully control access to them via a set of strongly-typed or dynamic interfaces. SDO offer a number of advantages: data encapsulation, better semantics in comparison to the previous approach (because manipulation API is organized around data objects from application domain), better modularity and reuse. However, SDO have a weakness (as in the case of data-as-a-service approach): the problem of interface design is not solved. Therefore, granularity mismatches are possible, but the usage of dynamic interfaces can alleviate the problem at the cost of code complexity.

As one can see all approaches have advantages and disadvantages. SQL as a data access API gives great flexibility by allowing to construct queries that match the granularity of any information need. However, SQL exposes too much control over the database and circumvents business logic rules. The data-as-a-service and SDO approaches, on the other hand, enforce business rules by exposing a set of Web operations, which encapsulate data manipulation and hide data organization. However, the granularity of the exposed operations often does not match the needs of application developers due to poor interface design.

#### 4 Business object query language

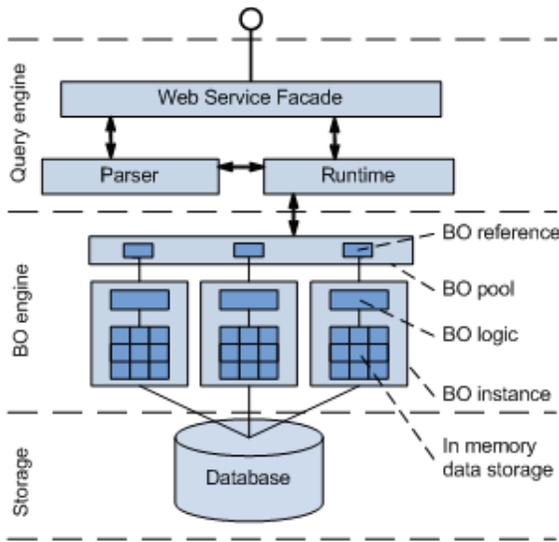
Forming a business object out of semantically related data and accessing the data via a fixed set of operations give good control over the data. On the other hand, exposing the data via a specific interface limits application developers in ways they can manipulate the data. To overcome this deficiency we suggest to use only a dynamic interface of a business object and automatically generate the sequence of calls to its operations, given a special *query* (compact, formal description of an action a developer wants to perform on the business object).

Despite the diverse semantics of business objects they all have the same structure (an array of attributes and associations) and behavior (a set of operations). The most basic set of operations a business object supports is called CRUD - *Create, Retrieve, Update, Delete*. Although too generic, this

set of operations has an advantage that any business object can support it. Therefore, all business objects can be derived from the same base class featuring the mentioned arrays (of attributes and associations) and CRUD-operations. Such uniform behavior and structure allow to introduce a query language for business objects very much like SQL for relational entities. We propose the following scenario:

1. A programmer composes a query, the description of what to retrieve from or change in the system, according to some SQL-like grammar, and sends the query as a string to the system by calling a generic Web service operation, for example *ExecuteQuery*.
2. The system parses the string to detect present clauses (*from, select, where, etc.*) and builds a query tree - an internal representation of the query. The tree is then passed for further processing to a query execution runtime, very much like in a DBMS.
3. Using the *from* clause the runtime obtains references to the business objects, on which the operations must be performed - source business objects. Then the runtime traverses the query tree in a specific order and converts recognized query tokens to appropriate CRUD-calls on the source business objects. For example, tokens from *select* clause are converted to *Retrieve* or *Retrieve-ByAssociationChain* operations, while tokens from *update* clause are converted to *Update* operations.
4. Having constructed the call sequence, the runtime binds corresponding string tokens to the input parameters of CRUD-operations. For example, the token *Customer.Name* of the *select* clause is interpreted as a call to *Retrieve* operation with the input parameter value "Name" on the business object Customer. Now everything is ready to perform the calls of CRUD-operations in the on-the-fly constructed sequence.
5. The last step is the composition of result set. The result is compiled in an XML document and sent back to the calling application.

In its essence the BOQL performs on-the-fly orchestration of calls to objects' CRUD-operations based on user-defined queries. These queries are transformed to a sequence of calls that perform the required action. BOQL has an advantage of supporting queries at any granularity level as in the case of SQL (because of using dynamic interface of business objects) without circumventing business rules (because CRUD-operations control data manipulation and enforce business logic rules). BOQL is made feasible by a uniform representation of business objects (in terms of the structure and behavior). As one can see BOQL already fulfills the requirements *i, ii* and *iv* from Section 2. The rest



**Figure 1. The architecture of the test system**

we consider to be implementation details and address in the next section.

## 5 Implementing BOQL

The current section demonstrates how an ERP system can support BOQL. The Figure 1 sketches the architecture of a prototyped system. BOQL is implemented by two elements: a business object engine and a query engine: the former manages business objects in a way BOQL assumes<sup>1</sup> and the latter provides access to them from outside of the owning process via a query-like interface. These two elements are instances of *BoEngine* and *QueryEngine* classes respectively. Both are created at the system's startup time. Business object engine is instantiated first to assemble business objects and store references to them in a pool. Then the instance of the query engine is created. It has access to the pool and can manipulate the objects.

Every business object encapsulates an in-memory table to cache data. The in-memory table is populated with data taken from a private database. Every object also encapsulates logic to synchronize its in-memory table with the database. To the query execution runtime an object is available via its interface: a collection of attributes and associations to other objects and CRUD-operations. How those are implemented is completely hidden inside the object. Neither business objects nor their in-memory cache and database tables can be directly accessed outside of the owning process. The direct access to the data is prohibited to enforce internal business logic. To access the business data an external application must use the standardized query-like

<sup>1</sup>meaning that it guarantees the compliance to CRUD-interface

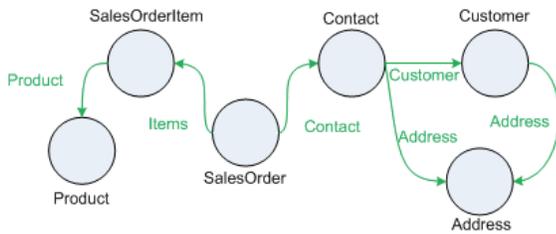
interface exposed by the query engine. When the latter receives a query it parses it and transforms recognized tokens to corresponding operation invocations.

The implementation platform for the prototype is .NET. The system is implemented as a Windows service and the query interface is published as a Web service hosted by Internet Microsoft Information Services (IIS). The Web service is meant to dispatch a query to the system and serves as a request entry point. There is no other way to invoke or access the system except for issuing a call to the Web service. The physical data storage is implemented as a Microsoft SQL Server 2005 database.

To make the architecture scalable, it must be multithread-aware and bottle-neck-free. The Web service can easily become a bottleneck in the system. Therefore, the actual BOQL query execution is factored out from the Web service to so called working processes. Each process can run on any physical server, to which the Web service hosting machine can establish a TCP connection. Every working process has a queue, to which the Web service appends BOQL queries, and a pool of working threads, which pull out the queries from the queue and process them. The queue is very important, as it enables asynchronous communication between the Web service and the working process. The former does not wait for the completion of the query and becomes quickly available to other client applications. To determine a working process that should be assigned a BOQL query the Web service uses round-robin scheduling algorithm. Hence, if the number of queries increases, the system's administrators must simply instantiate more working processes<sup>2</sup> and register them with the Web service.

Using BOQL requires the knowledge of business object model: the list of business objects the system has, their attributes and associations. To communicate this information we have developed a tool called *Schema Explorer*. It automatically retrieves metadata from the system and builds a business object graph (see Figure 2). Such a tool greatly simplifies the creation of BOQL queries by offering a plenty of useful functionality: business object search, association and attribute search, finding connections/paths between any two business objects, displaying a business object graph or its part, intellisense support for query editor, test execution of a query, to name just a few. The business object metadata is obtained using reflection mechanism of .Net Framework. Query engine exposes a number operations which internally use reflection to query the business object metadata. For example, if a developer wants to know what business objects the system has, the tool calls an operation which scans the pool and obtains the types of business objects instantiated by the system. To look up the list of associations of a given business object, say Customer, the tool issues a call to another operation that gets the names of elements in the As-

<sup>2</sup>of course free hardware capacity must be available



**Figure 2. Schema of the Web retailer's CRM**

sociations array of the corresponding business object. Because business object model does not change the metadata is cached in order to avoid the reflection overhead.

## 6 Example

Consider a Web retailer that sells items online and sub-contracts a logistics provider to ship sold products. The retailer operates in a geographically large market (e.g. the US or Europe). In this situation the consolidated shipment of items can generate considerable savings in delivery. Consolidation means that a number of items is grouped in a single bulk and sent as one shipment. The savings come from price discounts gained from higher transportation volume and less transaction cost per item shipped (because it is the bulk that is charged, but not individual items).

To manage their sales, the retailer uses a system with the business object graph as Figure 2 presents. We assume that the system exposes a query-like Web service interface as described in the section 5. The query returning the shipping address for all sales order items that are to be delivered looks as follows:

```
SELECT
  SO~id, SO.Contact.Address~postalAddr,
  SO.Items~id, SO.Contact.Customer~name,
FROM
  SalesOrder As SO
WHERE
  SO.Status = "ToDeliver"
GROUP BY
  SO.Contact.Address~city
```

By invoking the query-liked Web service and passing the above query to it, a third-party application consolidates the items by their destination. The next step for the application is to submit a request for quote to a logistics provider and get the price of transporting each group of items. Many logistics providers have a dedicated service interface for this, so the application can complete this step automatically. Once the quote has been obtained and the price is appropriate the products can be packaged and picked up by the logistics provider.

## 7 Conclusion

Efficient data manipulation API is essential to ERP systems. To be efficient, the API must satisfy a number of requirements: (i) allow data operations at any granularity level, (ii) guarantee data integrity, (iii) be scalable, (iv) allow cross-platform invocations, (v) be convenient to use. Existing approaches and APIs do not satisfy all these requirements. The current work contributes with the concept of query-like service invocation implemented in the form of a business object query language. BOQL offers both the flexibility of SQL and encapsulation of data-as-a-service and SDO approaches. In its essence, BOQL is on-the-fly orchestration of CRUD-operations exposed by business objects of an ERP system. The authors also showed that BOQL satisfies all five requirements. In addition the paper demonstrated how BOQL enables the development of enterprise composite applications. Furthermore, the major components of the architecture were outlined and prototyped with Microsoft .NET platform.

## References

- [1] Amazon product advertising api. <https://affiliate-program.amazon.com/gp/advertising/api/detail/main.html>. last visited: 22.03.2010.
- [2] Sap enterprise services workplace. <http://esoadocu.sap.com>. last visited: 22.03.2010.
- [3] Service data objects white paper. <http://www.osoa.org/display/Main/SDO+Resources>. last visited: 22.03.2010.
- [4] S. Aulbach, T. Grust, D. Jacobs, A. Kemper, and J. Rittinger. Multi-tenant databases for software as a service: schema-mapping techniques full. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1195–1206, 2008.
- [5] V. Borovski, S. Enderlein, and A. Zeier. Generic web services - extensible functionality with stable interface. In *IEEE International Conference on Web Services*, 2009.
- [6] M. Grund, J. Krueger, and A. Zeier. Declarative web service entities with virtual endpoints. In *Proceedings of the IEEE International Conference on Services Computing*, 2008.
- [7] D. Jacobs. Enterprise software as service. *Queue*, 3(6):36 – 42, 2005.
- [8] D. Jacobs, S. Aulbach, A. Kemper, and M. Seibold. A comparison of flexible schemas for software as a service. In *Proceedings of the 35th SIGMOD international conference on Management of data*, pages 881–888, 2009.
- [9] L. Resende. Handling heterogeneous data sources in a soa environment with service data objects. In *ACM SIGMOD international conference on Management of data*, 2007.

# Ontology-Based Tools in the Service of Hardware Verification

Eyal Bin, Alaa Ghanayim, Karen Holtz, Eitan Marcus, Ronny Morad,  
Ofar Peled, Michal Rimon, Gil Shurek and Elena Tsanko

IBM Research Lab in Haifa

Haifa University Campus, Haifa, Israel

{bin, alaa, holtz, marcus, morad, oferp, michalr, shurek, tsanko}@il.ibm.com

## Abstract

*By using an ontology-based approach as the underlying scheme of hardware verification tools, knowledge about the specific design under test (DUT) is kept separate from the tool's generic service engine. This separation enables tools to easily adapt to new DUTs. In this paper, we discuss how we successfully adopted an ontology-based approach to the development of hardware verification tools. We also present ClassMate, an ontology modeling platform developed and used in IBM that is especially suited to support hardware verification ontologies. The ontology-based approach has boosted the impact of IBM's hardware verification tools by facilitating reuse — enabling adaptation to a large number of complex DUTs including several generations of the same design, lowering maintenance costs per DUT, and driving the accumulation of deep domain knowledge.*

## 1. Introduction

Functional verification is the process of ensuring that a logic design conforms to its specification. In current industrial practice, simulation-based verification plays the major role in the functional verification of hardware designs. This technique verifies a design's correctness by simulating the description of the hardware expressed using some hardware design language [1], driving stimuli into this simulation model, and then checking the validity of the produced results. This model is termed *design under test* (DUT). The verification environment or *testbench* is created around the DUT and is composed of a large number of modules and tools. These include stimuli generators and drivers, monitors, coverage analyzers, checkers and reference models.

The development and maintenance of a testbench environment for a high-end server system is a large software endeavor involving dozens of verification engineers and spanning many years of effort, thereby challenging verification schedules and driving up hardware development costs. This

problem has become increasingly acute with the growing pressure for cost reduction and faster time-to-market.

Modern testbench environments are created using specialized languages and follow strict methodologies. Development environments supporting these languages and methodologies are available from leading design automation tool vendors [2, 3]. Testbench modules are built for a specific architecture, protocol, or DUT, and have deep knowledge about the DUT embedded in their code. Most testbench components do not provide good (or even any) separation of the application-specific knowledge from the core service engine. This means that the amount of sharing and reuse between testbench modules is limited, even between those that provide similar services for different DUTs. This level of reuse greatly contributes to the inflated development cost described above.

In this paper, we propose using an *ontology-based* approach for building hardware verification tools. With this approach, knowledge about the architecture of the DUT and the accumulated domain expertise are kept separate from the tool's main engine. Decoupling knowledge from control enables the same tool to be used for verifying different hardware designs or different generations of the same design. Thus whenever a new or follow-on hardware design is introduced, only its model need be specified; the actual application engine remains unchanged. The benefits of this approach are reduced time-to-market, higher quality tools (due to the stability of the application engine), and greater reuse of accumulated knowledge between hardware designs.

This paper also introduces ClassMate, an ontology modeling platform developed and used in IBM. The ClassMate modeling language can be viewed as an extension of UML 2.0 [4] Class Diagrams. This language provides several features that are especially suited for supporting hardware verification tools such as extended data types, constraints, refinement, and packages. The platform also provides a form-based graphical studio to facilitate rapid model development. ClassMate was developed to compensate for features and capabilities that were found lacking in existing on-

tology development languages and environments [5, 6, 7]. The modeling language of ClassMate is expressive enough to describe the intricacies of hardware designs and its expert knowledge, formal enough to facilitate maintenance and reuse, and easily convertible to the representation required for the operational needs of the verification tool.

ClassMate is currently used in IBM by several hardware verification tools including unit, processor, and system-level test generators [8, 9, 10], cache hierarchy initializers [11] and post silicon exercisers. These tools have been successfully deployed to verify IBM System p servers, IBM System z servers and gaming consoles (Microsoft XBOX360, PlayStation3, Nintendo Wii) [12]. It has been estimated that together these tools have saved IBM well in excess of 100 million dollars.

The rest of the paper is organized as follows. In Section 2, we describe the ontology-based tool scheme and its adaptation to hardware verification tools. In Section 3 we present the ClassMate platform, with emphasis on the modeling features that are essential for hardware verification. Finally, in Section 4, we review our experience with ontology-based hardware verification tools within IBM.

## 2. Ontology-based tools

This section discusses a software development scheme and a tool architecture intended to provide better productivity in the development and maintenance of software tools. We also describe the adaptation of this scheme to hardware verification tools.

The ontology-based architecture for tool development is suitable whenever a family of applications shares a well-defined application domain (e.g., travel planning and booking [13], business process modeling [14], test program generation [9]). Each family of applications shares similar principles of operation even though the applications themselves may provide services that are significantly different. In these examples, the development of the basic service engine and the adaptation for a specific application case require deep expert knowledge of the application domain and of the specific application case respectively.

### 2.1. Tool architecture

The key concept driving the ontology-based scheme is a domain specific modeling language. The language provides the terminology, i.e., the basic concepts, required to build an application-specific ontology. This application specific knowledge is then imported, or interpreted, by an application-domain service engine, which in turn uses it to provide the required application-specific services. Hence, the service engine coupled with the application-specific ontology becomes a specific service application (e.g., a per-

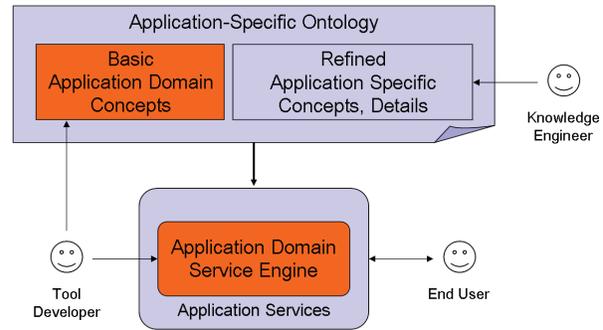


Figure 1. Ontology-based scheme

sonalized travel agent). Figure 1 is a diagrammatic illustration of the ontology-based scheme. The design and evolution of the service engine is tightly coupled with the domain specific modeling language. The engine interprets the application-specific ontologies by, and only by, the basic application domain concepts provided by this language.

The ontology-based tool architecture has created a new role, a *knowledge engineer*, on top of the traditional tool developer and user roles. The knowledge engineer is responsible for the development of the application-specific ontology.

This tool architecture requires a modeling platform to support the construction of application specific ontologies. It is desirable to use a general purpose modeling platform designed to support ontology-based tools. In this case, domain-specific languages would be derived from the language provided by the platform. In Section 3, we describe ClassMate, a modeling platform designed to support ontology-based tools.

### 2.2. Ontology-based hardware verification tools

Many components of a verification environment are tailored to verify a specific DUT. Some components have a well defined set of services and operating principles that hold beyond the specific DUT and therefore are qualified as tools. However, using the common development methodology these too would require dramatic adaptation, and sometimes total rewrite, in order to serve a new hardware design. Examples include microprocessor and system-level test program generators, system-level transaction and coherency checkers, behavioral reference models, and some system initialization and setup tools. Our experience in IBM shows that tools in this category benefit from adopting the ontology-based scheme (see Section 4 for examples).

The ontology supporting hardware verification tools should cover aspects of the DUT and its verification environment so that the application can interact correctly with the DUT and accomplish its verification task. These aspects may include: DUT architecture, interface protocols, system

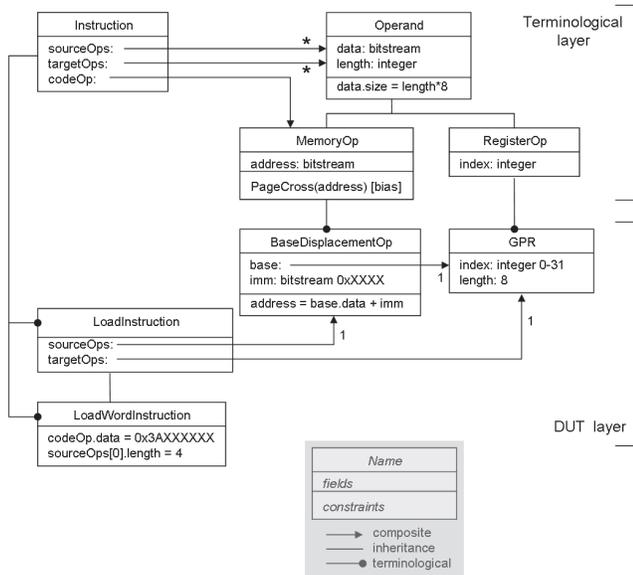


Figure 2. Model of a load word instruction

topology rules, micro-architectural details, behavioral aspects, validity assumptions and rules. A second component of the ontology should include design-specific expert guidance, generally referred to as *testing knowledge* (TK). TK should indicate effective methods to accomplish the application’s task (e.g., bug-prone areas and setups to guide or bias the generation of tests). This component would include any per-DUT tuning of the tool’s operating principles.

The following example illustrates how the ontology-based approach is deployed by a test program generator for micro-processor verification [9]. Consider the ontology fragment sketched in Figure 2. The model naturally decomposes into a two-tiered architecture. The *terminological layer* captures fundamental processor architecture and test program generation concepts and vocabulary. This layer constitutes the domain-specific modeling language supported by the test program generator. In contrast, the *DUT layer* is where DUT-specific concepts are developed by the application engineer using terms defined in the terminological layer of the ontology.

Instruction and Operand are generic concepts found in all processors and are therefore part of the terminological layer. All instructions have a set of source and target operands, and a single code operand. Operands have a length (number of bytes) and a data value whose size is a multiple of its length. Operands are further sub-classified into memory and register operands which characterize how they are addressed. The DUT layer models Power architecture [15] specific concepts. The Power architecture specifies that a load instruction transfers data between memory and one of the machine’s 32 general purpose registers. The ad-

dress of the source memory operand is equal to the sum of the value of a base register and an immediate value specifying the displacement. The LoadWordInstruction is a type of load instruction with a particular length, binary encoding and other unique attributes not described here.

The generic test generation engine would now traverse the Power architecture specific ontology to extract specifications of Power instructions. This is a reasoning step in which the engine uses the terminological layer in conjunction with embedded domain knowledge to interpret the specification and construct a *constraint satisfaction problem* (CSP) [16] for each appearance of an instruction in the test program. The CSP reflects the instruction’s structure, architectural rules, testing knowledge, program context and any specific user request. In a second reasoning step, the engine uses a constraint solver to randomly determine the instruction properties by sampling the solution space of the CSP. Modeling knowledge about the generation of instruction streams and the way it would be treated by the generation engine is beyond the scope of this example.

### 2.3. Advantages and trade-offs

The ontology-based development scheme drives the generalization of domain-wide expert knowledge and its encapsulation in a reusable engine. It promotes the formulation of a standard, domain-specific modeling language that captures the fundamental domain concepts and terminology, and encourages the use of declarative modeling techniques for knowledge representation. With the accumulation of these mechanisms it should effectively force modularity between domain-wide knowledge and application specific details. In terms of the expected return on investment (ROI) we aim to achieve the following advantages over traditional development processes: replicate the impact of a domain expert and reduce the dependency on the expert; accumulate domain knowledge from application to application; boost standardization and reuse across the application-domain, including reuse of ontology modules.

An ontology-based tool may be viewed as an application generator for a narrow application-domain. The advantages of the ontology-based development scheme and its ROI strongly depend on the selected application domain. There is a delicate balance to consider. By selecting an application domain that is too wide, the service engine becomes too generic to significantly boost development productivity for a specific application. By narrowing the domain, we risk serving only a few applications, thereby reducing the ROI for the additional complexity induced by this development scheme. With a ‘golden cut’ selection, we can have a service engine that embeds deep domain knowledge, boosts development and maintenance productivity, and reduces costs for a large set of specialized applications.

### 3. ClassMate platform

ClassMate is an ontology development platform built in IBM and used successfully by many of the company's hardware verification tools, several of which are briefly described in Section 4. It consists of a modeling language, a graphical studio for viewing and editing ontologies, an API for model reflection, and tools to reuse, validate, and query models. While ClassMate shares many features with existing ontology platforms, it goes beyond these systems by offering advanced features and concepts that are particularly suitable for hardware verification. In the rest of this section, we describe the ClassMate language and platform with an emphasis on these novel features.

#### 3.1. Types, constraints, instances and facets

All concepts in ClassMate are strongly typed. This enables the service engine to reason coherently about a DUT-specific ontology and allows for automated detection of modeling errors. ClassMate supports primitive (integer, boolean, etc.), enumeration, record and collection types. The *bitstream* primitive type is used to represent arbitrary but fixed sized bit vectors commonly found in hardware ontologies such as addresses and data values. Compound types such as records and collections are used to construct more complex user defined concepts. Record types consist of properties (data members) which are themselves typed. Properties have an optional default value which is used if no value is specified when the type is instantiated.

Types in ClassMate have extensional semantics in that they represent the possibly infinite set of values (or instances) the type can have. It is possible to restrict the extension of any type to a set of values which defines its *domain*. In the example of Figure 2, the domain of the index integer data member of GPR is 0-31, reflecting the fact that there are 32 general purpose registers in the Power architecture. Similarly, the LoadWordInstruction's code operand is restricted to the values contained in the 0x3AXXXXXX bitstream, where each X represents a don't care hexadecimal value. *Instances* are named, fully defined values of a type. Instances are particularly useful for restricting the domain of a type to a specific set of pre-defined values.

Powerful *constraints* that further restrict the extent of types are expressed through predicates over one or more properties. In hardware verification ontologies, constraints originate either from rules that govern the DUT or from the accumulated testing knowledge. The architectural rule that an operand address equals the contents of the base register plus the displacement is modeled as a constraint in our example. A TK indicating that it would be desirable for a memory access to cross page boundaries is modeled as a soft constraint intended to bias the generation of the in-

struction. ClassMate's modeling language provides a rich set of constructs for defining complex constraints including arithmetic, logical, bitwise, and set operators, as well as quantification over collections.

*Facets* act as hints to the application engine on how an entity of the model should be treated. They can be attached to any ClassMate entity. For example, facets can be used to specify if a constraint should be interpreted by the engine as a hard or soft constraint.

#### 3.2. Meta types, inheritance and refinement

As mentioned in Section 2.2, hardware verification models naturally decompose into terminological and DUT specific layers. ClassMate allows users to distinguish between ordinary types and meta types which are used to capture terminological concepts. This distinction is particularly important for an ontology-based tool since its service engine is constructed to be DUT-oblivious. Using reflection and guided by its knowledge of the meta types, a service engine can extract all the information necessary to enable its operation.

Types naturally form taxonomy hierarchies. This is true both for types defined in the terminological layer and in the DUT layer. In type hierarchies, a derived record type often introduces new data members not found in any of its base types. This type of derivation is analogous to inheritance found in object-oriented modeling. In ClassMate, we introduce the notion of *refinement* which extends derivation beyond traditional inheritance. Any restriction of the extent of a type is considered to be a refinement of the type. Examples of refinement include: restricting the domain of a type, adding constraints, restricting an arbitrary sized collection to have a fixed cardinality, and refining the underlying element type of a homogeneous collection. A refined record may include new data members or refine the type of an inherited one. The sourceOps property in our example combines both types of collection refinements. It is defined to be an arbitrary sized set of Operands in Instruction, whereas in LoadInstruction it is refined to be a single-element set of BaseDisplacementOps, which in turn are derived from Operand.

#### 3.3. Packages and redefinitions

Completely new hardware designs are rare. More commonly, a hardware design is a next generation follow-on to an already existing one. For this reason, ontologies for hardware verification are rarely built from scratch. Concepts may be common to all or some derivatives of the same hardware architecture or may be unique to a particular DUT. Thus, when creating ontologies for complex systems, it is useful to create packages containing partial models that can

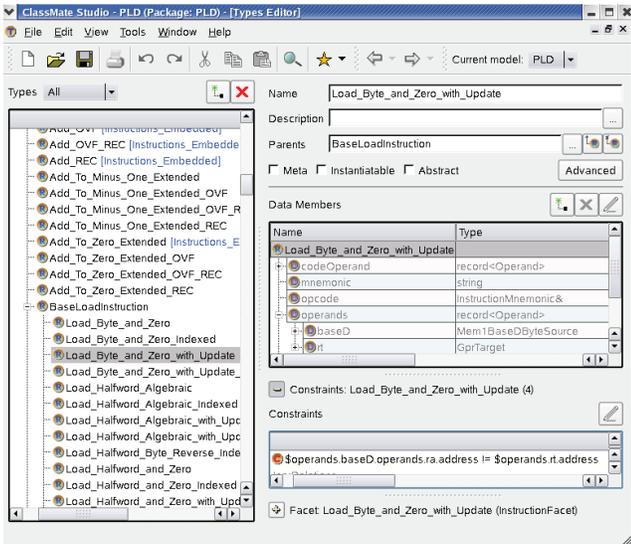


Figure 3. ClassMate Studio

be used as building blocks. A package contains a set of imported packages along with newly derived concepts. A DUT-specific ontology would therefore encompass a hierarchy of dependent partial-models or packages, designed to be reused between DUTs.

Frequently, in the process of developing follow-on models, it is useful to redefine a type introduced by an imported package without altering its original definition. In this way, other models that import the package are not affected, whereas the follow-on model reflects the type's new definition. ClassMate therefore supports type *redefinition*, a transformation that safely follows the rules of refinement but does not introduce a new type. The alternative to redefinition is to repartition packages and refactor the model hierarchy, which is a costly and error-prone activity. The ability to share sub-models between ontologies, and to redefine concepts in a type-safe manner for follow-on hardware designs, greatly facilitates model reuse.

### 3.4. ClassMate Studio

ClassMate Studio is a graphical development environment used for editing, browsing, navigating, and visualizing models. The studio adopts a form-based approach to facilitate rapid creation of potentially large ontologies dealing with types that contain a large number of properties. The tool provides four views of the model: types (shown in Figure 3), instances, constraint definitions and packages. In the types view, the left-hand side shows the type hierarchy. The right-hand side shows the contents of the currently selected type, including its composition hierarchy, constraints and facets.

ClassMate Studio provides advanced tools to assist in the

understanding, editing and refactoring of complex models. The first challenge a knowledge engineer faces before editing a model is to figure out where to apply the change and which components of the model would be affected. ClassMate assists in this task by searching for all references to a given entity across packages, and by generating diagrams of the inheritance and composition hierarchies for a given type. ClassMate supports several advanced refactoring scenarios: renaming types and properties, cloning and removing types, and restructuring type hierarchies including the insertion of intermediate types.

## 4. Ontology-based verification tools in IBM

Several tools in the hardware verification domain within IBM are ontology-based, following the operational scheme described in Section 2.2 and using ClassMate as their modeling platform. We briefly outline the services provided by some of these tools and their main modeling concepts before presenting some quantitative indicators based on our experience.

Genesys-Pro [9] is IBM's main processor-level test program generator. The Genesys-Pro model was used as a running example throughout this paper and will therefore not be further elaborated.

X-Gen [10] is a system-level test program generator for verifying systems. Its main modeling concepts are components (processor cores, bus bridges, etc.) and interactions performed between components.

The CacheLoader [11] application performs cache hierarchy initialization intended to drive the system into corner cases. Its model specifies cache structures and hierarchy, hashing functions and coherent initialization rules.

Generation of address translation resources is achieved via the DeepTrans [8] module. DeepTrans uses ClassMate to describe the address translation process, the translation resources, and the related architectural rules.

Table 1. Ontology usage results

	X-Gen	Genesys-Pro	Cache-Loader	Deep-Trans
Spec Size (pages)	2100	1800	15	200
No. DUTs	9	43	4	27
No. Follow-on DUTs	4	35	1	22
No. Basic Concepts	139	152	60	75
No. DUT Concepts	3784	2800	104	340
% reuse	68	56	85	56
New design effort	initial	1-2 mths	1 mth	3-7 days
	mature	1.5-2 yrs	1 yr	1-2 mths
Follow-on design effort	initial	1-2 wks	1 wk	2 days
	mature	8-10 mths	3 mths	1 wk

In Table 1, we present some quantitative indicators based on our experience with the above sample of ontology-based hardware verification tools. As a very rough measure of complexity, the average number of pages in the relevant

architecture specifications is shown. One can see that ontology-based verification tools have been used successfully to verify a large number of different and follow-on DUTs — up to 43 different DUTs are handled by Genesys-Pro. In each of these applications, the number of basic concepts in the terminological layer is relatively small, and yet it allows for the definition of a large number of concepts (up to several thousands) in the DUT layer. This shows that with the right ontology a solid basis for taxonomy hierarchies can be achieved. It can be seen that approximately 60 percent of modeling can be reused by follow-on DUTs. However, it should be noted that some follow-on DUTs are very similar to their base DUTs, resulting in reuse of 98 percent of the modeling concepts. The table also shows that utilizing the reuse capabilities of ClassMate results in a much faster bring-up time for follow-on DUTs — by a factor of 4 for initial models and a factor of 2 to 3 for mature models.

While the effort values listed in the table are high, note that most of the accounted DUTs are complex high-end server systems for which the development cost of DUT-tailored tools is estimated to be 3 to 7 times higher. In these cases, the high cost of tailored solutions would have risked affordability and schedules, a risk which most likely would have resulted in a compromise on quality and maturity of the tool.

## 5. Conclusion

In this paper, we have described how the ontology-based tool architecture has been adapted to the domain of hardware verification. Furthermore, we discussed the modeling features required to support hardware verification ontologies and showed how ontology modeling has successfully served several of IBM's prime verification tools. We presented ClassMate, an ontology modeling platform designed to support our tools. This practical, industrial scale platform was developed based on experience gathered over almost two decades of ontology-based tool development in the field of hardware verification.

Some research and development challenges still lie ahead. There is a need to architect ontologies and tools to enable the reuse of ontologies between multiple tools that serve the same DUT. Practical techniques should be devised for the modeling of behavioural aspects such as required for the creation of architectural reference models. The maintenance of model hierarchies for follow-on hardware designs needs to be simplified. We continually strive to achieve a better balance between pure ontological descriptions of the problem space (i.e., hardware designs and verification) and modeling that is more readily tuned for the underlying technologies (e.g., CSP solving).

Overall, the ontology based approach has increased the impact of IBM's hardware verification tools by boosting

reuse, thereby enabling adaptation to a large number of highly complex DUTs, lowering the per-DUT development costs, and driving the accumulation of deep domain knowledge, all of which help to meet tight verification schedules.

## References

- [1] Wile B., et al.: *Comprehensive Functional Verification - The Complete Industry Cycle*, Elsevier, 2005.
- [2] Palnitkar, S.: *Design Verification with e*, Prentice Hall, 2003.
- [3] Haque, F., Michelson, J. and Khan, K.: *The Art of Verification with Vera*, Verification Central, 2001.
- [4] UML 2.0 Specification, <http://www.omg.org/spec/UML/2.0>.
- [5] Smith, M. K., Welty C., and McGuinness D. L.: *OWL Web Ontology Language Guide*, <http://www.w3.org/TR/owl-guide>, 2004.
- [6] Noy, N. F., Fergerson, R. W. and Musen, M. A.: *The Knowledge Model of Protege-2000: Combining Interoperability and Flexibility*, EKAW 2000, pp. 17–32.
- [7] Denny, M.: *Ontology Building: A Survey of Ontology Editing Tools*, <http://www.xml.com/pub/a/2004/07/14/onto.html>.
- [8] Adir, A., Emek, R., Katz, Y., Koyfman, A.: *DeepTrans - A Model-Based Approach to Functional Verification of Address Translation Mechanisms*, MTV 2003, pp. 3–6.
- [9] Adir, A. et al.: *Genesys-Pro: Innovations in Test Program Generation for Functional Processor Verification*, IEEE Design and Test of Computers, Mar-Apr. 2004, pp. 84–93.
- [10] Emek, R. et al.: *Reuse in System-Level Stimuli-Generation*, HLDVT, 2005.
- [11] Bhatia, R., Bin, E., Marcus, E., Shurek, G.: *An Ontology and Constraint Based Approach to Cache Preloading*, HLDVT, 2010.
- [12] Shimizu, K. et al.: *Verification of the Cell Broadband Engine Processor*, DAC 2006, pp 38-343.
- [13] Ganzha, M. et al.: *Utilizing Semantic Web and Software Agents in a Travel Support System*, A. F. Salam and Jason Stevens (eds.), *Semantic Web Technologies and eBusiness*, 2006, pp.325–359.
- [14] Norton, B., Cabral, L., Nitzsche., J.: *Ontology-Based Translation of Business Process Models*, ICIW2009, pp.481-486.
- [15] Power ISA Version 2.06, <http://www.power.org>.
- [16] Bin, E, Emek, R., Shurek, G., Ziv, A.: *Using Constraint Satisfaction Formulations and Solution Techniques for Random Test Program Generation*, IBM Systems Journal, 2002, 41(3), pp. 386–402.

# Smarter Software Engineering: Knowledge factors contributing to improved Individual Performance

Narayanan Srinivasaraghavan, Craig McDonald and John Campbell  
School of Information Sciences and Engineering  
University of Canberra, Australia

**Abstract**—This research paper aims to understand the relative contribution levels of different knowledge factors to improvements in performance. A theoretical framework is developed that involves nine types of knowledge required for an individual to perform Software Engineering (SE) roles. Based on this, the research explores the contribution that additional knowledge makes to perceived performance improvement after an individual joins a SE team. The results indicate that Technique skills and Configuration knowledge (knowledge of application systems) contribute most to improvements in performance after an individual joins a SE team. It is found that Role has statistically significant relation with Contribution of Configuration knowledge to improvements in performance. These findings allow smarter software engineering. By targeted Knowledge Management (KM) initiatives to improve success rates and reduce failures.

**Keywords**—component; Smarter Software Engineering, Knowledge, Performance (key words)

## I. INTRODUCTION

This research aims to identify the contribution of various knowledge types to improved performance in Software Engineering in enterprise context. Individual knowledge is a fluid mix of framed experience, values, contextual information and expert insight that provides a framework for evaluating and incorporating new experiences and information [1]. SE is a knowledge intensive activity [2-7] requiring both tacit and explicit knowledge [8-11].

The following research question is examined: What are the contributing knowledge factors to improvement in individual performance? An understanding of contributing factors to performance in SE can allow enterprises to carry out better targeted KM, thus reducing SE project failures [11-13] and mitigate risks such as 1) schedule and budget overruns and 2) low quality [12, 14].

## II. LITERATURE REVIEW

### A. Knowledge Types

The literature review shows that several types of knowledge, skills or abilities are required to perform various SE roles<sup>1</sup>. Beazley et al. [16] provided a seven knowledge type

<sup>1</sup> Studies relating to knowledge and performance for business application software and systems development fall between Information Systems (IS) and SE fields [15]. Hence, the literature review included research papers on systems

framework for knowledge required to perform well in a given position. The three levels of skills classification (soft skills, technical skills and business concept skills) by Bailey and Stefaniak [17] was applied to Beazley et al. [16] framework. The application system knowledge (named as configuration knowledge) is identified as the content specific knowledge that is very specific to performing SE roles. Based on these, nine knowledge types are identified as required to perform SE roles. These are 1) Technique skills, 2) Soft skills, 3) Configuration knowledge (functional and technical knowledge of the application system), 4) Social network knowledge, 5) Business domain area knowledge, 6) Systems knowledge, 7) Process knowledge, 8) Cultural knowledge and 9) Heuristics knowledge.

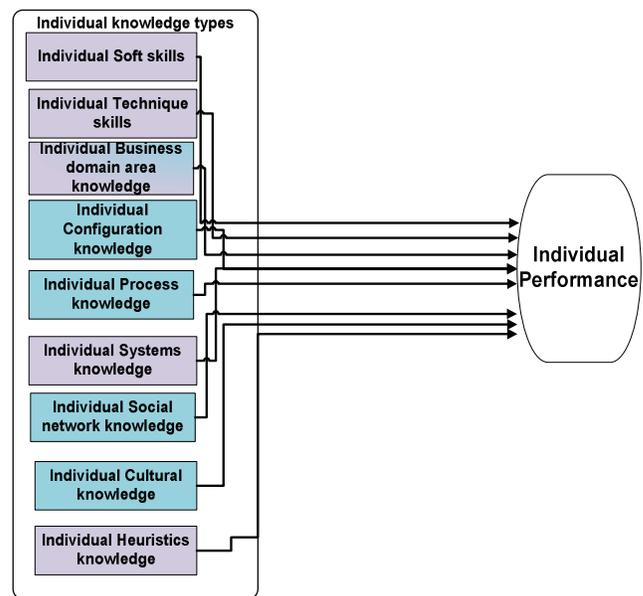


Figure 1. Theoretical framework

Technique skills, such as 1) Web programming, 2) Unix, 3) C++, [17-20] have been found to be useful to perform programmer and programmer-analyst roles. Technique skills such as business analysis skills, project management skills are required for business analysts and project managers respectively [18, 21-25].

Walz et al., [26] found that the knowledge is shared amongst members of the SE team showing the importance of development from SE and IS areas.

social network knowledge. Curtis et al., [27] found that a software engineer would normally communicate most frequently with team members, slightly less frequently with other teams on the project, much less often with corporate groups and except for rare cases, very infrequently with external groups [27].

Soft skills refer to non-technical skills performed in the intra- and inter-personal domains that facilitate the application of technical skills and knowledge [28]. Soft skill types have been demonstrated to be important in performing the job of a programmer [17], business analyst [29], project manager [30, 31], architects [32, 33] and systems analysts [34-36].

SE is about building, enhancing or maintaining an application software system in an enterprise context. Configuration knowledge or application system knowledge refers to the functional and technical knowledge of the specific software application(s), which is the focus of the SE activity. Several field studies have demonstrated the need for application system knowledge to perform various SE roles [26, 27, 37, 38]. These field studies have also recognised the need for business domain area knowledge to perform various roles in SE [26, 27, 37, 38]. Through experimentally created design contexts, Adelson and Soloway [39] demonstrated that the contextual business domain area knowledge is central to designers. It has been found that business domain knowledge is required for programmers [17], business analysts [34-36], architects Frampton et al., [33], business analysts [29] and project managers [30, 40].

Systems knowledge, including problem-solving skills and analytical and modelling skills as well as knowledge of development methodologies [25], has been found to be needed [33] to perform the roles of programmers, architects and systems analysts [18] and project managers [30, 40].

The importance of following processes in SE has been emphasised in literature [41, 42]. Many organisations have implemented process maturity frameworks such as CMMI [43] and SPICE [44]. The guide to SWEBOK includes the SE process knowledge area. Hence, process knowledge is identified in the theoretical framework.

Telliglu and Wagner [45] found that SE practice depends on context and culture. The issues relating to communication and coordination breakdowns in large design projects as identified by Curtis et al., [27] could be related to lack of cultural knowledge amongst the individuals. Cash et al., [46] identified that in the e-commerce era many competencies that are based on an understanding of culture were required. It was found that an understanding of business culture is required to perform the role of programmers [17]. Chan [47] found evidence for cultural, social norms and practice to performing various roles in enterprise system development.

Heuristics means a useful shortcut, an approximation or a rule of thumb for guiding search [48]. Vitalari and Dickson [36] provided evidence that some types of heuristic problem solving behaviours lead to better performance in systems analysts. Zmud's [49] categorisation of knowledge items related to the need to know the policies of the organisation some of which can be considered to be heuristics in nature

[50]. Chan [47] found that company specific policies and rules are important to performing various roles in Enterprise System development.

## B. Performance

WordNet defines performance, as the *“act of performing; of doing something successfully; using knowledge as distinguished from merely possessing it”* [51]. Individual performance is the efficiency with which an individual is able to deliver the expected outcomes [52]. Curtis et al. [27] provided evidence from a field study to the proposition that the thin spread of application domain knowledge and less social network knowledge affected performance. Rasch and Tosi [53] developed an integrated model for performance and demonstrated that individual intellectual KSA has the strongest direct effect on perceived performance. Wade and Parent [54] carried out a survey of Web masters and found that skill deficiencies in both organisational skills and technical skills lead to lower job performance. White and Leifer [55] obtained perceptions from systems professionals on skills and performance and found that skills such as technical skills, systems knowledge, business domain knowledge and soft skills have an impact on the overall performance. Thus, it has been demonstrated in the literature that knowledge, skills and ability contribute to performance [53-56].

Based on the literature review, several types of knowledge have been identified that are needed to perform SE roles. Based on these a theoretical framework is formulated (see Figure 1.).

## III. RESEARCH DESIGN

An exploratory study was designed to understand the contribution of different knowledge factors to improved performance after individuals join the software project teams. As part of a larger survey on knowledge needs, a sub-survey was designed to collect perceived relative contribution to performance of different knowledge factors, if performance improved at all. An open ended question in the questionnaire allowed for capturing any other type of knowledge factor that has implications for performance. Extensive qualitative data was also collected.

A non-probabilistic purposive sampling approach was used to collect data. The researcher being a consultant working in the SE industry for more than fifteen years, using his personal relationships was able to secure volunteers to collect data from fellow colleagues and professional contacts in their organisations.

## IV. RESULTS

A total of 178 responses were collected through the survey. Programmer-analysts (78) topped the list followed by Programmers (28) and business analysts (28). Nine respondents (5.05% of total) carried out combination roles and eleven (6.1%) carried out other roles. The number of participants with greater than 5 years of experience (121) outweighs the count with less than 5 years (49). More than half of respondents came from a Government/public sector background (57.8%) while others worked in private sector domains such as banking, telecom and media. The relevant SE systems were

implemented using a range of technologies and platforms that included .NET, Java, Mainframe, ORACLE and AS-400.

### A. Qualitative results

The survey contained a question asking the participants to provide one significant example of how the increase in knowledge (of different types) contributed to their increased performance. Sixty one comments were obtained. The analysis of qualitative data reveals support to the proposition that the different types of knowledge contribute to increase in performance. Following are some examples of comments:

- Business domain area knowledge: An understanding of business domain knowledge such as insurance particularly during requirements analysis can help in improved performance.
- Configuration knowledge: Overview and architecture of the system and its functionality of the system including business rules, technical components of the system, its database structures help in resolving problems, analyse impacts and in improved performance.
- Cultural knowledge: An understanding of cultural items through knowledge sharing sessions and interactions with senior management, an understanding of quality requirements for products, expectations from people on work outputs, impact to people have helped with increasing the performance.
- Heuristic knowledge: Management of templates and its use and implementing lessons learned from project management reviews have helped with improved performance
- Process knowledge: Better knowledge of auditing procedures, test processes and management, procedures for analysis has helped with improved performance.
- Social network knowledge: Improved team communication and knowledge sharing culture, expanded stakeholders, personal relationships, connecting people, closer working relationships, an understanding of point of contact helped improved performance. The following comment demonstrates the importance of social network knowledge to improved performance. ‘*Social Network knowledge – once I knew who did what within the organisation, ability to quickly resolve external problems which were affecting the application I supported improved dramatically.*’
- Technique skills: Improved technical skills in new or existing technologies contribute to improved performance.

Participants did not provide qualitative comments for 1) soft skills and 2) systems knowledge. Further, no other knowledge factor was identified through the open ended question in the questionnaire that allowed for capturing any other type of knowledge factor.

### B. Quantitative results

#### 1) Ranking of Knowledge types

For every knowledge factor identified in the theoretical framework (nine types of knowledge; see Figure 1. ), perceived contribution of increased knowledge to improved performance is captured. The mean perceived contribution to improved performance for the nine knowledge factors identified in the theoretical framework was computed and ranking assigned. It is found that (see TABLE I) increases in 1) Technique skills (specialised skills to carry out the job) and 2) Configuration knowledge (the knowledge of the application system) are the two ranking important factors contributing to improved performance in Software Engineering. This is followed by increases in systems knowledge, business domain knowledge, soft skills and social network knowledge respectively. Increase in cultural knowledge contributes least to improvements in performance. Increases in Process knowledge and heuristics knowledge rank 7 and 8 respectively.

Based on the quantitative and qualitative results, it can be concluded that all the knowledge factors identified in the theoretical framework contribute to improved performance. As per the rankings, the increase in technique skills contributed most to improved performance.

TABLE I. RANKING BASED ON MEAN CONTRIBUTION

Rank# <sup>⊙</sup>	Type-of-knowledge <sup>⊙</sup>	Mean-perceived-contribution-to-improved-performance <sup>⊙</sup>
1 <sup>⊙</sup>	Increase-in-Technique-skills <sup>⊙</sup>	2.15 <sup>⊙</sup>
2 <sup>⊙</sup>	Increase-in-Configuration-knowledge <sup>⊙</sup>	2.12 <sup>⊙</sup>
3 <sup>⊙</sup>	Increase-in-Systems-knowledge <sup>⊙</sup>	2.04 <sup>⊙</sup>
4 <sup>⊙</sup>	Increase-in-Business-domain-knowledge <sup>⊙</sup>	2.01 <sup>⊙</sup>
5 <sup>⊙</sup>	Increase-in-Soft-skills <sup>⊙</sup>	2.00 <sup>⊙</sup>
6 <sup>⊙</sup>	Increase-in-Social-network-knowledge <sup>⊙</sup>	1.95 <sup>⊙</sup>
7 <sup>⊙</sup>	Increase-in-Process-knowledge <sup>⊙</sup>	1.77 <sup>⊙</sup>
8 <sup>⊙</sup>	Increase-in-Heuristics-knowledge <sup>⊙</sup>	1.68 <sup>⊙</sup>
9 <sup>⊙</sup>	Increase-in-Cultural-knowledge <sup>⊙</sup>	1.66 <sup>⊙</sup>

#-Rank based on mean contribution to improved performance<sup>⊙</sup>

-Scale used 0 = nil, 1 = small, 2 = moderate, 3 = great<sup>⊙</sup>

#### 2) Role and Contribution of Configuration knowledge to improvements in Performance

After analysing using ANOVA between Role and contribution of different knowledge factors, it was found that Role has a statistically significant ( $p < 0.030$ ) relation with Contribution of configuration knowledge to improvements in performance (see TABLE II). However, ANOVA of pair-wise comparisons revealed no significant relationships between specific roles and *Contribution of configuration knowledge to increase in performance*. Further analysis of means (see TABLE III) reveal that the *Contribution of configuration knowledge to improved performance* is the highest for architects. This is followed by both programmers, programmer-analysts, designers and business analysts. The contribution of configuration knowledge to improvements in performance is the lowest for testers and project managers. Such a variation in average contributions can be expected due to differing responsibilities of different roles [57].

Thus, this finding shows that while application system knowledge contributes to improved performance for all roles, some roles appear to have a higher need than other roles. Hence KM activities should target configuration knowledge improvements for programmers, programmer-analysts, and business analysts.

TABLE II. SIGNIFICANT RELATION BETWEEN ROLE AND CONTRIBUTION OF CONFIGURATION KNOWLEDGE TO INCREASE IN PERFORMANCE

Dependent Variable		Sum of Squares	df	Mean Square	F	Sig.
C29B Contribution of configuration Knowledge To inc. in performance	Contrast	23.071	21	1.099	1.757	.029
	Error	85.659	137	.625		

The F tests the effect of A6.Role. This test is based on the linearly independent pair wise comparisons among the estimated marginal means.

TABLE III. MEANS CONTRIBUTION OF CONFIGURATION KNOWLEDGE TO INCREASE IN PERFORMANCE FOR DIFFERENT ROLES

A6.Role	Mean	Std. Error	95% Confidence Interval	
			Lower Bound	Upper Bound
1-BusinessAnalyst	2.043	.164	1.719	2.368
2-Programmer	2.240	.158	1.928	2.552
3-Programmer/Analyst	2.203	.095	2.015	2.390
4-Architect	2.667	.455	1.767	3.566
5-Designer	2.200	.352	1.503	2.897
6-Tester	1.750	.394	.971	2.529
7-ProjectManager	1.769	.219	1.337	2.201

### C. Limitations

Several limitations exist in regard to the general-survey research. These are listed below.

- Perception only – This research collects the perceived contributions to performance from individuals. Such perceptions have been used in the literature [18, 20, 40, 54] to capture knowledge requirements and are found to be a practical way to measure knowledge needs. These perceptions on knowledge may or may not reflect reality.
- Tacit knowledge: The inarticulable tacit component of knowledge present within individuals cannot be captured using perception studies on individuals. This research may not have captured this tacit knowledge adequately.
- Non-probabilistic: The sampling methodology used in this study is purposive-judgemental sampling approach. This is a non-probabilistic sampling methodology. Hence, the results of the study cannot be generalised to a wider population.
- The survey collects opinions at a certain point in time and reports the results. The performance increases can change in the future due to various reasons (example: technological change, changes to role

definitions). The survey cannot predict future.

### V. CONCLUSIONS

The research ranked the knowledge types in terms of their contribution to improvements in performance. Increased Technique skills and Configuration knowledge rank as the top contributors to improved performance after individuals join the SE team. It was found that role has a statistically significant ( $p < 0.030$ ) relation with Contribution of Configuration knowledge to improvements in performance. These findings allow smarter SE practice by carrying out targeted KM initiatives. For instance, imparting Technique skills and Configuration knowledge can be taken up as a priority for newcomers. Practitioners can take up the challenge of equipping themselves with these types of knowledge upon joining new SE teams. Experience Factories [58] that include experience packages such as product packages, and technique skill packages (example: tool packages) need to be developed and implemented from a KM viewpoint. Through such increases in knowledge to individuals in SE teams, it is possible to improve performance, thus increasing success rates and reducing failures.

### REFERENCES

- [1] T. Davenport, and L. Prusak, *Working Knowledge : How Organizations manage what they know ?*, Boston, MA.: Harvard Business School Press, 1998.
- [2] C. R. de Souza, S. Quirk, E. Trainer *et al.*, "Supporting collaborative software development through the visualization of socio-technical dependencies," in Proceedings of the 2007 international ACM conference on Supporting group work, Sanibel Island, Florida, USA, 2007.
- [3] T. Dingsoyr, "Knowledge management in medium-sized software consulting companies," Computer Science, Norges Teknisk-Naturvitenskapelige Universitet, Norway, 2002.
- [4] R. L. Glass, "The relationship between theory and practice in software engineering," *Commun. ACM*, vol. 39, no. 11, pp. 11-13, 1996.
- [5] P. N. Robillard, "The role of knowledge in software development," *Commun. ACM*, vol. 42, no. 1, pp. 87-92, 1999.
- [6] P. N. Robillard, P. d'Astous, D. t. Franoise *et al.*, "Measuring cognitive activities in software engineering," in Proceedings of the 20th international conference on Software engineering, Kyoto, Japan, 1998.
- [7] Y. Ye, "Supporting software development as knowledge-intensive and collaborative activity." p. 15.
- [8] A. Forward, and T. Lethbridge, "The relevance of software documentation, tools and technologies: a survey," in Proceedings of the 2002 ACM symposium on Document engineering, McLean, Virginia, USA, 2002.
- [9] L. P. W. Land, A. Aurum, and M. Handzic, "Capturing Implicit Software Engineering

- Knowledge,” in Australian Software Engineering Conference, Canberra, Australia, 2001.
- [10] J. Parsons, and C. Saunders, “Cognitive Heuristics in Software Engineering: Applying and Extending Anchoring and Adjustment to Artefact Reuse,” *IEEE Transactions on Software Engineering* vol. 30, no. 12, pp. 873-888, 2004.
- [11] B. Philp, and B. Garner, “Knowledge Mediation in Software Quality Engineering,” in Australian Software Engineering Conference, Canberra, Australia, 2001.
- [12] K. Ewusi-Mensah, *Software Development Failures*, Boston, MA, USA: MIT Press, 2003.
- [13] T. DeMarco, *Controlling Software Projects : Management, Measurement and Estimation*, New York, NY: Yourdon Press, 1982.
- [14] R. Baskerville, J. Pries-Heje, and B. Ramesh, “The enduring contradictions of new software development approaches: a response to 'Persistent Problems and Practices in ISD',” *Information Systems Journal*, vol. 17, no. 3, pp. 241-245, Jul, 2007.
- [15] R. S. Pressman, *Software engineering : a practitioner's approach*, 7th ed., New York: McGraw-Hill Higher Education, 2009.
- [16] H. Beazley, J. Boenisch, and D. Harden, *Continuity Management : Preserving Corporate Knowledge and Productivity when employees leave*, Hoboken, N.J.: John Wiley & Sons, Inc, 2002.
- [17] J. Bailey, and G. Stefaniak, “Industry perceptions of the knowledge, skills, and abilities needed by computer programmers,” in Proceedings of the 2001 ACM SIGCPR conference on Computer personnel research, San Diego, California, United States, 2001.
- [18] D. M. S. Lee, E. M. Trauth, and D. Farwell, “Critical skills and knowledge requirements of IS professionals: A joint academic/industry,” *MIS Quarterly*, vol. 19, no. 3, pp. 313, 1995.
- [19] D. J. McCubbray, “The systems analyst of the 1990's,” in Proceedings of the ACM SIGCPR conference on Management of information systems personnel, College park, Maryland, United States, 1988.
- [20] R. R. Nelson, “Educational Needs as Perceived by IS and End-User Personnel: A Survey of Knowledge and Skill Requirements,” *MIS Quarterly*, vol. 15, no. 4, pp. 503, 1991.
- [21] M. J. Gallivan, D. Truex III, and L. Kvasny, “Changing patterns in IT skill sets 1988-2003: a content analysis of classified advertising,” *SIGMIS Database*, vol. 35, no. 3, pp. 64-87, 2004.
- [22] C. R. Litecky, K. P. Arnett, and B. Prabhakar, “The paradox of soft skills versus technical skills in IS hiring,” *Journal Of Computer Information Systems*, vol. 45, no. 1, pp. 69-76, Fal, 2004.
- [23] S. Nansi, “Critical information systems management issues frameworks and relationships with individual IS executives and organisational environments: A study in Singapore,” Ph.D thesis, International Graduate School of Management, University of South Australia, Adelaide, 1998.
- [24] B. Prabhakar, C. Litecky, and K. Arnett, “IT skills in a tough job market,” *Commun. ACM*, vol. 48, no. 10, pp. 91-94, 2005.
- [25] P. A. Todd, J. D. McKeen, and R. B. Gallupe, “The evolution of IS job skills: A content analysis of IS job advertisements from 1970 to 1990,” *MIS Quarterly*, vol. 19, no. 1, pp. 1, 1995.
- [26] D. B. Walz, J. Elam, and B. Curtis, “Inside a software design team: knowledge acquisition, sharing, and integration,” *Commun. ACM*, vol. 36, no. 10, pp. 63--77, 1993.
- [27] B. Curtis, H. Krasner, and N. Iscoe, “A field study of the software design process for large systems,” *Commun. ACM*, vol. 31, no. 11, pp. 1268--1287, 1988.
- [28] T. M. Kantrowitz, “Development and Construct Validation of a Measure of Soft Skills Performance,” Psychology, Georgia Institute of Technology, 2005.
- [29] N. Evans. "The Need for an Analysis Body of Knowledge (ABOK) - Will the Real Analyst Please Stand Up ?," 1 Jan 2009, 2009; <http://proceedings.informingscience.org/InSITE2004/053evans.pdf>.
- [30] K. Schwalbe, *Information Technology Project Management*, Massachusetts: Course Technology Thomason Learning, 2002.
- [31] B. Ives, and M. H. Olson, “Manager or Technician? The Nature of the Information Systems Manager's Job,” *MIS Quarterly*, vol. 5, no. 4, pp. 49, 1981.
- [32] The Open group. "TOGAF Architecture Skills Framework," <http://www.opengroup.org/architecture/togaf8-doc/arch/chap30.html>.
- [33] K. Frampton, J. Thom, J. Carroll *et al.*, “Information technology architects: approaching the longer view,” in Proceedings of the 2006 ACM SIGMIS CPR conference on computer personnel research: Forty four years of computer personnel research: achievements, challenges \& the future, Claremont, California, USA, 2006.
- [34] G. Hunter, “Excellent systems analysts: key audience perceptions,” *SIGCPR Comput. Pers.*, vol. 15, no. 1, pp. 15-31, 1994.
- [35] G. Hunter, and S. Palvia, “Ideal, advertised and actual systems analyst skills: the Singapore context,” *Information Technology & People*, vol. 9, no. 1, 1996.
- [36] N. Vitalari, and G. Dickson, “Problem solving for effective systems analysis: an experimental exploration,” *Commun. ACM*, vol. 26, no. 11, pp. 948-956, 1983.
- [37] B. Curtis, D. Walz, and J. Elam, “Studying the process of software design teams,” in Proceedings of the 5th international software process workshop on

- Experience with software process models, Kennebunkport, Maine, United States, 1990.
- [38] J. D. Herbsleb, and E. Kuwana, "Preserving knowledge in design projects: what designers need to know," *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 7--14: ACM Press, 1993.
- [39] B. Adelson, and E. Soloway, "The Role of Domain Experience in Software Design," *IEEE Transactions on Software Engineering*, vol. 11, no. 11, November-1985, 1985.
- [40] S. Nansi, and D. Bennett, "Requisite IS management knowledge and skills construct: a survey," *SIGCPR Comput. Pers.*, vol. 19, no. 1, pp. 3--19, 1998.
- [41] W. S. Humphrey, *A discipline for Software Engineering*: Addison Wesley Longman (Singapore) Pte Ltd, Indian Branch, 1995.
- [42] A. Wasserman, "Toward a discipline of software engineering," *IEEE Software*, vol. November, no. 23-27, 1996.
- [43] Software Engineering Institute. "Process Maturity Profile : CMMI v.1.1 Class A Appraisal Results," 09 Sep 2004, 2004; <http://www.sei.cmu.edu/sema/pdf/CMMI/2004aug.pdf>.
- [44] ISO-SPICE. "What is SPICE?," 20th May 2008, 2008; <http://www.sqi.gu.edu.au/spice/what.html>.
- [45] H. Telliglu, and I. Wagner, "Software Cultures : Exploring cultural practices in managing heterogeneity within system design," *Commun. ACM*, vol. 42, no. 12, pp. 71-77, December 1999, 1999.
- [46] E. Cash, P. Yoong, and S. Huff, "The impact of e-commerce on the role of IS professionals," *SIGMIS Database*, vol. 35, no. 3, pp. 50-63, 2004.
- [47] R. Chan, "Structuring and modelling knowledge in the context of enterprise systems," Centre for Information Technology Innovation, Queensland University of Technology, 2003.
- [48] G. Gigerenzer, and P. M. Todd, *Simple heuristics that make us smart*, Oxford: Oxford University Press, 1999.
- [49] R. Zmud, *Information Systems in Organizations*, Oakland, New Jersey: Scott, Foresman and Company, 1983.
- [50] P. A. Busch, D. Richards, and C. N. G. Dampney, "Visual mapping of articulable tacit knowledge."
- [51] Princeton University. "Performance. (n.d)," <http://dictionary.reference.com/browse/performance>.
- [52] Macquarie Online Dictionary, "Macquarie Online Dictionary," 2008.
- [53] R. H. Rasch, and H. L. Tosi, "Factors affecting software developers' performance: An integrated approach," *MIS Quarterly*, vol. 16, no. 3, pp. 395, 1992.
- [54] M. R. Wade, and M. Parent, "Relationships Between Job Skills and Performance: A Study of Webmasters," *Journal of Management Information Systems*, vol. 18, no. 3, pp. 71, 2001.
- [55] K. B. White, and R. Leifer, "Information Systems Development Success: Perspectives from Project Team Participants," *MIS Quarterly*, vol. 10, pp. 215-223, 1986.
- [56] R. W. Zmud, "Individual Differences and MIS Success: A review of the empirical literature," *Management Science*, vol. 25, no. 10, pp. 966, 1979.
- [57] ACS. "ICT Careers Portal," 10th Jan 2008, 2008; <http://www.acs.org.au/ictcareers/index.cfm?action=list&sgrID=200707131235515720>.
- [58] V. Basili, and F. McGarry, "The experience factory: how to build and run one (tutorial)," in *Proceedings of the 19th international conference on Software engineering*, Boston, Massachusetts, United States, 1997.

# Lattice-Context Based Digital Paper Search

Chongyang Shi,Zhendong Niu  
School of Computer Science  
Beijing Institute of Technology  
Beijing,China  
Email: {happysprite,zniu}@bit.edu.cn

Xiyi Cheng  
Jianda Aopeng Education and Science  
Beijing,China  
Email: xy21wj@126.com

## Abstract

*Based on concept lattice, this paper proposes a new digital paper search paradigm that ranks search outputs, while controlling the diversity of keyword-based search query output topics. During pre-querying, papers are assigned into pre-specified lattice-based contexts patterns, and query-independent context scores are attached to papers with respect to the assigned contexts. When a query is executed, relevant contexts are selected, search is performed within the selected contexts, context scores of papers are revised into relevancy scores with respect to the query and the context that they are in, and query outputs are ranked within each relevant context. Our experiments indicate that the proposed lattice context-based search approach produces search results with up to 50% higher precision, and reduces the query output size by up to 60% than CNKI search.*

## 1 Introduction

In order to (i) minimize the scope of query in the large numbers of papers, and (ii) provide controlled ways of eliminating query output topic diversity, (iii) present a new method which can effectively rank query output papers, we propose a new digital paper search paradigm, called Formal concept analysis-Based Search (FBS) approach, which is a context based search model using lattice, as follows:

1. We perform two query-independent pre-processing steps before any query session starts: assign papers into lattice-based contexts;and compute lattice (importance) scores for papers. Therefore, each lattice context contains two types of information: (i) the intrinsic lattice information of paper set(object) and the attribute set owned by the paper set and (ii) the context score of each paper.
2. Then, at search time, we perform the following steps:

- (a) Select search lattice-contexts automatically
- (b) Perform keyword-based search within the selected contexts
- (c) Within each context, compute relevancy scores of located papers, re-rank search results, and return the located papers

With the FBS approach, (i) search input includes only papers residing in the selected lattice-contexts as opposed to all papers (ii) search output is enhanced by a highly useful context-based paper classification, (iii) topic diffusion across search results is controlled, and (iv) query output sizes are reduced to include only search results in the contexts of interest.

Since the FBS approach performs a search within selected lattice contexts, some important results might be missing if they are not in the selected contexts. As an alternative, step 2 of the FBS approach can be modified to include all search results (FBS\_all) as follows:

- (a) Select search contexts automatically
- (b) Perform keyword-based search across all papers to select the publications to be returned
- (c) For the returned papers that reside in the selected contexts, compute relevancy scores of these papers in each context.
- (d) Re-rank search results and return the located papers

## 2 Classifying Papers with Scores to Lattice-contexts

This section presents an approach to automatically locate papers of contexts. The approach constructs lattice context patterns from training data set and uses those patterns to locate the paper set of the context.

## 2.1 Lattice Context Patterns Building and Paper's Context Assignment

Papers acting as object items would be assigned to lattice context one by one after the course of patterns building, so here we use the incremental lattice building algorithms[6]. And when new papers are added to the lattice patterns, according to the increment lattice building way, the patterns of contexts should be updated. To avoid this problem of updating and ensure all the papers included by lattice contexts, we must train partial dataset from all based on which stable patterns are constructed. And while new papers are added to the stable patterns, all contexts could hold the line.

## 2.2 Assigning Lattice Context Scores to Papers

The lattice context score of a paper in each context is computed using text-based similarity measures based on the Term Frequency-Inverse Document Frequency (TFIDF) [5] model. In each context  $c$ , a paper  $p$ 's context score is defined as the text-based similarity score between context's centroid and  $p$ . The centroid of context is computed by averaging all papers' score in context. In other words, papers in context that are highly similar to the centroid of  $c$  receive high context scores.

Based on the papers of a context, one centroid is constructed. After counting the occurrences of the term in each paper, our approach averages all the occurrences score which is stored as an attachment within the context. The average occurrences score of centroid is defined as

$$Avg\_Occurrences\_Score = \frac{\sum_M n_i}{N * M} \quad (1)$$

where  $n_i$  is the number of times the context term appears in the paper  $i$ , and  $N$  is the length of a vector representing the paper.  $M$  is the number of the papers in the context. Then in our case, the context score of  $p$  in  $c$  is computed as

$$Score(p) = sim(p_c, p) \quad (2)$$

Where  $p_c$  is the centroid of  $c$ , and  $sim(p_c, p)$  is the text-based similarity between  $p$  and  $p_c$ .

Since lattice contexts are represented hierarchically, a paper  $p$  can reside in both context  $c_i$  and  $c_i$ 's descendant contexts. Compared to  $c_i$ ,  $c_i$ 's descendant contexts are more specific, and the descendant contexts' paper sets are less diverse. Hence, a high context score for  $p$  in  $c_i$ 's descendant contexts means that  $p$  is highly relevant to  $c_i$ . Therefore, a final score computation step takes place as follows. Let  $p$  reside in context  $c_i$  with score  $s_i$ , and descendant contexts  $c_k \dots c_n$  of  $c_i$  with scores  $s_k, \dots, s_n$ . Then  $p$ 's score in context  $c_i$  is modified as  $max(s_j), j \in i, k, \dots, n$ .

## 3 Selecting Contexts for Keyword Search

A context-based search query maybe any set of keywords. After mapping a given query to a set of query contexts, we perform the search and rank search results within these contexts. In literature[3], we introduce a novel similarity evaluating model based on rough formal concept analysis and information content similarity method. Given an arbitrary query term, it can therefore be viewed as an undefinable set of attributes in formal concept. Following the theory of rough set, such a set of attributes can be approximated by definable set of attributes, namely, the intensions of formal concepts. Since lower approximation is the greatest definable set of the concept, we build our similarity measure on the lower approximation. And the similarity model between the term and the context  $c$  which is express by lattice information  $(O, D)$  based on lower attribute approximations is

$$Sim(q, (O, D)) = \omega \frac{|(q_{LO} \cap O)|}{|(q_{LO} \cap O)| + (m_a - l_a)} + (1 - \omega) \frac{|(q_{LA} \cap D)|}{|(q_{LA} \cap D)| + (n_a - r_a)} \quad (3)$$

You can find more details of the similarity model in [3].

Finally, given a query term  $q$ , and using the above similarity model, then those contexts(formal concepts) with sufficiently high similarity higher than a threshold  $t$  are selected to be the query lattice contexts of  $q$ .

## 4 Search and Rank Search Results

Search results returned from the lattice-based search are ranked by their relevancy scores with respect to the context and the query term. The relevancy score of paper  $p$  to query  $q$  in context  $c_i$  is computed as

$$R(p, q, c_i) = w_{context} \cdot Context\_Score(p, c_i) + w_{matching} \cdot Text\_Matching\_Score(p, q) \quad (4)$$

where  $Context\_Score(p, c_i)$  is the context score of  $p$  in context  $c_i$ ,  $Text\_Matching\_Score(p, q)$  computes the similarity between  $p$  and  $q$ , and  $w_{context}$  and  $w_{matching}$  are weights of the context score and the text matching score, respectively.  $w_{context} + w_{matching} = 1$ . By default, we define  $w_{matching} > w_{context}$  (i.e., we used  $w_{matching} = 0.8$  and  $w_{context} = 0.2$  in the experiments). In this definition, the text-matching scores between the query keyword and the papers are considered more important than the context scores of the papers. However, the weights can be adjusted based on users' preference. For example, if the user wants to increase the significance of the contexts,  $w_{matching}$  will be reduced while  $w_{context}$  will be increased, and search results within the contexts will be ranked with respect to the new weights.

But users may want to view a single result set independent of the individual searched contexts. To effectively rank search results for the latter case, scores of a paper residing in multiple contexts need to be merged into a final score. When appearing in multiple contexts, paper p’s overall relevancy score  $R(p,q)$  to the query  $q$  is computed using (1) the relevancy score of  $p$  to  $q$  in each context, and (2) the relevancy of each context containing  $p$  to  $q$ , as follows:

$$R(p, q) = \frac{\sum_{i=1}^{n_p} (w_{PaperRelevancy} R_1(p, q, c_i) + w_{context} R_2(c_i, q))}{n_p} \quad (5)$$

where  $R_1(p, q, c_i)$  is the relevancy score of  $p$  to  $q$  in the context  $c_i$ ,  $R_2(c_i, q)$  is the relevancy score of the context  $c_i$  to the query  $q$ ,  $n_p$  is the number of contexts that contain  $p$ ,  $w_{PaperRelevancy}$  and  $w_{context}$  are the weights of  $R_1$  and  $R_2$ , respectively,  $w_{PaperRelevancy} + w_{context} = 1$ . We define  $w_{PaperRelevancy} > w_{context}$  (i.e., we used  $w_{PaperRelevancy} = 0.6$  and  $w_{context} = 0.4$  in the experiments).

## 5 Experimental setup

We downloaded, parsed, and populated our database with information from 10000 full-text CNKI[2] papers. All selected papers came from the computer science area of information retrieval. Fifty top frequent keywords extracted from the keyword part of these papers were selected as attributes to construct paper’s concept lattice. Based on trained papers’ information of all, we built the stable lattice patterns, and then the rest papers were added to these lattice context patterns one by one. And to evaluate the accuracy of the context-based search approach, recall and precision scores of selected queries were used. In addition to recall and precision, we used the harmonic mean F1, which combines recall and precision. F1 is defined as

$$F1_t = \frac{2}{\frac{1}{Recall_t} + \frac{1}{Precision_t}} \quad (6)$$

### 5.1 AB-evaluating set

Here, we developed an approach to find the Artificially Built-evaluating set of a query automatically. The AB-evaluating set was used to evaluate queries with no human judgments of their search results. Through domain expert evaluations of a small number of queries, we refined the AB-evaluating set creation process, and manually verified its correctness. Then the AB-evaluating set was used extensively in the experiments to evaluate search query recall and precision scores.

#### 5.1.1 AB-evaluating set construction

To construct an AB-evaluating set, we use an approach similar to the pearl-growing search strategy [4]. This approach expands the AB-evaluating set with citations of a paper in the initial answer set  $S1$ . Since a paper usually cites or is cited by other papers that are relevant to it, citations of a paper in  $S1$  are potentially relevant to the query term. There are two approaches involving the citation based expansion.

1. **Text-based expansion:** This approach uses the text-based similarity measure to locate additional papers. Since a paper in  $S1$  is highly relevant to the keyword query, papers that are cited to the paper are potentially relevant to the query. Thus, papers cited with high similarity scores to  $p$  are added to the AB-evaluating set.
2. **Citation-similarity-based expansion:** Citation similarity [1] is computed as follows:

$$Sim_{Citation}(p1, p2) = BibWeight * Sim_{bib}(p1, p2) + (1 - BibWeight) * Sim_{coc}(p1, p2) \quad (7)$$

where  $p1$  and  $p2$  are papers,  $p1 \in S1$ ,  $p2 \notin S1$ ,  $Sim_{bib}$  is the bibliographic coupling score,  $Sim_{coc}$  is the co-citation score,  $BibWeight$  is the bibliographic coupling weight,  $CocWeight = 1 - BibWeight$  is the co-citation weight, and  $0 \leq BibWeight \leq 1$ .  $Sim_{bib}$  is defined as

$$Sim_{bib}(p1, p2) = \frac{k_1}{M_b} \quad (8)$$

where  $k_1$  is the number of common citations between  $p1$  and  $p2$ , and  $M_b$  is the maximum number of common citations between any pair of papers in the database.

$Sim_{coc}$  is defined as:

$$Sim_{coc}(p1, p2) = \frac{k_2}{M_c} \quad (9)$$

where  $k_2$  is the number of papers co-cite  $p1$  and  $p2$ , and  $M_c$  is the maximum number of papers that co-cite any pair of papers in the database.

## 5.2 Experimental results

In this section, we compare recall, precision, and harmonic mean of recall and precision (F1) when performing different search approaches.

### 5.2.1 Comparing context-based results against CNKI results

Here we compare recall and precision scores from the FBS approach to CNKI’s general keyword-based search. Papers

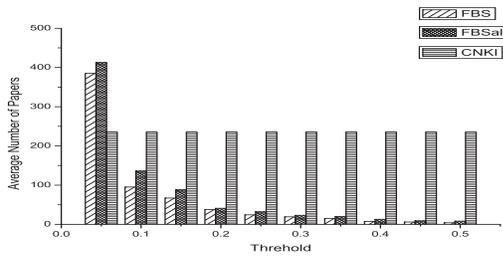


Figure 1. The average number of papers returned from CNKI, FBS, and FBS\_all

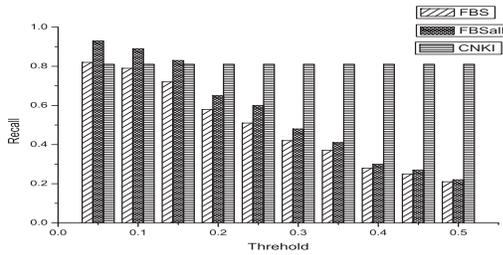


Figure 2. Average recall of CNKI, FBS, and FBS\_all

that are in CNKI search results but not in our database are filtered out before evaluations. Figure 1 shows the average number of papers from CNKI, FBS, and FBS\_all approaches. Figure 2 and 3 compares the average recall and precision scores of CNKI, FBS, and FBS\_all approaches. Figure 4 illustrates the average F1 scores of the three approaches.

Experimental results show the following:

(1) At  $t > 0.10$ , CNKI recall is higher than the context based (FBS and FBS\_all) recall. This is due to CNKI

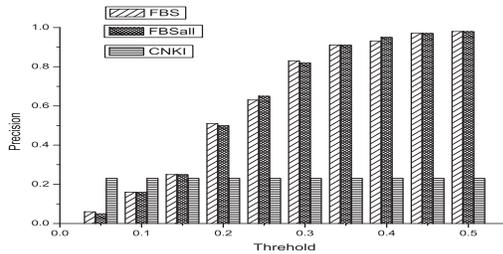


Figure 3. Average precision of CNKI, FBS, and FBS\_all

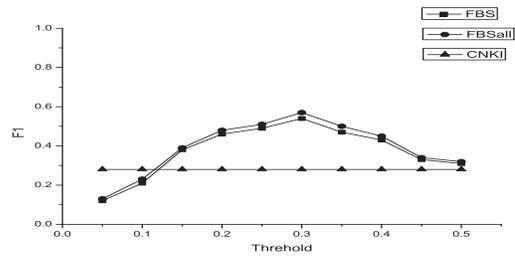


Figure 4. Average F1 scores of CNKI, FBS, and FBS\_all

searching and returning more papers on average than the context-based search approaches.

(2) The context-based approaches produce approximately 50% higher precision at high thresholds and approximately 25% higher precision at moderate thresholds.

(3) At moderate thresholds, the context-based approaches yield approximately 25% higher F1 scores than CNKI. Moreover, from Figure 1, the number of context based search results at moderate thresholds are much smaller (approximately 10 times) than CNKI search results.

(4) The FBS approach reduces the query output size by up to 60% as compared to the CNKI.

## 6 Conclusion

Trough our way, we (1) minimize query output topic diversity, (2) reduce query output size,(3) decrease user time spent scanning query results, and (4) increase query output ranking accuracy. Using CNKI publications as the testbed, our experiments indicate that the proposed lattice context-based search approach produces search results with up to 50% higher precision, and reduces the query output size by up to 60% than CNKI search.

## References

- [1] A.Al-Hamdani. Querying web resources with metadata in a database. *PHD Dissertation, CWRU*, 2004.
- [2] CNKI. <http://www.cnki.net/>.
- [3] C.Y.Shi. Combining ics semantic factor into concept similarity evaluating based on rfca. *iiWAS 2009*, 2009.
- [4] R. D.T.Hawkins. Online bibliographic search strategy development. *Online*, 1982.
- [5] G.Salton and C.Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage. Int. J.*, 5(24):613–620, 1988.
- [6] R. R.Godin and H.Alaoui. Incremental concept formation algorithms based on galois lattices. *Computation Intelligence*, 1995.

# Evaluating the Weighted Sum Algorithm for Estimating Conditional Probabilities in Bayesian Networks

Simon Baker<sup>1</sup>    Emilia Mendes<sup>2</sup>

The University of Auckland  
 Computer Science Department  
 Private Bag 92019, Auckland, New Zealand  
 {<sup>1</sup>sbak030@aucklanduni.ac.nz, <sup>2</sup>emilia@cs.auckland.ac.nz}

**Abstract** —The primary challenge in constructing a Bayesian Network (BN) is acquiring its Conditional Probability Tables (CPTs). CPTs can be elicited from domain experts; however, they scale exponentially in size, thus making their elicitation very time consuming and costly. Das [1] proposed a solution to this problem using the weighted sum algorithm (WSA). In this paper we present two empirical studies that evaluates the WSA’s efficiency and accuracy, we also describe an extension for the algorithm to deal with one of its shortcomings. Our results show that the estimates obtained using the WSA were highly accurate and make significant reductions in elicitation.

*Bayesian Network; Conditional Probability; Weighted Sum Algorithm; Knowledge Elicitation, CPT Elicitation, Expert*

## I. INTRODUCTION

A Bayesian Network (BN) is a probabilistic modelling technique that allows for reasoning under uncertainty. BNs have been applied in many areas including: forecasting, estimation, classification, recognition, and inference [2, 3]. A BN consists of two components: The first is a Direct Acyclic Graph (DAG) that represents factors of interest (as nodes) and associated causal relations (as edges). For example, Figure 1 shows a naive BN for forecasting the rate of growth for a hypothetical plant, given the amount of sunlight and water it receives.

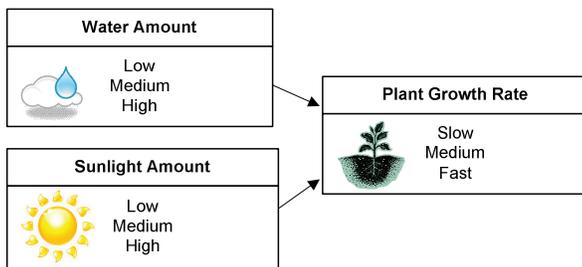


Figure 1. A naive Bayesian Network example

The second component of a BN is a set of Conditional Probability Tables (CPTs), one for each node. Table 1 shows the CPT for the *Plant Growth Rate* node from Figure 1. The left side of the CPT lists all possible configurations of its parents’ states, and the right side of the CPT contains conditional probability values for each configuration. Every row in the child nodes’ CPT must contain values that sum to exactly one.

A BN’s DAG and CPTs can be automatically learnt from data [4], or by domain expert elicitation [5, 6], where experts knowledgeable in a given domain are asked to visualize the

relations and factors that constitute the causal structure of the model, as well as the probability values required for the CPTs, or using a combination of both. Domain expert elicitation is often favoured when there is no data available for automatic learning, or when the model is qualitative in nature, and hence, difficult to record measurable data [6, 7].

TABLE I. CPT FOR THE "PLANT GROWTH RATE" NODE

CPT for 'Plant Growth Rate'				
Water Amount	Sunlight Amount	Slow	Medium	Fast
Low	Low	0.85	0.10	0.05
Low	Med	0.20	0.50	0.30
Low	High	0.22	0.48	0.30
Med	Low	0.50	0.25	0.25
Med	Med	0.25	0.60	0.15
Med	High	0.23	0.49	0.28
High	Low	0.30	0.50	0.20
High	Med	0.40	0.45	0.15
High	High	0.00	0.10	0.90

Although domain expert elicitation is considered as a more pragmatic approach, the reality is that eliciting thousands of probabilities manually becomes exceedingly arduous and time consuming. CPTs grow exponentially with respect to the number of parental states. Therefore, even a seemingly small BN (e.g. under 20 nodes) can potentially contain very large CPTs.

## II. MOTIVATION

One of the key obstacles for BN practitioners is acquiring probability parameters for Conditional Probability Tables [6, 7]. Throughout the last three decades a number of techniques were proposed to help mitigate this problem. Pearl’s Noisy-OR Gate technique [8] is perhaps one of the more established techniques, followed by its generalization (Noisy-MAX Gate) [8, 9]. However, these techniques often make fundamental assumptions about the BN. For example, they assume that parent nodes must be independent, and that each node must have an “absent” state. In many domains, however, these assumptions cannot be satisfied. To remedy many of these constraints Das [1] proposed a technique known as the weighted sum algorithm (WSA). However, unlike the Noisy-OR and Noisy-MAX techniques, there are no empirical studies, as far as we know that assess the WSA’s efficiency and estimation accuracy.

Therefore, our aim is to empirically assess the WSA’s efficiency and estimation accuracy using as benchmark manual domain expert elicitation. The main contribution of this paper

is to provide empirical evidence on the use of the WSA for CPT probability generation. Another contribution is to propose an extension to the WSA to remedy a certain issue encountered during the application of the algorithm (detailed in Section III).

The remainder of this paper is organised as follows: The next Section introduces the WSA, followed by a description of the methodology used in the empirical assessment, our results and threats to their validity, and finally our conclusions.

### III. THE WEIGHTED SUM ALGORITHM

The WSA is based on two heuristics originally proposed by Kahnman and Tversky [10, 11]. The first states that the more cognitively accessible an event is, the more likely it is perceived to occur (known as the availability heuristic). The second heuristic is to mentally simulate a scenario to assess the ease with which different results are produced given an initial set of parameters and operating constraints (known as the simulation heuristic). Using as basis these two heuristics, Das introduces the notion of compatible parental configuration [1], to be used to elicit the more cognitively accessible scenarios from the experts, and later to generate the remaining CPT using a weighted sum calculation. To describe the weighted sum algorithm we will refer to the BN in Figure 2.

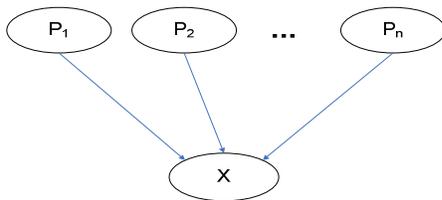


Figure 2. Generic naive Bayesian Network

We represent a node's individual states in superscript, so  $\{x^1, x^2, \dots, x^n\}$  are all the states of the child node  $x$ ; for the parent nodes, we'll use numbers in subscript to distinguish between different parents. For example  $\{p_1^1, p_1^2, \dots, p_1^n\}$  are all the states of the first parent  $P_1$ .

To formally define the notion of parental compatibility, let the parent  $P_i$  be assigned an arbitrary state  $p_i^v$  i.e.  $P_i = p_i^v$ , and let  $P_j$  be another parent, such that  $P_j$  is considered compatible with  $P_i = p_i^v$  only when  $P_j$  is in some state ( $p_j^w$ ) that is most likely, according to the expert's knowledge, to coexist with ( $P_i = p_i^v$ ). Therefore, we will use the notation  $Comp[P_i = p_i^v]$  to represent the set of states that are compatible with  $P_i = p_i^v$  for all parents.

$$Comp[P_i = p_i^v] = \{p_j^w, \forall j \neq i \mid \max_{w=1 \dots |p_j|} P(p_j^w | p_i^v)\} \quad (1)$$

The expert is asked to elicit the probabilities of compatible parental configurations for each state in each parent node (whenever the combination of states is meaningful (i.e. compatible) to the expert), and then the WSA algorithm is used to generate the probabilities of a child node's CPT. For example, if we refer to the BN in Figure 1, the expert might find it easier to estimate the probabilities for *Plant Growth Rate* for the case when the *Water Amount* is low, which would likely coincides when the *Sunlight Amount* is high, so therefore  $Comp[Water = Low]$  would contain the parental state  $Sunlight = High$ . Therefore, what makes this algorithm unique, compared to other techniques, is that it focuses on

asking experts questions that are easy to visualize and simulate because they relate to more realistic probabilities. Once this is done, then the algorithm generates the remaining probabilities that are likely harder to estimate.

However, an important assumption must be made when applying this algorithm: the expert must be capable of identifying realistic compatible parental configurations of each state for each parent. If the expert fails to do so, the algorithm will generate less trustworthy probabilities.

The algorithm takes as input:

1. The conditional probabilities corresponding to the compatible parental configurations for every parental state.
2. A relative weight value (between zero and one) for each parent node, denoting the degree of influence a parent has on a child node. The relative weights for all parent nodes must add up to exactly one. A relative weight equal to zero means that a parent has no influence at all over a child node, and can therefore be omitted from the network; conversely, a relative weight equal to one indicates that a parent node is the only determinant of the conditional probabilities in the child node.

The above inputs are then used as weighted sum corresponding to each state configuration in the child node's CPT, as follows:

$$P(X = x^i | p_1^a, p_2^b, p_3^c, \dots, p_n^z) = w_1 P(X = x^i | Comp[P_1 = p_1^a]) + w_2 P(X = x^i | Comp[P_2 = p_2^b]) + w_3 P(X = x^i | Comp[P_3 = p_3^c]) \dots + w_n P(X = x^i | Comp[P_n = p_n^z]) \quad (2)$$

for  $i = 1 \dots |x|$

Where  $w$  is the relative weight value for a parent, e.g.  $w_1$  is the relative weight value for  $P_1$ . To illustrate its usage, we refer back to our example presented in Figure 1 and Table I. To calculate an arbitrary cell in the CPT that has not been elicited, for instance, the probability that the *Plant Growth Rate* is *Fast* given that the *Water Amount* is *Low* and *Sunlight Amount* is *Medium*. Assuming that both parents are equally influential (i.e. both have an equal relative weight value of 0.5), then the weighted sum calculation for this cell is as follows:

$$P(GrowthRate = Fast | Water = Low, Sunlight = Medium) = 0.5 \times P(GrowthRate = Fast | Comp[Water = Low]) + 0.5 \times P(GrowthRate = Fast | Comp[Sunlight = Medium]) \quad (3)$$

If different parents' states share the same compatible parental configuration it is possible to reduce the number of elicitation questions due to this overlap. In addition, it also makes the computation of the algorithm more efficient; however, in most cases, this computation will be negligible. For example, if there are 5 parent nodes each with 3 states, but none of the states share any parental configuration, then 15 elicitation questions are needed to satisfy the first input of the algorithm, whereas if we take the opposite extreme, when there is an optimum compatible parental configuration overlap, only 3 questions are required. Although such optimum overlap is unlikely to be frequent, it is still beneficial to detect overlaps in order to reduce the elicitation effort.

There are circumstances when the domain expert is unable to choose exactly one compatible parental configuration for a given parental state assignment, i.e. there may be a set of compatible parental configurations for that given state. Das does not provide a solution for such situations. However,

during our experimentation with this method, we proposed a possible solution for this problem. We suggested an extension of the algorithm [12, 13] such that if a part of the given CPT configuration can be matched with a compatible parental configuration, then this value would be used (which is the default situation presented in Equation 2); otherwise, an average of all valid compatible parental configurations' probabilities would be used. Our proposal was later discussed with the WSA's author, who agreed with our suggestion.

To state the proposed extension formally, let  $\Omega[P_i = p_i^v]$  be a subset of  $Comp[P_i = p_i^v]$  such that it contains all of the compatible states that share the same parent, in other words, it is the set of states that have at least one other state from the same parent that is also compatible with  $P_i = p_i^v$ , i.e.:

$$\Omega[P_i = p_i^v] = \{\forall p_j^a \in Comp[P_i = p_i^v] \mid \exists p_j^b \in Comp[P_i = p_i^v]: a \neq b\} \quad (4)$$

And let  $\omega$  be any subset of  $\Omega[P_i = p_i^v]$  such that each state has a unique parent (i.e. set of states that do not share the parent with another state). We can define  $\omega$  as follows:

$$\omega = \{S \subset \Omega[P_i = p_i^v], S \neq \emptyset \mid \forall p_j^m, p_k^n \in S: j \neq k\} \quad (5)$$

We therefore extend the original notation for representing the set of compatible parental configurations for the parental assignment  $P_i = p_i^v$  (Equation 1) to be conditional on  $\omega$ ; we define this conditional parental configuration as the following:

$$Comp\left[\frac{P_i = p_i^v}{\omega}\right] = \{p_j^w \mid \max_{w=1 \dots |p_j|} P(p_j^w \mid p_i^v, \omega)\} \quad (6)$$

which is the set of all states that are compatible with  $P_i = p_i^v$  and all of the compatible parental assignments represented by  $\omega$ . Using this definition, we can extend the Weighted Sum Algorithm (Equation 2) to handle all situations when there are more than one compatible parental configurations for a given parental assignment, as follows:

$$P(X = x^i \mid Config) = \sum_{p_i^v \in Config} w_i f(p_i^v, Con) \quad (7)$$

where  $Config$  is a given configuration in the child CPT that we are generating the probability for,  $w_i$  is the weight associated with parent  $i$  and  $f(p_i^v, Config)$  is a conditional function defined as follows:

$$f(p_i^v, Config) = \begin{cases} P(X = x^c \mid Comp\left[\frac{P_i = p_i^v}{Config}\right]) & \text{If } Config \subseteq \Omega[P_i = p_i^v] \\ \frac{\sum_{\omega \in \Omega[P_i = p_i^v]} P(X = x^c \mid Comp\left[\frac{P_i = p_i^v}{\omega}\right])}{|\Omega[P_i = p_i^v]|} & \text{Otherwise} \end{cases} \quad (8)$$

*for c = 1 ... |x|*

In other words, if a part of the given configuration can be matched with a compatible parental configuration, then its elicited probabilities would be used, otherwise, an average of the closest (in similarity) compatible parental configuration probabilities would be used.

Note that our solution increases the number of probabilities elicited using the WSA method because for each additional parental state that is considered to be compatible for a given parent, the number of elicited probabilities using this method grows exponentially, which may approach manual elicitation in the extreme scenario where no compatible configurations are selected.

## IV. METHODOLOGY

The BNs used in our empirical assessment of the WSA method were elicited from experts in software and Web development. These BNs were built to forecast development costs for Web companies. Two Web effort estimation BNs were used, thus leading to two separate case studies. The following factors were fixed in each of these studies:

- A single domain expert participated in eliciting all required information. By fixing the domain expert we eliminated any potential discrepancies that could be introduced by multiple domain experts.
- An expert-driven BN model constructed using only manual elicitation was used as benchmark because it had already been validated using real data and it was confirmed by the domain expert(s) to fully reflect their beliefs. Therefore, the benchmark model was used as basis to measure the accuracy of the WSA algorithm.
- The input probabilities for the WSA algorithm were taken directly from the benchmark BN, and were not elicited again. This was done to avoid discrepancies in probabilities used in the models being compared (i.e. if probabilities were elicited twice, once using manual elicitation and once using the WSA algorithm, the expert could erroneously provide different probability values for exactly the same cell in the CPT). Therefore, the only inputs that were elicited for the WSA algorithm were the selection of the compatible parental configurations, and the weights for each parent.

We evaluated the weighted sum algorithm by:

- Determining reductions in elicited probabilities that it achieves. This is done by simply counting the required input probabilities required by the algorithm for a given CPT and then comparing it with the CPT size (i.e. what would be required if one would use manual elicitation).
- Measuring the accuracy of the algorithm by calculating the absolute error (Euclidean distance) between every cell in the generated CPT and the reference CPT in the benchmark BN model. Here we used the same technique previously used to evaluate CPT generation techniques (e.g. [14]). The Wilcoxon signed ranked test was employed to test the statistical significance of the results ( $\alpha = 0.05$ ). A non-parametric test was chosen because we could not guarantee that the data from some of the CPTs in the benchmark BN were normally distributed.

## V. RESULTS

In this section we summarise the results for the two case studies conducted.

### Case Study 1

The benchmark model contains 15 nodes, where four were considered to be nontrivial in terms of probability parameters (highlighted in Figure 3) and were therefore used as reference CPTs. A CPT is considered as nontrivial whenever it has at least two parent nodes, each presenting at least two states [12, 15, 16]. The sizes of the four CPTs are listed in Table II.

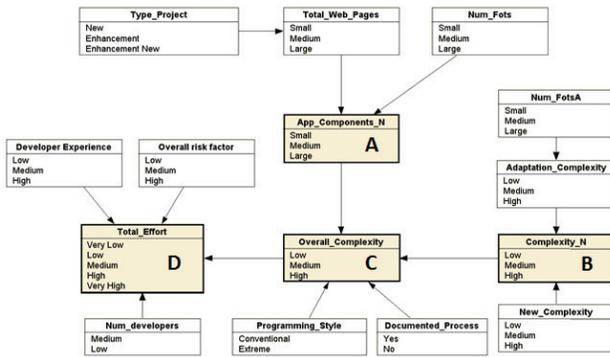


Figure 3. Benchmark Bayesian Network model for case study 1

TABLE II. THE SIZE OF CPTs IN CASE STUDY 1

CPT	Size	Number of Parents
A	27	2
B	27	2
C	108	4
D	270	4

The domain expert was confident in identifying the compatible parental configurations for CPTs A, and B. However this was not the case for the remaining CPTs, given that the expert was unable to select exactly one compatible parental configuration for most parent states in CPT C, and to a lesser extent in CPT D. we therefore applied our proposed extension (described in Section III). We elicited all computable parental configurations that the expert could identify for a given parental state, and later averaged their probability values. The percentage reduction in probability elicitation for each of the four CPTs (see Figure 4) shows that CPT C achieved only 33.33% reduction in elicitation due to the expert's indecisiveness in selecting compatible parental configurations.

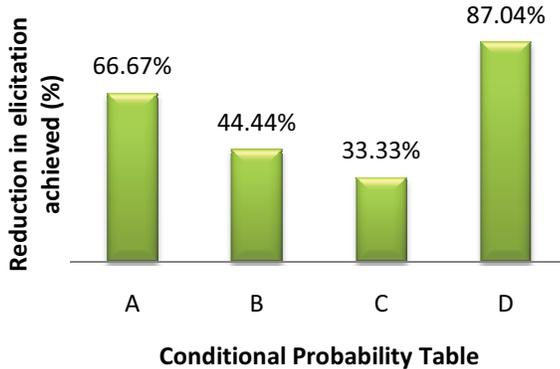


Figure 4. Elicitation reductions achieved in case study 1

With regards to accuracy, Figure 5 illustrates the distribution of absolute error values for each generated CPT. CPT C achieved nearly perfect accuracy (most values presenting an absolute error equal to zero), which is also due to the proposed extension of the algorithm, where higher accuracy is achieved due to further elicitation of probabilities. With the exception of CPT D, all median absolute error values were under 0.05, i.e. median accuracy of 95%, which is a low error rate in comparison to other studies assessing other CPT generation techniques [14, 17, 18].

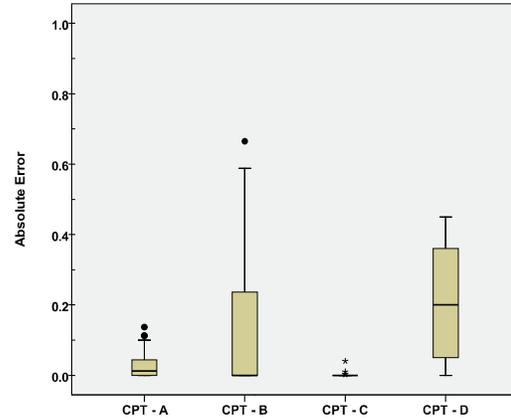


Figure 5. Distribution of absolute errors for case study 1

Despite the abovementioned results, no statistically significant differences in accuracy were observed between the two models being compared.

### Case Study 2

The benchmark model used in this study contained 16 nodes; seven were considered nontrivial in terms of probability parameters (highlighted in Figure 6) and were used as reference CPTs. Table III lists the seven reference CPT sizes.

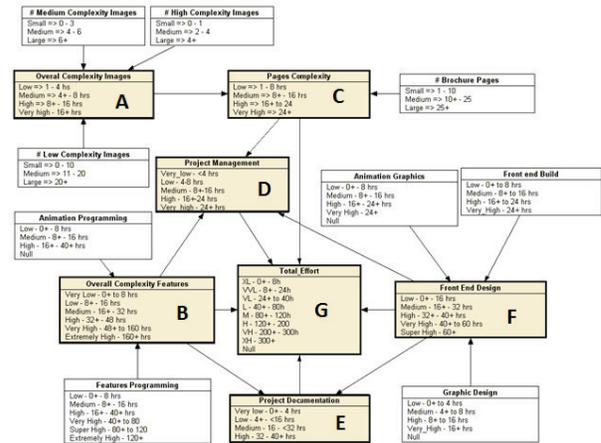


Figure 6. Benchmark Bayesian Network model for case study 2

TABLE III. THE SIZE OF CPTs IN CASE STUDY 2

CPT	Size	Number of Parents
A	108	3
B	144	2
C	48	2
D	600	3
E	120	2
F	500	3
G	21,600	5

During this case study we chose not to apply our proposed extension to the WSA in order to be able to assess the WSA as it was originally proposed; thus, whenever the domain expert was encountering difficulties in selecting a single compatible parental configuration from a set of valid configurations, they

were asked to arbitrarily select any configuration from the valid set.

A significant reduction in elicitation effort was achieved in case study 2, specifically in CPT G (the largest CPT used in both case studies), where only 72 probabilities were needed by the WSA algorithm, compared to 21,600 probabilities obtained using manual elicitation (see Figure 7). The reductions achieved in this case study clearly show the linear asymptotic growth that the WSA algorithm can achieve compared to the exponential growth of manual elicitation.

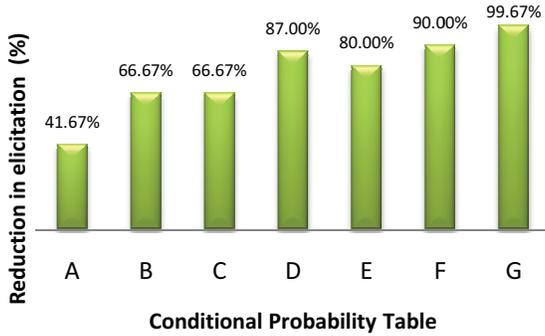


Figure 7. Elicitation reduction achieved in case study 2

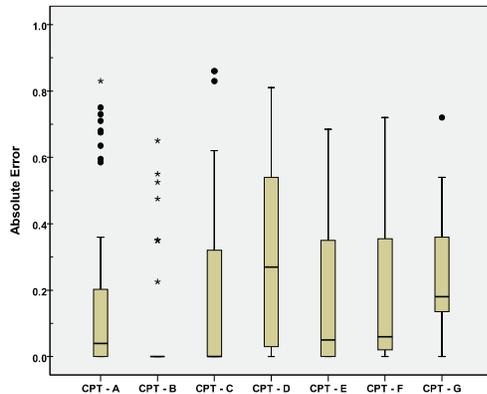


Figure 8. Distribution of absolute errors for case study 2

Regarding the accuracy, one can observe that with the exception of CPT D and G, the median absolute error was under 0.1, with CPT-B being the most accurate while CPT D the least (shown in Figure 8). Although the accuracy results in case study 2 were slightly lower than those for case study 1, it still presents relatively more accurate results when compared to other techniques (e.g. interpolation, and noisy gates).

These accuracy results also suggest that even though the expert was required to select exactly one compatible parental configuration for each parental state, the results achieved were not significantly more accurate than those for case study 1. This implies that our proposed extension to the WSA algorithm did not seem to significantly increase the accuracy whenever a domain expert could not select exactly one parental configuration. This was perhaps unexpected, as one would assume additional elicited information would result in much higher accuracy.

TABLE IV. CASE STUDY 2 WILCOXON SIGNED RANKED TEST RESULTS

CPT	Wilcoxon Z-value
A	-0.409
B	-0.687
C	-0.213
<b>D</b>	<b>-2.441</b>
E	-0.552
<b>F</b>	<b>-1.986</b>
<b>G</b>	<b>-22.404</b>

Table IV shows the seven Wilcoxon signed ranked test results. Three of which (emphasized in bold) fall outside the  $\pm 1.96$  range, implying statistically significant differences in the obtained estimates. Note that the three CPTs (D, F, and G) that have resulted in statistically significant differences all have sizes greater or equal to 500, whereas the remaining CPTs are much smaller in size (as described earlier in Table III). Thus, the results in this case study suggest that large CPT estimates tend to yield statistically significant differences compared to probability parameters obtained using manual elicitation.

In addition to the above, we also evaluated the accuracy of the weighted sum algorithm by comparing its prediction with that of the manually elicited BN model. We used a validation data set of 19 projects. Table V summarises the results of the comparison using the four commonly used error metrics: Mean Magnitude Relative Error (MMRE), Median Magnitude Relative Error (MdMRE), Mean Estimate Magnitude Relative Error (MEMRE), and Median Estimate Magnitude Relative Error (MdEMRE).

TABLE V. PREDICTION ACCURACY (IN TERMS OF RELATIVE ERROR)

Accuracy Measure	Manual Elicitation	The Weighted Sum Algorithm
MMRE	18.64%	17.66%
MdMRE	21.68%	20.54%
MEMRE	22.15%	21.64%
MdEMRE	17.92%	18.13%

The results in Table V show that the Weighted Sum Algorithm achieved very comparable prediction accuracy to that of manual elicitation, in fact, the results show a marginal improvement with the Weighted Sum Algorithm.

## VI. THREATS TO VALIDITY

Although we believe that the results of the two case studies described herein provide useful empirical evidence regarding the benefits of the WSA algorithm, there are threats to the validity of these results that must be taken into account:

The first threat relates to the number of experts who participated in each case study, given that a greater number of experts could perhaps increase the level of certainty relating to identifying compatible parental configuration for each parental state. Clearly further investigation is needed in order to confirm whether the accuracy changes whenever the number of experts increases.

Another threat relates to the size of the BN models employed in both case studies. Although both BN models did not seem to be small, further investigations involving very large BN models is needed to confirm/refute the patterns observed herein.

A related point is the applicability of the WSA to a particular domain. As stated previously, both cases studies

were based in the Web engineering domain. Certain intricacies specific to this domain might affect the performance of the WSA, for example, it might be more difficult to identify parental compatible configurations in other domains compared to the one used herein.

## VII. CONCLUSIONS

CPT sizes grow exponentially relative to the number of parental states. Therefore, a seemingly undersized Bayesian Network model can still contain very large CPTs. For that reason, abating this exponential growth is an imperative requisite for eliciting larger and more comprehensive BNs.

The WSA relies on the notion of compatible parental configurations to elicit probabilities for scenarios that are easier to recall by the expert, and hence likely to be easier to elicit and be more reliable. The algorithm utilizes a very simple calculation based on elicited parental weights, and a weighted sum. However, the original algorithm proposed by Das does not describe how to deal with situations where the expert cannot select a single compatible parental configuration for a given parental state. Therefore, we proposed an extension for the algorithm that remedies this situation by averaging the probabilities of valid compatible parental configurations that expert might select.

The aims of our work was therefore to empirically assess the WSA in terms of elicitation reduction and accuracy. Another secondary goal was to see how effective our proposed extension to algorithm was.

We assessed the WSA in two case studies. The first case study consisted of four nontrivial CPTs, two CPTs to which our proposed extension to the original WSA algorithm was applied to. The outcome of the first case study showed that overall the WSA algorithm achieved a significant reduction in the effort eliciting probabilities, and achieved very high accuracy, with a median absolute error value less than 0.05 for three of the four CPTs.

The second case study used a larger benchmark BN model with seven nontrivial CPTs; however, in this case study, the proposed extension to the algorithm was not applied, and instead, the expert was asked to arbitrarily select a compatible parental configuration in the event that there is more than one valid configuration compatible with a given prenatal state.

The results of the second case study showed slightly less accurate results, however, still very much comparable to that of the first case study, perhaps suggesting that our proposed extension might yield slightly better accuracy, but not necessarily worth the tradeoff of eliciting more probabilities. The elicitation reduction results in the second case study were even more significant than the first. With the largest CPT of (21,600 parameters) being reduced by 99.67% to only 72 parameters, which illustrates the linear growth of the input probabilities required by the algorithm, compared to the exponential growth of manual elicitation. The Wilcoxon signed ranked test results also suggest that the algorithm only yields statistically significant results when the CPTs are large (on the scale of 500 or more parameters).

In conclusion, the WSA has demonstrated to significantly alleviate the CPT elicitation burden in both case studies, whilst yielding high accuracy levels. The algorithm's simplicity and lack of constraints it assumes makes it an attractive technique

for solving the CPT elicitation burden, and should be considered by BN practitioners.

## ACKNOWLEDGEMENTS

We like to acknowledge all the participating companies and experts in this research for providing their time and expertise. This work was sponsored by the Royal Society of New Zealand (Marsden research grant 06-UOA-201).

## REFERENCES

- [1] B. Das, "Generating Conditional Probabilities for Bayesian Networks: Easing the Knowledge Acquisition Problem," *CoRR*, vol. cs.AI/0411034, 2004.
- [2] T.A. Stephenson, *An Introduction to Bayesian Network Theory and Usage*, bIDIAPRR, 2000.
- [3] U.B. Kjærulff and A.L. Madsen, "Probabilistic Networks—An Introduction to Bayesian Networks and Influence Diagrams," *Aalborg University*, 2005.
- [4] R.E. Neapolitan, "Learning Bayesian networks," *Proc. Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM New York, NY, USA, 2007.
- [5] E. Mendes, et al., "Building an Expert-based Web Effort Estimation Model using Bayesian Networks," *13th International Conference on Evaluation & Assessment in Software Engineering*, 2009.
- [6] X.-x. Hu, et al., "Using Expert's Knowledge to Build Bayesian Networks," *Proceedings of the 2007 International Conference on Computational Intelligence and Security Workshops*, no. 1333212, 2007, pp. 220-223; DOI <http://dx.doi.org/10.1109/CIS.Workshops.2007.63>.
- [7] L.C. van der Gaag, et al., "How to elicit many probabilities," *Book How to elicit many probabilities*, Series How to elicit many probabilities, ed., Editor ed.^eds., Morgan Kaufmann, 1999, pp. 647--654.
- [8] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*, Morgan Kaufmann, 1988.
- [9] D. Heckerman and J.S. Breese, "Causal independence for probability assessment and inference using Bayesian networks," *IEEE Transactions on Systems, Man and Cybernetics, Part A*, vol. 26, no. 6, 1996, pp. 826-831.
- [10] A. Tversky and D. Kahneman, "Judgment under uncertainty: Heuristics and biases," *Judgment under Uncertainty: Heuristics and Biases*, 1982, pp. 3-20.
- [11] A. Tversky and D. Kahneman, "Availability: A heuristic for judging frequency and probability," *Judgment under Uncertainty: Heuristics and Biases*, 1982, pp. 163-178.
- [12] S. Baker, "Towards the Construction of Large Bayesian Networks for Web Cost Estimation," Department of Computer Science, University of Auckland, Auckland, 2009.
- [13] S. Baker and M. Emilia, "Assessing the Weighted Sum Algorithm for Automatic Generation of Probabilities in Bayesian Networks," *Proc. International Conference on Automation and Information (ICIA2010)*, 2010.
- [14] A. Zagorecki and M. Druzdzal, "An empirical study of probability elicitation under Noisy-OR assumption," *Proc. Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference (FLAIRS. 2004)*, pp. 880-885.
- [15] F.V. Jensen, *An Introduction to Bayesian Networks*, UCL Press, 1996.
- [16] O. Woodberry, et al., "Parameterising Bayesian Networks," *Proc. Australian Conference on Artificial Intelligence*, 2004, pp. 1101-1107.
- [17] A. Onisko, et al., "An experimental comparison of methods for handling incomplete data in learning parameters of Bayesian networks," *Physica Verlag*, 2002, pp. 351.
- [18] T. Zhong and M. Brenda, "Developing Complete Conditional Probability Tables from Fractional Data for Bayesian Belief Networks," *Journal of Computing in Civil Engineering*, vol. 21, no. 4, 2007, pp. 265-276.

# An Ontology-based Mapping Repository for Meta-querier Customization

Xiao Li, Randy Chow  
Department of CISE, University of Florida  
Gainesville, FL, 32611, USA  
{x11, chow}@cise.uf.edu

## Abstract

*There exist an increasing number of (semi-)structured information sources over the Web. To increase the access efficiency of the heterogeneous information sources, customized meta-queriers are built by integrating the user-specified Web query forms. Each meta-querier requires mappings for resolving semantic interoperability among data sources. A potentially large set of mappings need to be stored and discovered. Thus, this paper proposes an ontology-based mapping repository, called M-Ontology, for efficient storage and discovery of these mappings. We develop a novel semantics-based approach to model the mappings. Based on this model, a well-defined mapping ontology is generated through incremental insertion of unstructured query-form mappings. We also present a reuse-oriented mapping discovery algorithm for many-to-many complex mappings using M-Ontology. The discovered mappings are recycled to further enrich the mapping ontology, which in turn enhance the discovery of new mappings. Finally, the experiments show promising results in the feasibility and effectiveness of our algorithms.*

## 1. Introduction

WWW has been experiencing a tremendous growth of (semi-)structured information, in particular, the databases behind deep web [17, 6]. Data integration techniques are often applied over these (semi-)structured sources to increase the efficiency of information access. A *meta-querier* achieves higher efficiency by providing users a uniform query form (called a *global form*) to a set of integrated data sources. User queries through such a global form are translated to respective local queries and then the combined local query results are returned. The query translation is based on the *mappings* between the global query form and the query forms (called *local forms*) of the underlying data sources.

Much effort has been made on the construction of a single domain-specific meta-querier (e.g., WISE-Integrator [14] and Meta-Queriers [7]). However, different users have different selection criteria for data sources, e.g., functionalities, data qualities, site credibility. It is impossible to build a one-size-fits-all meta-querier to satisfy the diverse user needs and preferences [27]. *Customizability* is an essential requirement

for meta-querier construction systems. It is highly desirable to allow users to build their own meta-queriers by selecting their preferred data sources. However, this implies a potentially large number of customized meta-queriers to accommodate various user needs. The rising scalability of meta-queriers renders the existing approaches impractical.

To construct and maintain a large number of meta-queriers, the arguably most critical problems are *storage* and *discovery* of mappings, which are the cornerstones of a customized meta-querier. To address such problems, this paper proposes an ontology-based mapping repository, called *M-Ontology*. The existing research often regards mapping repositories as simple storage containers of mappings with their related information. However, in the context of customized meta-queriers, we believe that mapping repositories need to have richer semantics and play a more important role because of the necessity of building and maintaining a potentially large scale of meta-queriers.

Based on this observation, M-Ontology views a mapping as an instance of a relation connecting two concepts. The concepts can be automatically extracted from the query forms of data sources that are connected by mappings. A relation in M-Ontology is a higher-level abstraction of mappings based on their semantics. Such a semantics-based mapping modeling is purposely designed for better understanding by humans. M-Ontology functions like a domain-specific knowledge base for mapping sharing and reuse to facilitate ease of machine automation. The implementation of such a mapping repository has three major design considerations:

1) *Sharing of mappings*: The large scale of customized meta-queriers often indicates that repetitive mappings need to be stored and managed. Separate storage might cause a high degree of data redundancy and potential update anomalies. Thus, M-Ontology is not for a single meta-querier, but shared by a group of meta-queriers in the same application domain.

2) *Understanding of mappings*: The mapping repository is intended for use not only by machines but also by humans. We develop a novel semantics-based approach to model the mappings. Based on this model, a well-defined mapping ontology can be constructed by our proposed mapping insertion algorithm.

3) *Reuse of mappings*: Discovering the mappings between

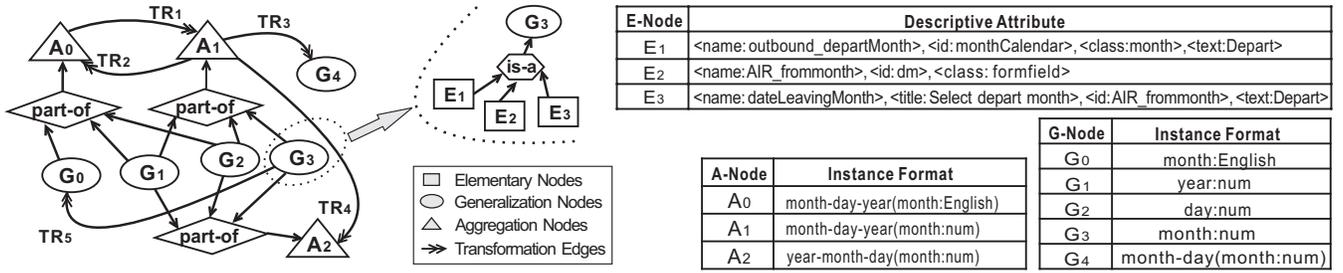


Figure 1. A fragment of M-Ontology for the airline-ticket-booking domain.

global forms and local forms is a critical issue in the meta-querier construction. Although much effort has been made in mapping discovery, finding complex mappings with expressions is still an open problem. We believe that reusing the existing mappings is one of the best (or maybe the only) ways to find the new mappings, especially those with complex expressions. We propose a semi-automatic approach to discover many-to-many mappings through M-Ontology.

The rest of this paper is organized as follows. Section 2 presents the modeling and construction of the M-Ontology. Section 3 describes the algorithm of mapping discovery through M-Ontology. Section 4 presents the implementation and the experimental analysis. Finally, we discuss the related work and future work in Section 5 and 6.

## 2 M-Ontology

### 2.1 Modeling

In meta-queriers, global and local query forms can be regarded as a set of query conditions [25, 15] (referred to here as *schema elements*). Each schema element consists of a HTML control (e.g., a checkbox), its associated instances (i.e., possible user inputs) and descriptive attributes (called *D-Attributes*, e.g., name, id and class).

Generally, an element-level mapping can be defined as *Mapping* ( $List_{E1}, List_{E2}, Exp, ConV$ ), where  $List_{E1}$  and  $List_{E2}$  are two ordered lists of schema elements, whose semantics are relevant to each other. The element number of a list can be one or greater than one, and thus mapping cardinality might be  $1 : 1$ ,  $1 : n$  or  $n : m$  ( $n > 1$  and  $m > 1$ ).  $Exp$  denotes a high-level declarative expression that specifies the transformation rules from  $List_{E1}$  to  $List_{E2}$ . Expressions can be list-oriented functions (e.g. equivalence, concatenation, mathematic expressions) or other more complex statements (e.g., if-else, while-loop). In addition, the format of  $Exp$  should be both human-understandable (i.e., can be easily modified by normal users), and machine-processable (i.e., can be automatically transformed to executable rules).  $ConV$  is a confidence value (ranging between 0 and 1), indicating how confident the mapping holds.

With this mapping representation, we can present a graph model for the proposed mapping ontology. M-Ontology is not language specific. Like most classical ontology models [23, 13], M-Ontology can be represented by a directed graph, as illustrated in Fig.1, where a node denotes a concept with a set of associated instances; an edge indicates a relationship between two concepts. The relation types are restricted

to data transformation. Three types of concept nodes and a major type of relation edge are explained as follows.

**E-Nodes:** An Elementary Node (called an *E-Node*) represents the concept of a schema element. It is the most fundamental concept unit for constructing high-level concepts. E-Nodes aim at eliminating *language heterogeneity*. Language heterogeneity refers to the difference in the languages used to represent query forms. Each schema element is encapsulated into an E-Node without semantic loss. The semantics of an E-Node is reflected by the *D-Attributes* of the corresponding schema element (e.g., name, id, domain). Each E-Node also has *Instances* and *I-Constraints*, which are the known data instances and the instance constraints, such as the instance type and domain. For example, the E-Node  $E_1$  in Fig.1 corresponds to a schema element whose D-Attributes are  $\langle \text{name:outbound\_departMonth} \rangle$ ,  $\langle \text{id:monthCalendar} \rangle$ ,  $\langle \text{class:month} \rangle$  and  $\langle \text{text:Depart} \rangle$ , and instances are represented in a number from '1' to '12'. It can be correctly inferred from its D-Attributes and instances that this element indicates a concept about departure months.

**G-Nodes:** Generalization Nodes (called G-Nodes) are the generalization of E-Nodes that share the same semantics and instance formats but their syntactic representations are different. The semantics of a G-Node is equivalent to that of its inclusive E-Nodes. The D-Attributes and D-Labels of a G-Node  $gn$  are two sets of word tuples  $Set_{gn}^{DA}$  and  $Set_{gn}^{DL}$ . Each word tuple  $(w_i, wf_i)$  consists of a distinct word  $w_i$  and the corresponding appearance frequencies  $wf_i$  in  $gn$ .  $Set_{gn}^{DA}$  is generated from the normalized D-Attributes of the included E-Nodes whose insertion is already verified by humans. The D-Labels can be automatically derived from frequent appearance of D-Attributes or manually input by humans. The detailed algorithms for finding the D-Attributes and D-Labels of a G-Node are discussed in Section 2.2. In addition, the Instances and I-Constraints of a G-Node are directly obtained from the inclusive E-Nodes. For example,  $G_3$  has the same instance set and I-Constraints as  $E_1, E_2$  and  $E_3$ .

The purpose of G-Nodes is for *syntactic heterogeneity*. The E-Nodes in  $Set_{E-Node}$  share the same semantics and instance formats but their syntactic representations are different. That is, the instances of these E-Nodes share the same format with the identical semantics if they represent the same object, but the D-Attributes of these E-Nodes could be totally different. To take an example, in Fig.1, the concept of G-Node  $G_3$  is generalized from three verified E-

Nodes. Although these E-Nodes have equivalent semantics and instances format, their D-Attributes are different in the attributes' numbers, types and values. It is not necessary to impose any syntactic transformation when these E-Nodes exchange their instances.

**A-Nodes:** An Aggregation Node (called an *A-Node*) is generated by aggregating an ordered list of G-Nodes. A-Nodes are introduced for representing many-to-many mappings. Its Instances and D-Labels can be fetched from its inclusive nodes, and then combined in the same order. For example, two A-Nodes  $A_1$  and  $A_2$  in Fig.1 represent the concept “departure dates” by aggregating the same three G-Nodes  $G_1$ ,  $G_2$  and  $G_3$ . These G-Nodes respectively represent a calendar year, day and month by numbers. Although their semantics are equivalent, their instance formats are different due to different element orders.  $A_1$  uses middle endian forms “month-day-year” (i.e., starting with the month).  $A_2$  uses big endian forms “year-month-day” (i.e., starting with the year). Note that aggregation cannot be directly applied on E-Nodes in M-Ontology. In other words, A-Nodes only connect with G-Nodes. This requirement enforces E-Nodes to be encapsulated into G-Nodes. It indicates more concepts can be generated by clustering schema elements.

**T-Edges:** A Transformation Edge (called *T-Edge*) connecting two nodes represents a transformation relation. Combining with the G-Nodes and A-Nodes, T-Edges aims to address three different types of heterogeneity between two semantic-overlap concepts. They are *instance heterogeneity*, *structural heterogeneity* and *semantic heterogeneity*: 1) Instance heterogeneity refers to the variety in instance formats for the same entity. For example, both  $G_0$  and  $G_3$  represent the departure months, but they respectively use different formats, i.e., numbers and character strings. Thus, a T-Edge  $TR_5$  might be created for transforming instances from  $G_3$  to  $G_0$ , if necessary. 2) Structural heterogeneity often occurs when the aggregation orders of concepts are different. For example,  $A_1$  and  $A_2$  represent a date in a different order, i.e., “month-day-year” and “year-month-day”. A T-Edge  $TR_4$  is used to link  $A_1$  and  $A_2$ . 3) Semantic heterogeneity represents the differences in the concept coverage and granularity. For instance,  $TR_3$  is used to connect two concepts  $A_1$  and  $G_4$  with different coverage. Different from  $A_1$ ,  $G_4$  only represents the departure date without “year”.

**Metadata:** M-Ontology stores not only mappings but also their related context information, i.e., the metadata of mappings. The metadata is mainly used to enhance system effectiveness and robustness. The metadata includes the who, what, where, when and how aspects associated with the whole lifecycle of mappings, including their creation, verification, modification and deletion.

The proposed mapping modeling organizes mappings based on their semantics in a straightforward approach. It supports five types of heterogeneity: language, syntactic, instance, structural and semantic heterogeneity. Such a semantic modeling makes sense of unorganized mapping information so as to enhance the understanding by humans, espe-

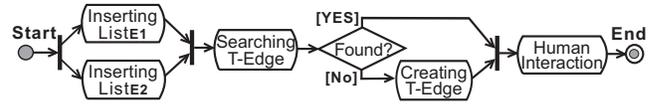


Figure 2. The flowchart of ontology construction.

cially when the scale of mappings is potentially large.

## 2.2 Incremental Construction

M-Ontology is an ontology-based domain-specific mapping repository. Its construction is through incremental insertions of element-level mappings. The incremental approach is very critical for building a long running and shared mapping repository. Since both users and data sources are autonomous units, source selection and local query forms are dynamic. New mappings continue to be inserted as new data sources are inserted or old ones evolve. M-Ontology requires periodic update. It is more effective to update it incrementally rather than re-learn the ontology from scratch. M-Ontology splits the semi-automatic mapping insertion into three sequential phases as illustrated in Fig.2: 1) inserting the element lists,  $List_{E1}$  and  $List_{E2}$ ; 2) inserting the expression  $Exp$ ; 3) validating the insertion by humans.

- **Element Insertion** (phase 1) is to identify concepts for a mapping ontology  $mo$  through individually inserting each element of an element list  $list_E$ . Besides updating the ontology  $mo$ , this phase also returns an *abstract node*  $cn$  corresponding to  $list_E$  for the next phase (Note that for the simplicity of description, an *abstract node* is introduced hereafter to represent a node that might be a G-Node/A-Node.)

In this phase, each schema element  $e$  in  $list_E$  is checked if it already exists in the ontology  $mo$ . If true, its G-Node is (or a part of) the target abstract node. If false, it is encapsulated into a new E-Node  $en$ . Then, the function  $SearchGNode$  seeks the most suitable G-Node  $gn$  in  $mo$  for inserting  $en$ . If not found,  $en$  is inserted into a new G-Node; otherwise,  $en$  is directly inserted into the found  $gn$ . If  $list_E$  only contains a single element,  $gn$  is the abstract node  $cn$  that will be returned; or else each of the found G-Nodes  $gn$  is inserted into a new A-Node  $an$  in the same order with the original one of schema elements in  $list_E$ . Finally, the function  $SearchANode$  goes over  $mo$  to find an A-Node  $anInMO$  with the same contents like  $an$ . If such an A-Node exists,  $anInMO$  is the node which needs to be returned; or else  $an$  will be returned as an abstract node. The core of this phase is two search functions:

- SearchANode** can be implemented by directly evaluating the equivalence between  $an$  and the A-Nodes in  $mo$ . Two A-Nodes are equivalent only when they include the same G-Nodes in the same order.

- SearchGNode** is implemented by comparing the semantic similarities of all the G-Nodes in  $mo$  with the E-Node  $en$ . The target G-Node  $gn$  is the one with the highest similarity value  $sim_{gn-en}$  that must exceed a given threshold. If no G-Node is eligible, a null value is returned. The similarity  $sim_{gn-en}$  between  $gn$  and  $en$  is determined by two steps:

- Step 1: Preprocessing.** The D-Attributes and Instances of  $en$  are collected and then normalized by NLP (natural lan-

guage processing) techniques: tokenization, stop-word removal and stemming. The resulting texts are considered as two bag of words, that is, two unordered sets of words,  $Set_{en}^{DA}$  and  $Set_{en}^{INS}$ .

Step 2: *Calculating*. Resulting from the previous human-verified insertion of E-Nodes, the G-Node  $gn$  has an instance set  $Set_{gn}^{INS}$  and two word sets  $Set_{gn}^{DL}$  and  $Set_{gn}^{DA}$ , respectively, corresponding to its D-Labels and D-Attributes. If  $gn$  and  $en$  are constraint-compatible, their semantic similarity is calculated by,

$$sim_{gn-en} = w_1 \times sim_1(Set_{gn}^{DL}, Set_{en}^{DA}) + w_2 \times sim_2(Set_{gn}^{DA}, Set_{en}^{DA}) + w_3 \times sim_3(Set_{gn}^{INS}, Set_{en}^{INS})$$

where  $w_1, w_2$  and  $w_3$  are three weights in  $[0,1]$  such that  $\sum_{i=1}^3 w_i = 1$ . The similarity functions  $sim_1, sim_2$  and  $sim_3$  are implemented by a hybrid matcher that combines several linguistics-based matchers, such as WordNet-synonym distances, 3-gram distances and Jaccard distances. Many algorithms for schema/ontology matching have been proposed (see the surveys[22, 11]). Most of them can be easily employed in this phase. As this is not the target of this paper, the details are not explained in this paper.

• **Expression Insertion** (phase 2): The target of this phase is to identify and create T-Edges. The inputs of phase 2 include an expression  $Exp$ , and two abstract nodes  $cn_1$  and  $cn_2$  that respectively correspond to  $List_{E1}$  and  $List_{E2}$ . Phase 2 checks if there exists a T-Edge with the same expression as  $Exp$  from  $cn_1$  to  $cn_2$ . If it does not exist, a T-Edge containing  $Exp$  is created; otherwise, this phase is finished.

• **Human Interaction** (phase 3): Humans are responsible for the validation and correction of nodes and edges. For alleviating possible redundancy and error propagation, only the nodes and edges that have been validated by humans can affect the future mapping insertion. *VerifyENode* and *VerifyT-Edge* are two of the main verification functions.

i) **VerifyENode**: After an E-Node  $en$  is verified, its two normalized bags of words,  $Set_{en}^{DA}$  and  $Set_{en}^{INS}$ , start to affect the corresponding G-Node  $gn$ . The D-Attributes  $Set_{gn}^{DA}$  and Instances  $Set_{gn}^{INS}$  of  $gn$  need to be updated to combine the words occurring in the E-Node. The machine-found D-Labels  $Set_{gn}^{DL}$  might also evolve by selecting the D-Attributes with the top-k frequencies (e.g.,  $k=5$ ). In case of updates on the G-Node, the relative A-Node is also updated if necessary.

ii) **VerifyTEdge**: The verification of a new T-Edge affects not only its original associated mapping, but also all the indirectly connected E-Nodes inserted in the past and future. Such an activity indicates  $n \times m$  mappings are validated, where  $n$  and  $m$  are respectively the value of possible E-Node combination of the connected abstract nodes.

Following the above mapping insertion algorithm, M-Ontology is constructed and enriched as more and more mappings are inserted. The core of this algorithm is a representative-based clustering algorithm that is straightforward to human users. Users can easily improve the performance by manually changing the representatives, e.g., adding or removing several D-Labels and D-Attributes.

### 3. Mapping Discovery

Mapping discovery is a critical operation for finding the mappings between two query forms. Our approach is to match two forms through an intermediary mapping ontology. Its core is the reuse of previous mappings. The main idea is based on a feature of M-Ontology: when an E-Node is inserted into an existing G-Node, all the mappings associated with this G-Node will be automatically assigned to this E-Node. That is, all the E-Nodes in the same G-Nodes share their mapping information since they have the identical semantics in the same formats. This feature greatly enhances the reuse of existing mappings, compared with the direct reuse [22, 24, 10, 8]. The matching between  $QForm_1$  and  $QForm_2$  consists of three sequential phases:

• **Searching Abstract Nodes** (phase 1): This phase is to find two sets of abstract nodes  $ASet_1$  and  $ASet_2$  that respectively correspond to  $QForm_1$  and  $QForm_2$ .

Step 1: *G-Node searching*. For each element of the query forms, the aforementioned function *SearchGNode* can be employed to find the most suitable G-Node from M-Ontology. Then, all the suitable G-Nodes are inserted into the corresponding abstract-node sets.

Step 2: *A-Node searching*. An A-Node is selected into the abstract-node sets only when all its inclusive G-Nodes are already contained in the sets.

• **Discovering Mappings** (phase 2): This phase is to discover query form mappings from  $QForm_1$  to  $QForm_2$  and vice versa, represented as  $MP_1^2$  and  $MP_2^1$  respectively.

Step 1: *Searching the reachable abstract nodes*. An abstract node  $cn$  is considered *reachable* from  $ASet_1$  only if it is able to find an edge to  $cn$  from any node in  $ASet_1$ . Thus, it is simple to obtain two sets of reachable nodes  $RSet_1$  and  $RSet_2$  respectively from  $ASet_1$  and  $ASet_2$  in a mapping ontology.

Step 2: *Finding the overlap*. The overlaps  $RSet_1 \cap ASet_2$  and  $RSet_2 \cap ASet_1$  separately indicate the mappings  $MP_1^2$  and  $MP_2^1$  whose expressions are in the connected T-Edges. The overlap  $ASet_1 \cap ASet_2$  denotes the semantics-equivalence mappings between two query forms. The demanded mappings can be directly derived from the above three overlaps.

• **Validating & Correcting Mappings** (phase 3): The final phase is to verify and correct the machine-discovered mappings by human beings. These verified mappings are recycled to incrementally construct M-Ontology by the proposed construction algorithm in Section 2.2.

### 4 Implementation and Experiments

M-Ontology is built on top of an open-source system “Alignment Server” [1]. This system provides some basic services, like mapping storage and ontology matching. To evaluate the feasibility and effectiveness of M-Ontology, we conduct experiments to simulate ontology construction and the subsequent mapping discovery. Mapping diversity and repetition depend on user selection of data sources, the composition of local query forms and the degree and frequency

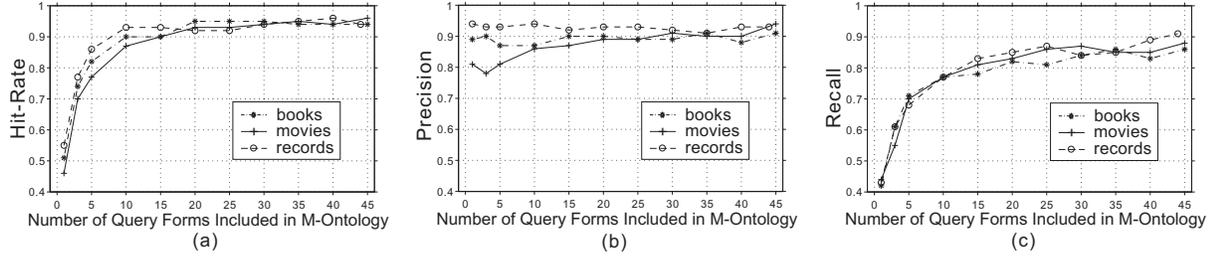


Figure 3. A fragment of M-Ontology for the airline-ticket-booking domain.

of their changes. It is hard to simulate the complete processes and performances of real applications. Thus, the aims of our experiments are to assess: i) *Feasibility*: how high the percentage of schema elements in real-world query forms have corresponding concepts stored in M-Ontology during the incremental construction process? ii) *Effectiveness*: how well do our algorithms perform correctly in searching out suitable G-Nodes for a query form (i.e., the function SearchGNode) after the insertion results of a set of query forms have been verified (i.e., the function VerifyENode)?

**Data Sets and Performance Metrics.** To examine the performance of M-Ontology in real-world data integration problems, we collect 141 query form URLs from the UIUC web integration repository[2] after removing the inactive websites. All these query-form URLs are from three domains: Books(47), Movies(49) and Music Records(45), where the value in parentheses denotes the number of URLs in the domain. The information of query forms is manually extracted from the HTML source codes to generate the corresponding Web query forms. These query forms are represented in OWL to accommodate the mapping format of Alignment Server. We use three metrics to evaluate the performance of the algorithms, *Hit-rate*, *Precision* and *Recall*. *Hit-rate* is designed to measure the feasibility of M-Ontology. *Precision* and *Recall* are widely used in evaluating the effectiveness of information retrieval systems. *Precision* measures the degree of correctness of G-Node searching. *Recall* measures the degree of completeness of G-Node searching. In the setting of M-Ontology, these metrics can be defined for a specific value  $m$  as: i)  $Hit-rate = MO_e / NUM_e$ , ii)  $Precision = Crt_e / Total_e$ , iii)  $Recall = Crt_e / MO_e$  where,  $NUM_e$  is the total element number;  $MO_e$  is the number of elements that can be correctly identified with the G-Nodes;  $Crt_e$  is the number of elements that are matched to the correct G-Nodes;  $Total_e$  is the number of elements that are matched to a specific G-Node. Note all the above element numbers are from a query form to be inserted/matched.

**Experiment Scenarios and Results.** Assume that  $m$  Web query forms from the same domain are already inserted into M-Ontology based on our proposed algorithms. The insertions of these forms have been already corrected and verified by human beings so that the attributes (e.g.,  $Set_{gn}^{DA}$ ) of the corresponding G-Nodes are updated to combine the verified schema elements. We attempt to find the correct G-Nodes for a single query form  $X$  that is not in M-Ontology. Form  $X$  is randomly selected from the forms that are not stored in

M-Ontology. That is,  $X$  is different from any form in M-Ontology for the purpose of removing the influence of possible repetitiveness. As shown in Fig.3, the same trends can be observed in all three domains. Each experimental result shows eleven values of  $m$  ranging from 1 to 45. The performance measures for each  $m$  are calculated by the average of 200 samples, each of which is automatically generated from the query-form sets in that domain. Each sample includes  $m$  different forms existed in M-Ontology and the form  $X$ . All these query forms are randomly selected.

In Fig.3(a), when  $m$  reaches 15, at least 90% elements in  $X$  can find out the correct G-Nodes so that these elements can directly reuse the associated mappings. The importance of existing query forms on Hit-rates are clearly indicated, as a sharp increase of Hit-rate can be seen when  $m$  is smaller than 15. The remaining curve seems to indicate that the additional forms ( $>15$ ) almost do not affect the Hit-rates since 90% concepts are already collected from the first 15 forms. However, this is not always true since more mapping edges might be added with more forms inserted. The experimental results also match the conclusion by a related study [6] showing that the aggregated vocabularies used to describe schema elements are “clustering in localities and converging in size”.

In Fig.3(b), the values of Precision stay around 90% after the initial learning process ( $m > 10$ ). Although more G-Nodes are available for classification with  $m$  increasing (Fig.3(a)), our proposed searching algorithm also can classify most schema elements into the correct concepts.

In Fig.3(c), a sharp increase of Recall can be seen before  $m$  reaches 10, followed by a steady and slow improvement with  $m$  increases. This phenomenon indicates that D-Labels of G-Nodes cannot be correctly identified when the inclusive schema elements are very few. When  $m$  increases, the contents of D-Labels become steady but Instances and D-Attributes can accumulate more useful information from the newly verified query form insertion. Thus, Recall increases slowly and steadily after  $m$  is larger than 10.

From the above experiments, we see that schema elements in the same domain can be clustered to generate a relatively “small” mapping ontology. That indicates it is feasible to construct a mapping ontology as a knowledge base shared by meta-queriers in the same domain. The performance results in terms of Precision and Recall demonstrate the effectiveness of the algorithms even without considering human intervention. Furthermore, as the number of data sources on

the Web has been steadily increasing [17, 6], better performance of M-Ontology can be expected when more mappings are included.

## 5. Related Work

Most mapping repositories [19, 21, 26] simply employ a large table to store schema-level or element-level mappings. Its columns represent different properties of mappings, such as schema/element names, mapping expressions, and similarity values. Others use graph models in which nodes denote schemas [4, 9] or elements [3, 20] and edges represent mappings. In a mapping graph, a new mapping edge between two nodes could be discovered by combining all the edges in a mapping path between these two nodes [18, 12, 5]. However, all these mapping repositories model the mappings directly without organizing them based on their semantics. In contrast, M-Ontology attempts to identify and store the hidden concepts from instances in a straightforward approach. It aims at better mapping management and reutilization not only by machines but also by users.

Currently, the techniques of schema/ontology matching [22, 11] can hardly find mappings with complex expressions, which are very common in meta-queriers. Our approach attempts to overcome this bottleneck by reusing the existing expressions in mapping ontologies. Different from the previous work [22, 24, 10, 8], our strategy is not to directly reuse individual mappings, but to reuse their previous concept classification. Due to space limit, the detailed related work is not elaborated here. Interested readers are referred to our full version [16].

## 6. Conclusion

In the Web era, numerous customized meta-queriers are needed in meeting various user requirements. This paper proposes an ontology-based mapping repository to efficiently store and discover mappings. Although it is primarily designed for meta-querier customization, it should be useful for applications where numerous similar mappings exist, e.g., the data mediation problem in the composition of semantic web services.

We have implemented the basic M-Ontology. Future work in this direction includes: applying the proposed mapping ontology to the generation of global query forms (a.k.a., schema integration) in the context of dynamic and customized meta-queriers, and extending the domain-specific M-Ontology to support cross-domain data integration.

## References

- [1] Alignment api and alignment server. <http://alignapi.gforge.inria.fr/>.
- [2] The UIUC Web integration repository. <http://metaquerier.cs.uiuc.edu/repository>.
- [3] S. Bergamaschi, S. Castano, and M. Vincini. Semantic integration of semistructured and structured data sources. *SIGMOD Record*, 28(1):54–59, 1999.
- [4] P. A. Bernstein. Applying model management to classical meta data problems. In *CIDR*, 2003.
- [5] P. A. Bernstein, T. J. Green, S. Melnik, and A. Nash. Implementing mapping composition. *VLDB J.*, 17(2):333–353, 2008.
- [6] K. C.-C. Chang, B. He, C. Li, M. Patel, and Z. Zhang. Structured databases on the Web: Observations and implications. *SIGMOD Record*, 33(3):61–70, 2004.
- [7] K. C.-C. Chang, B. He, and Z. Zhang. Toward large scale integration: Building a MetaQuerier over databases on the Web. In *CIDR*, pages 44–55, 2005.
- [8] R. Dhamankar, Y. Lee, A. Doan, A. Halevy, and P. Domingos. iMAP: discovering complex semantic matches between database schemas. In *SIGMOD Conference*, pages 383–394, 2004.
- [9] H. H. Do. *Schema Matching and Mapping-based Data Integration*. PhD thesis, University of Leipzig, <http://lips.informatik.uni-leipzig.de/pub/2006-4>, 2006.
- [10] C. Drumm, M. Schmitt, H. H. Do, and E. Rahm. Quickmig: automatic schema matching for data migration projects. In *CIKM*, pages 107–116, 2007.
- [11] J. Euzenat and P. Shvaiko. *Ontology Matching*. Springer-Verlag New York, Inc., 2007.
- [12] R. Fagin, P. G. Kolaitis, L. Popa, and W. C. Tan. Composing schema mappings: Second-order dependencies to the rescue. *ACM Trans. Database Syst.*, 30(4):994–1055, 2005.
- [13] A. Gómez-Pérez, O. Corcho, and M. Fernández-López. *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. Springer, 2004.
- [14] H. He, W. Meng, C. T. Yu, and Z. Wu. Automatic integration of Web search interfaces with WISE-Integrator. *VLDB J.*, 13(3):256–273, 2004.
- [15] J. Hong, Z. He, and D. A. Bell. Extracting Web query interfaces based on form structures and semantic similarity. In *ICDE*, pages 1259–1262, 2009.
- [16] X. Li and R. Chow. An ontology-based mapping repository for dynamic and customized data integration. Technical Report REP-2009-483., Dept. of CISE at University of Florida, 2009. <http://www.cise.ufl.edu/~chow/techreport483.pdf>.
- [17] J. Madhavan, S. Cohen, X. L. Dong, A. Y. Halevy, S. R. Jeffrey, D. Ko, and C. Yu. Web-scale data integration: You can only afford to pay as you go. In *CIDR*, pages 342–350, 2007.
- [18] J. Madhavan and A. Y. Halevy. Composing mappings among data sources. In *VLDB*, pages 572–583, 2003.
- [19] S. Melnik, E. Rahm, and P. A. Bernstein. Rondo: A programming platform for generic model management. In *SIGMOD Conference*, pages 193–204, 2003.
- [20] P. Mitra, G. Wiederhold, and M. L. Kersten. A graph-oriented model for articulation of ontology interdependencies. In *EDBT*, pages 86–100, 2000.
- [21] N. F. Noy, N. Griffith, and M. A. Musen. Collecting community-based mappings in an ontology repository. In *ISWC*, pages 371–386, 2008.
- [22] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB J.*, 10(4):334–350, 2001.
- [23] S. Staab and R. Studer, editors. *Handbook on Ontologies*. International Handbooks on Information Systems. Springer, 2004.
- [24] W. Wu, C. Yu, A. Doan, and W. Meng. An interactive clustering-based approach to integrating source query interfaces on the deep Web. In *SIGMOD Conference*, pages 95–106, 2004.
- [25] Z. Zhang, B. He, and K. C.-C. Chang. Understanding Web query interfaces: Best-effort parsing with hidden syntax. In *SIGMOD Conference*, pages 107–118, 2004.
- [26] A. V. Zhdanova and P. Shvaiko. Community-driven ontology matching. In *ESWC*, pages 34–49, 2006.
- [27] P. Ziegler, K. R. Dittrich, and E. Hunt. A call for personal semantic data integration. In *ICDE Workshops*, pages 250–253, 2008.

# Knowledge Engineering to Visualize Complexity for Legacy Modernization Planning

Sarah B. Lee, Sajjan G. Shiva  
Department of Computer Science  
University of Memphis  
Memphis TN, USA  
sblee@memphis.edu, sshiva@memphis.edu

K. S. Braunsdorf  
FedEx Services Corporation  
Memphis TN, USA  
kevin.braunsdorf@gmail.com

**Abstract**— As legacy systems have matured, most have become more complex leading to increased maintenance costs. Companies must find ways to leverage their existing investments and make incremental changes that bring immediate value to the business through modernization efforts. System complexity will dictate the approach to modernization. This paper proposes a knowledge engineering approach for developing a complexity visualization and risk profile that can be used to plan a modernization strategy.

**Keywords**— risk mitigation, knowledge engineering, application complexity, legacy modernization

## I. INTRODUCTION

D. Good in [1] defines a legacy application as “any application based on older technologies and hardware, such as mainframes, that continues to provide core services to an organization.” Legacy applications consume a large share of resources, with sixty to eighty-five percent of the information technology (IT) budget being spent on average for maintaining systems that are unable to meet the changing competitive needs of the business. IT organizations need to reduce the amount they spend maintaining legacy applications and environments [2]. These applications can create havoc within the organization due to a variety of older technologies, multiple user interfaces, disparate data sources, and different authentication processes that cause users to login multiple times [3]. Documentation may be sparse or nonexistent and the knowledge of how an application interacts with the overall set of applications may be unknown.

Modernization efforts attempt to evolve legacy systems when maintenance and enhancement no longer support business goals [4]. Benefits to the business include improving the ability to react to business demand, reducing total cost of ownership, enabling consolidation of services, and reducing reliance on legacy skill sets [1]. A major challenge to the modernization effort is how to introduce new technology solutions and integrate with existing systems without losing valuable data and processes. Organizations need to reuse the content of existing applications and effectively leverage the latest technology solutions [2].

As older systems have been extended, new systems implemented, and infrastructure aged, the complexity of the application landscape has increased. Complexity is a notion that most people understand conceptually, but is very difficult to describe. A complex system involves heterogeneous connectivity with many interacting subsystems. It is difficult to separate the subsystems due to their heavy dependencies on each other [5].

Knowledge engineering (KE) is a phrase coined in 1983 in [6]. Feigenbaum and McCorduck argue that well-engineered knowledge will embody the details and dynamic nature of information resulting into more usable presentations of knowledge [6]. The enormity of the data to be considered in evaluating complexity presents a daunting, if not impossible, manual task. A reduction in the human effort required can be achieved through gathering, selecting, analyzing, synthesizing, weighting, and evaluating data, information, insight, and decisions, and then transforming this collection into knowledge [7].

This paper proposes an approach for developing a complexity model through knowledge engineering. The evaluation of complexity will be used to determine an approach to modernization. Legacy system modernization is an exercise in risk mitigation. Rebuilding a legacy system on a new platform is more time consuming and presents a higher risk, especially for high business value applications. The determination of a modernization approach depends on the risk and the energy required application. Knowledge about the complexity of legacy systems will aid in these decisions.

## II. PROPERTIES OF COMPLEXITY

Complexity is hard to model, but as a context for discussions about the complexity of an IT system it is even harder to quantify and communicate. Elements acquire the properties of complexity through connectedness to other elements in a structure. Elements acquire the properties of complexity via their connectedness to other elements in a structure. Each connection allows the transfer of the property to other element.

Complexity generates delay in response to change. The potential impact of a structural change to an adjacent element is that complexity of the changed element transfers to the

structure at large. A simple change may cascade through the interconnected system growing and shrinking in impact as it moves. The speed of response to the original change is related to the time required to update each affected element and the number of elements that must compensate for the change.

Complexity makes implementation mistakes more likely as the design is harder to quantify and communicate. Communicating the requirements and design of a large, complex structure requires a different approach than for a smaller, less-complex structure. In a less complex structure, there is often little thought about the larger impact of system changes. Larger changes tend to have unintended results, and require more time than many step-wise refinements. The greater the complexity of a system, the higher the energy required to make a change.

Complexity travels like conduction when the link between elements changes structurally, potentially changing more elements than intended. Now the cohesive peers must compensate for the changes in the updated element. Conduction does more to limit structural refinement than any other factor. The fear of unintended results often prevents the positive maturation of an application.

Complexity also travels like convection when an element which is indirectly connected to the target element must make a compensating change. For example routers in a TCP/IP network don't know that any given data stream might have additional or updated fields in it. Likewise a gateway that just passes messages through to a backend may inadvertently pass messages with a different protocol version to clients, unaware of the updated protocol.

Complexity travels like radiation when the effect on a common resource is not obvious because the elements were not considered part of the same structure. A resource may be consumed, limited, or denied by an adjacent element.

The relationship of energy to complexity has been studied in the context of many different types of complex systems, not just those in IT. Tainter in [8] states that energy has always been the basis of cultural complexity. The availability of energy is a constraining factor in the pursuit of any goal. Future investments in complexity require an increase in energy necessary for progressive innovation. This underscores the constraints in the energy-complexity relationship [8]. Reduction of the interconnectedness or the number of unique elements will lower the complexity of a system.

### III. TECHNOLOGY APERTURE

A finite number of people cannot support an infinite variety of technologies and structures. Thus, every organization has a limit to what it can support. The supporting organization must stay within its "span of effectiveness" to ensure its success [9].

Fig. 1 is provided by the authors to assist with conceptualizing a *technology aperture* model. There is a leading edge of new hardware, software, protocols, and staff coming into the organization as new structures are installed. There is also a trailing edge for the oldest elements remaining in the system.

The system components that are newer are placed on the right. On the left is a trailing edge, defined by the oldest components. The remaining elements are linked between these extremes by technology, including hardware, software, protocols and engineering. Connectivity between the elements is represented by the dependencies between them: data flow, physical connections, and infrastructure relationships. The edges are not straight lines; they shift and warp with changes to each border structure.

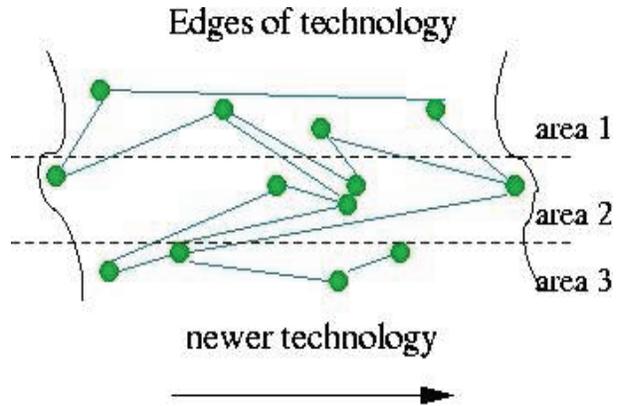


Fig. 1. Technology Aperture Model

The space between the trailing edge and the leading edge contains elements that are connected by the dependencies between them. This area is called the technology aperture. Any given complex IT system's technology aperture tends to grow over time. The energy required to support the aperture grows in proportion to this derived complex structure. It is unthinkable to never advance the right edge, and it is hard to retire elements from the left edge. The most certain result is that complexity is increased until the organization's technical or financial span is inadequate to handle the support energy. To reduce this energy drain for legacy systems, they must be retired or modernized to lower the organization's total structural complexity and run costs.

There are four dysfunctional states, shown in Fig. 2, that break the overall system and should be avoided: sparse, collapsed, disjoint, and entropic. The width of the red triangle represents complexity. The left side represents a system that is more heavily differentiated; the right side represents an overly connected system. The area inside the triangle is the span of effectiveness [9]. The dashed line in the triangle represents the development cusp where the new structure begins to exhibit useful functionality. As a structure becomes more mature the number of elements in the structure tends to grow. There is a progression first to the goal, followed by paths to other less energy-optimal states.

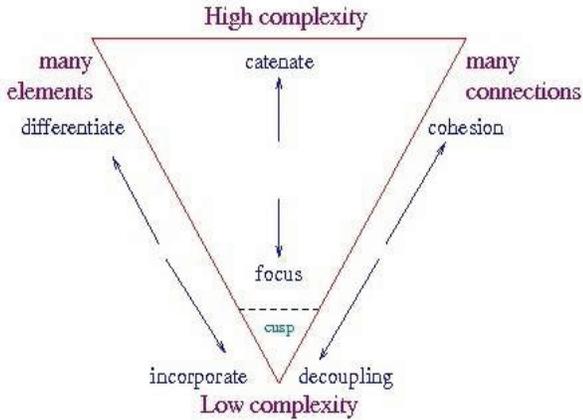


Fig. 2. System States

A new software structure starts out conceptually with very few components. It may have data sources and client services designed to fit next to it, but the new service is not yet existent. This is called the *sparse state*. As development work creates components for the new structure there is a threshold where connectivity between the elements in the structure allows some basic function.

A structure is continually refined until it meets the *goal state*, which is defined as a state where the structure fulfills its objective, and has a positive balance of input energy to productive output, also known as return on investment (ROI). Requests for new features will move the goal away from the structure's present effective zone.

When a structure becomes energy-inefficient, the cost to make progress is no longer worth the value gained. New versions of the structure might require massive rework, re-tooling, and retraining to maintain the key functions. This state is the *collapsed state*: the structure is too general due to functionality that was not part of the original objectives.

If a structure picks up unrelated functions by adding a new interface, it might force the user to create the data flow between the elements, such as having to cut and paste data from one input screen to another. This is the *disjoint state*: the parts of the structure are not connected together in a useful way.

The last labeled state is also a high energy state, but the input energy comes from the development team rather than the users. In the *entropy state*, a structure is so complex that no development effort can refine the structure to create progress. No amount of energy available at present can be applied to the structure to get a net positive return from the existing structure.

#### IV. APPROACH

Moving from theoretical concepts on complexity to a practical application is not without challenges. Scherf describes the general challenge well in [5]: “The complexity depends on the prior knowledge and expectations about the system and the difficulty to get information, to find an adequate description, and to derive an appropriate internal model of the system behavior. Without sufficient or correct

prior knowledge, surprises may occur frequently, which makes it difficult to deal with the complex system and to fulfill a given task.”

Initial sources of data for this experiment include an application inventory database in a large corporate IT organization. Included in the inventory are the attributes that depict which development and business groups are responsible for each application. Technical details about direct interfaces are also available.

The topology of software systems can be adequately represented as a directed graph when all of the interacting subsystems are known. The number and type of interactions between subsystems are significant factors in determining system complexity [10]. Ma, He, and Du describe in [10] a method for using structure entropy for measurement of structural complexity. Parameters listed for this method include connectivity, ripple degree, abstraction degree, and edge weight. Connectivity is simply the number of direct connections that a node in the directed graph has. Ripple degree is defined as the set of nodes reachable by a depth-first search starting at a node  $v$ . This set is called the reachability set [10].

The weight of an edge in the graph is defined using the significance of the type and purpose of the interaction [10]. For this experiment, the edge weight is defined using the technology type for each interface, along with the directionality of connections between applications. For each type of interface, a weight was assigned. Values of ‘unknown’ or ‘null’ are given a higher weight to flag these components as requiring manual investigation to verify the technology.

For each application, the number of interfaces ( $n$ ) and a directionality metric ( $d$ ) for each type of technology was determined. If a node has bidirectional interfaces,  $d = \text{degree}$ . If a node has multiple interfaces with the same node,  $d = \text{degree}$ . Otherwise,  $d = 1$ . The number of interfaces ( $n$ ) is multiplied by  $I$ , which represents the value of the weight assigned to each interface type. The interface complexity for each application is defined as  $\sum (I * n) (2^d)$ .

An adjacency list, representing all edges, was built for the directed graph that is comprised of all the applications and their interfaces. The betweenness centrality was calculated and used as a complexity metric. An application with high betweenness has a lot of influence over the system flow. If a node occurs on many shortest paths between other nodes, it will have a higher betweenness value.

From data in the application inventory, the following items were identified as valuable in the quest for knowledge about legacy applications: application state, last date of a software change, and number of development resources required to maintain the application. If an application state is *containment* or *retirement*, and there are no recent changes noted, a conclusion is derived that there is no work being done to replace or retire the application. If the application state is *mainstream* and no recent change is recorded, the resources required to maintain the application must be evaluated. If the number of maintenance resources is relatively high, the application has a high cost to maintain and might be a candidate for modernization.

## V. ANALYSIS

To identify the complexity of each application and the company's tolerance for risk, a scatter plot was developed. The color of each point indicates the interface complexity. The interface metric values were binned with the lower values represented by a lighter shade of color in the graph. This metric reflects the potential impact of changing the application.

The higher on the y-axis graph that a point resides, the higher the betweenness centrality, indicating a higher risk to the company. Points are plotted on the x-axis based on the business value determined from the application's business domain attribute. To identify the business value, the business domain for each application was identified and assigned a weight that represents the potential risk to business continuity. This provides a basis for determining the company's tolerance for risk for each application.

Analysis of available data revealed that the majority of the applications reflect an application state of *mainstream*. For those marked 'to be retired' or 'contained', the value of last change date should be considered along with the number of maintenance resources. No recent change indicates little effort to retire or contain. A determination must be made as to whether the cost to change a system, either through replacing the business logic with another application or reengineering the current app, may be greater than the cost of living with the current state.

Knowledge gained from this analysis enables mapping of each application to the quadrants in Fig. 3. A modernization strategy can then be assigned. For the 'High Complexity/Low Risk' category, the organization can begin to decouple components as upgrades are implemented. Every change should include an objective of reducing complexity. For 'Low Complexity/Low Risk', complexity may be increased temporarily in order to reach a modernized goal state. Newer components can be added to build a bridge to the next generation structure. For 'High Complexity/High Risk', a recommended approach is replacing part of the structure with 'black box' parts to hide the details. Finally, the 'Low Complexity/High Risk' quadrant represents systems where older components should be replaced with newer, faster components.

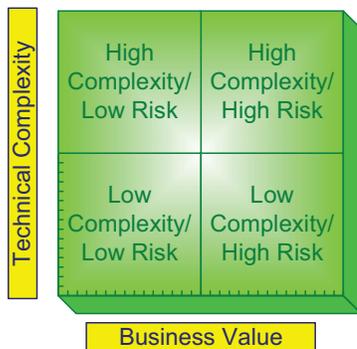


Fig. 3: Application Mapping Quadrants

## VI. SUMMARY

This experience has shown the value of knowledge engineered from legacy application metadata. The goal of determining the risk associated with each application was realized and successfully applied to modernization strategy planning, supporting the goal of reducing complexity in domains with high business risk. The amount of data for most modernization projects will be too large for normal human consumption and processing. Knowledge engineering of application meta-data into estimates of resource impact and potential risks and costs is applicable in a wide array of problem domains.

### ACKNOWLEDGMENT

The views expressed herein are those of the above named authors and not necessarily those of their employers.

### REFERENCES

- [1] D. Good, "Legacy Transformation", CLUB DE INVESTIGACIÓN TECNOLÓGICA, 2002, [Online] Available: <http://www.semdesigns.com> [Accessed: March 17, 2010].
- [2] V. Anxolabehere, "Modernization of Legacy Applications: When Is the Right Time? ", 2009, [Online] Available: <http://wwpi.com/> [Accessed: March 17, 2010].
- [3] M. Kavis, "Modernizing Legacy Applications for SOA," *SOA Institute*, June 25, 2009. [Online] Available: <http://www.soainstitute.org/articles/article/modernizing-legacy-applications-for-soa.html>. [Accessed: March 15, 2010].
- [4] V. Iyer, "Legacy Modernization: Modernize and Scale", April 2008 [Online] Available: <http://www.infosys.com/microsoft/resource-center/Documents/legacy-modern.pdf> [Accessed March 22, 2010]
- [5] O. Scherf, "Complexity: A Conceptual Challenge", In: *Understanding Complexity in the New Millennium: Proceedings of the 2000 World Congress of the System Sciences* (J. Wilby and G. Ragsdell, eds.). Kluwer Academic Press, London.
- [6] E. Feigenbaum and P. McCorduck (1983), *The fifth generation* (1st ed.), Reading, MA: Addison-Wesley, ISBN 9780201115192.
- [7] N. Ashrafi, P. Xu, M. Sathasivam, J. Kuilboer, W. Koelher, D. Heimann, F. Waage, "A Framework for Implementing Business Agility through Knowledge Management Systems," in Proceedings of the 2005 Seventh IEEE International Conference on E-Commerce Technology Workshops.
- [8] J. Tainter, "Complexity, Problem Solving, and Sustainable Societies," 1996, from GETTING DOWN TO EARTH: Practical Applications of Ecological Economics, Island Press, 1996; ISBN 1-55963-503-7 [Online]. Available: <http://dieoff.org/page134.htm>. [Accessed: January 20, 2010].
- [9] W. Livingston, *The New Plague*, Bayside, NY: F.E.S. Limited Pub., 1986.
- [10] Y. Ma, K. He, D. Du, "A Qualitative Method for Measuring the Structural Complexity of Software Systems based on Complex Networks," In Proceedings of the 12<sup>th</sup> Asia-Pacific Software Engineering Conference, Dec. 2005, p. 7.

# Using QVT for adapting question analysis to restricted domain QA systems

Katia Vila  
Department of Informatics  
University of Matanzas, Cuba  
kvila@dlsi.ua.es

Jose-Norberto Mazón and Antonio Ferrández  
Department of Software and Computing Systems  
University of Alicante, Spain  
{jnmazon,antonio}@dlsi.ua.es

**Abstract**—A Question Answering (QA) system must provide concise answers, from a set of text documents, to questions stated by the user in natural language. Current QA systems are oriented to answer open-domain questions, but question analysis must be adapted to the characteristics of a restricted domain for being successfully applied to real-world scenarios. Unfortunately, research addressing question analysis has only produced ad-hoc solutions and no significant attention has been paid to develop comprehensive approaches to effortlessly carry out this adaptation. To overcome this drawback, this paper presents a model-driven approach for systematically and automatically adapt QA systems to new domains by means of a set of model transformations established by using the QVT (Query/View/Transformation) language.

**Keywords**-question answering; metamodeling; model-driven development; model transformations

## I. INTRODUCTION

Question Answering (QA) is defined as the task of extracting the right answer to a specific question in natural language from a collection of text documents or corpus [1]. QA systems can be classified into two types, depending on the application domain: Open-Domain Question Answering (ODQA) and Restricted-Domain Question Answering (RDQA) systems. While the former is concerned about a wide spectrum of questions, the latter is properly tuned to specific domain features [2]. A common architecture for these systems (see Fig. 1) consists of (i) an *indexing module* which includes lexical, syntactical or semantic analysis to be applied to the document collection; and (ii) a *searching module* that carries out every required process to answer a user's question: question analysis (i.e., detecting the expected answer type), information retrieval (i.e., retrieving the relevant passages), and answer extraction (i.e., finding the expected answer from the retrieved set of passages). Importantly, to properly carry out the question analysis in this searching module, patterns<sup>1</sup> that contain knowledge from the domain are required.

Therefore, *adaptability* of patterns by using knowledge from a specific domain is the cornerstone of deriving RDQA from ODQA systems. Furthermore, obtaining a RDQA systems may suffer from *portability*, *reusability*, *productivity*,

<sup>1</sup>Note that for the purpose of this paper, we will refer to all the possible strategies for detecting relationships between elements in the question (e.g., logic forms, regular expressions, syntactical relations, etc.) as patterns.

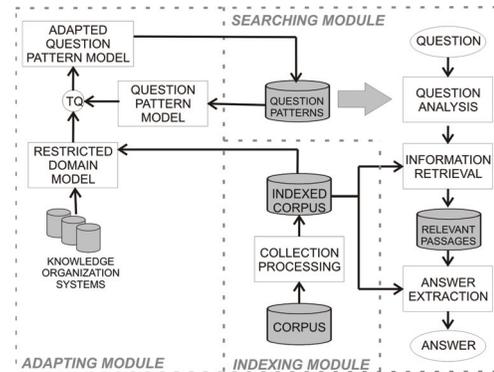


Figure 1. Overview of a generic ODQA system.

*integration* and *interoperability* problems as is detailed in Sect. II. Interestingly, similar drawbacks have attracted the attention of software engineering researchers for years, who has focused on developing techniques for overcoming them. Therefore, our initial hypothesis is that QA systems, as software systems, can borrow software engineering techniques to deal with knowledge sources in order to easier create new patterns to obtain RDQA systems. Specifically, the adaptation of ODQA patterns to a new domain can be seen as a model-driven development scenario in which models of question patterns are defined and managed in an easy and structured way with a high degree of automatization. Model-driven development addresses the complete life cycle of software development by using models [3]. The foundation for creating these models in a meaningful, precise and consistent manner is provided by metamodeling, while defining relations that must hold between the metamodel elements, in order to automatically derive target models from source models in a correct manner, is carried out by model transformations such as the MOF 2.0 Query/View/Transformation (QVT) language<sup>2</sup>.

Bearing these considerations in mind, in this paper, we propose to include a new module in the QA system: the *adapting module*. This module implements our novel approach for using model-driven development in order to adapt existing ODQA patterns of the question analysis to new

<sup>2</sup><http://www.omg.org/cgi-bin/doc?ptc/2005-11-01>

domains. The main benefits of our approach is that the adaptation of patterns is automatically done in a systematic, well-structured, and standard manner, which significantly reduce the amount of manual labor required in adapting QA systems to a restricted domain by using an innovative point of view borrowed from software engineering.

## II. RELATED WORK

Nowadays, question analysis is mainly done in the ODQA by using linguistic knowledge previously acquired and codified in form of patterns by an expert in a manual manner. Therefore, *adaptability* of patterns by using knowledge from a specific domain is the cornerstone of deriving RDQA from ODQA systems. Furthermore, RDQA systems may suffer from other concerns: (i) Patterns are created for a specific QA system [4], so a great knowledge about the code of the QA system is required which hinders the task of tuning the patterns for a new domain and prevents the *reusability* and *portability* of patterns in QA systems from other domains or from ODQA systems. (ii) The process of creating new patterns for a specific domain in a manual fashion [4], [5], is time-consuming and prone-to-fail, which affects *productivity*. (iii) Patterns are tuned by using different kinds of knowledge resources from a specific domain (also known as Knowledge Organization Systems, KOS [6]); according to the level of detail or granularity of the knowledge they refer to, two kind of KOS exist: generic KOS (such as WordNet<sup>3</sup>) or the more precise domain KOS (such as Agrovoc thesaurus<sup>4</sup> for the agricultural domain). However, these KOS have their own formats and interfaces, which must be unified by the QA system, thus being a costly task [1], [2], which poses a problem of *integration* and *interoperability* of KOS.

To overcome these drawbacks, we propose a model-driven development based on a set of QVT transformations to automatically adapt question analysis in ODQA system to a restricted domain from the collection of documents by taking into account the available knowledge resources.

## III. ADAPTATION TO RESTRICTED DOMAINS

Our approach consists of an *adapting module* (see Fig. 1) that allows to adapt an ODQA to a restricted domain. This module is based on modeling restricted domain terms and concepts and question patterns. These models must conform to a restricted domain metamodel and a question pattern metamodel, respectively. Furthermore, a set of QVT transformations are defined upon these metamodels to obtain models of the question patterns adapted to the new domain.

<sup>3</sup><http://wordnet.princeton.edu/>

<sup>4</sup><http://www.fao.org/agrovoc/>

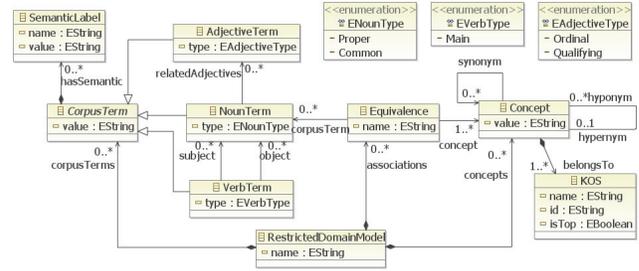


Figure 2. Overview of our Restricted Domain Metamodel.

### A. Restricted domain metamodel

Our purpose is creating models that are useful for representing terms from the restricted-domain corpus and joining them with their corresponding concepts from the knowledge resources. To this aim we have defined the *Restricted Domain* metamodel which contains the adequate elements to create a variety of these models (see Fig. 2). The core element in this metamodel is the *RestrictedDomain-Model* metaclass which is useful for creating a model for a particular restricted domain. The *CorpusTerm* metaclass refers to the relevant terms of the corpus, which can be adjectives (*AdjectiveTerm* metaclass), nouns (*NounTerm*) or verbs (*VerbTerm*). Moreover, the *VerbTerm* metaclass has relations to indicate which *NounTerm* can be seen as subject or as object. Also the *NounTerm* can be related to an adjective which modifies it. Every kind of *CorpusTerm* has its own type (coming from several *Enumerations* as shown in Fig. 2). Finally, every *CorpusTerm* may have some semantic information (*SemanticLabel* metaclass). Furthermore, *Concept* and *Equivalence* metaclasses allow that the elements of this *Restricted Domain* metamodel must be semantically enriched with concepts and relationships from several KOS. The *Concept* metaclass refers to an element from a particular KOS. Each of these elements has a *value* to represent it. Besides, each concept can be related to one or more concepts through relations of synonymy, hypernymy and hyponymy. Each concept may be related to more than one *KOS* for which the *name* is indicated and also an *ID* for the concept within this *KOS*. This *KOS* metaclass has an *isTop* attribute that states if the concept is a top concept in that *KOS*. Equivalences between a term and a concept can be defined. To this aim the metaclass *Equivalence* represents an association between *Concept* and *NounTerm*.

Throughout this section we will use a case study to exemplify our approach. It is based on the agricultural domain from a corpus of the Cuban Journal of Agricultural Science<sup>5</sup>. In order to define a restricted domain model, first relevant terms must be obtained. Sample relevant terms from our case study are: “chlorimuron”, “sulphonylurea”,

<sup>5</sup><http://www.ica.inf.cu/productos/rcca/>

“group”, “division” and “growth” as common nouns; “inhibit”, “control” and “belong” as main verbs; and “chemical” and “cellular” as qualifying adjectives. Also, the following syntactical information was extracted: “group” is related to the “chemical” adjective, “division” and “growth” nouns are related to the “cellular” adjective, while “sulphonylurea” noun is subject of verbs “inhibit”, “control” and “belong”. This information is stored in the corresponding *CorpusTerm* classes of the restricted domain model. The noun term “chlorimuron” is checked to appear in the domain KOS (Agrovoc in this case). It is found, so a new concept “chlorimuron” is created and also a new equivalence between both. Then, the KOS is navigated by its broader term relationships in order to find those concepts which are hypernyms. The concept “sulphonylurea” is then found, and also “sulphonamides” and “amides” as top concept in this domain KOS. Afterwards, this top concept is intended to be mapped to some concept in the generic KOS (in this case WordNet). It is found “amide”. Then, by recovering hypernymy relations, concepts “organic compound”, “chemical compound”, and “substance” as top concept in the generic KOS are found.

### B. Question pattern metamodel

Open-domain question patterns must be represented into a model in order to be able to adapt them to the domain represented in a restricted domain model. To this aim we have defined the *Question Pattern* metamodel which contains the adequate elements to create a variety of these question pattern models. These models will define the system question typology, i.e. question types that the system will be able to classify, thus detecting the kind of expected answer and the key words of the question. Fig.3 shows this metamodel. A pattern is represented as a *Pattern* metaclass in order to have several associated expressions (i.e. *Expression* and *Association* metaclasses) which represent a pattern. Moreover, a pattern is associated to an answer type (i.e. *AnswerType* metaclass), in such a way that the kind of expected answer is known when the classification of the question by choosing the pattern that best fits with the question. A metaclass *Expression* is used to consider every kind of expressions. For example, syntactical labels such as PP-preposition, PtDt-interrogative pronoun or determinant, VBC-verbal head, SNP-simple noun phrase, SPP-simple preposition phrase and their values (e.g., an expression PtDt could have “which” as value). Expressions may have some related concepts (e.g., a SNP may have hyponyms of certain concepts within their expression). A metaclass *Association* relates expressions in order to know a sequential order. It has an antecedent and a consequent. An *AnswerType* metaclass refers to one or more concepts in order to determine the type of the answer. A metaclass *Concept* that contains one or more IDs stored in the attribute that identifies different KOS where this concept is supposed to appear and also several verbs that frequently are associated to the concept.

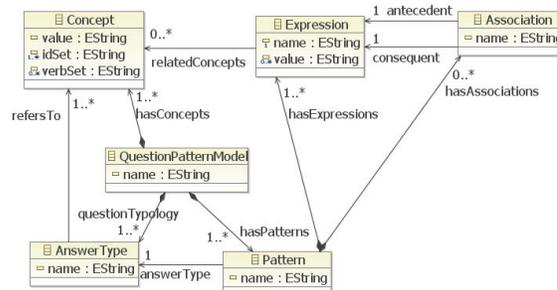


Figure 3. Overview of our Question Pattern Metamodel.

An example of pattern from our previously developed ODQA system (AliQAn [5]) is called “patternEO1”, and it has the following expressions: PtDt (value: “which”), VBC (value: “be”) and SNP (whose related concepts are “musical\_instrument” and “building”); as associations “PtDt-VBC” and “VBC-SNP”; and the answer type is “entity\_object”. Lastly, it is worth noting that this pattern makes ODQA system able to identify questions of kind: “Which is the musical instrument that Beethoven was playing?”.

### C. QVT for adapting question patterns to restricted domains

A set of QVT relations has been developed to adapt question pattern models to restricted domain: *ODQuestionPattern2RDQuestionPattern*, *Hyponym2QuestionPattern*, and *DomainConcept2PatternConcept*. These relations collect both the restricted domain model, and models of existing open-domain question patterns, in order to generate new pattern models specifically tuned for the domain. The *ODQuestionPattern2RDQuestionPattern* QVT relation (see Fig. 4) fixes the level of granularity to define a new question typology since in restricted domain is highly advisable that these kinds of taxonomy are refined in order to improve the precision of the system. For example, if a taxonomy level of 1 is chosen then every concept without hypernym is considered for the new pattern and if a more refined taxonomy is required then the selection criteria could be more restricted. This criteria is reflected in the *when* clause of the *ODQuestionPattern2RDQuestionPattern* relation. Specifically, those concepts that have a number of hyponyms greater than 2 are chosen ( $c0.hyponym.size() > 2$ ) from the restricted domain model and matched with some concept from the question pattern model according to other precondition stated in the *when* clause: an hypernym of the concept from the restricted domain model corresponds to a concept in the question pattern model or one hypernym ( $c0.hypernym.value=c1.value$  or  $c0.hypernym.value=c1.hypernym.value$ ). This last precondition is useful to search for relationships between concepts which have been added to the adapted question pattern model (new concepts) and those existing concepts in the question pattern model (old concepts) which allow

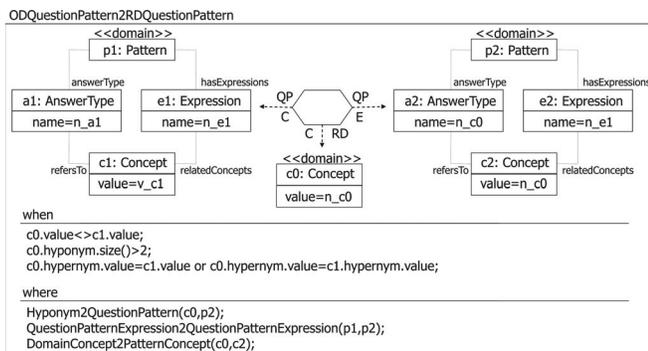


Figure 4. Obtaining a restricted-domain question pattern.

deciding if patterns of an old concept can be used to define patterns for a new concept. Assuming that every top-level concept of the generic KOS used by the ODQA system has patterns, we can follow the criteria that if the old concept and the new concept have a common hypernym provided by the same KOS, i.e. they are siblings in the hierarchy, then top-level patterns of the old concept can be reused. Once the *when* clause holds, concepts from the restricted domain model (*Concept* c0) and the question pattern model (*Concept* c1) are matched in the source models, then the following set of elements are created in the target model: *Pattern* p2, *AnswerType* a2, *Expression* e2, and *Concept* c2. While p2, a2 and e2 correspond to elements in the source question pattern model, c2 takes its value from the restricted domain model. Finally, the *where* clause allows to execute the following QVT relations to complete the new question pattern: *Hyponym2QuestionPattern*, and *Domain-Concept2PatternConcept*. Also the *QuestionPatternExpression2QuestionPatternExpression* relation is executed. Since this relation simply deals with expressions of the source question pattern model in order to be created in the target pattern model, it is not further explained in this paper.

Recalling our case study, concept “sulphonylurea” has “substance” as hypernym top concept which is not among old concepts. However, “object” is an old concept with defined patterns and it has the same top concept hypernym than “substance” which is “physical entity”. Concepts “substance” and “object” are then siblings and patterns from “object” can be reused to the new concept “sulphonylurea”. Therefore, a new pattern for the “sulphonylurea” concept is created from the old pattern previously explained for the “object” concept (i.e. “patternEO1”). The name of the new pattern is “patternSulphonylurea1”, and it has the following expressions: “PtDt”, “VBC” and “SNP”; as associations they have “PtDt-VBC” and “VBC-SNP”; and the answer type is “substance\_sulphonylurea”. For our case study, values for the expressions of the new created pattern are obtained based on the *QuestionPatternExpression2QuestionPatternExpression* relation:

PtDt (value: “which”) and VBC (value: “be”). The *Hyponym2QuestionPattern* QVT relation aims to include in the target question pattern model those concepts that are hyponyms of the concept created in the previous relations. When this relation is applied to our case study, the following concepts related to the SNP expression of the new pattern are obtained: “sulphonylurea”, “chlorimuron”, “bensulfuron”, and “chlorsulfuron”. Finally, the *Domain-Concept2PatternConcept* QVT relation checks concepts and related information in the restricted domain model in order to fill information in concepts of the target question pattern models. When executing this relation with the “sulphonylurea” concept, the verb set “inhibit”, “control” and “belong” is stored in the corresponding *Concept* class of the new pattern model. Interestingly, once our approach is applied to the case study, a new kind of restricted-domain question can be answered by the QA system, e.g., “Which is the sulphonylurea used for the weeds control in the soja crops?”.

#### IV. CONCLUSIONS AND FUTURE WORK

In this paper we have presented our model-driven approach for using the QVT language for tackling the complex task of automatically adapting an ODQA system to restricted domains in a systematic, well-structured, and comprehensive manner. Our short-term future work consists of carrying out a set of experiments to measure the effectiveness of our approach in the whole process of QA in order to compare it with manually adapting the QA system.

#### ACKNOWLEDGMENT

This research has been partially funded by the Valencia Government under project PROMETEO/2009/119, and by the Spanish government under project Textmss 2.0 (TIN2009-13391-C04-01).

#### REFERENCES

- [1] D. Mollá and J. L. Vicedo, “Question Answering in Restricted Domains: An Overview,” *Computational Linguistics*, vol. 33, no. 1, pp. 41–61, 2007.
- [2] D. Ferrés and H. Rodríguez, “Experiments Adapting an Open-Domain Question Answering System to the Geographical Domain Using Scope-Based Resources,” in *Multilingual Question Answering Workshop of the EACL 2006*.
- [3] J. Bézivin, “On the unification power of models,” *Software and System Modeling*, vol. 4, no. 2, pp. 171–188, 2005.
- [4] A. Peñas *et al.*, “Overview of ResPubliQA 2009: Question Answering Evaluation over European Legislation,” in *Working Notes of Cross Language Evaluation Forum (CLEF)*, 2009.
- [5] S. Roger *et al.*, “Using AliQAn in Monolingual QA@CLEF 2008,” in *CLEF*, 2008, pp. 333–336.
- [6] G. Hodge, *Systems of Knowledge Organization for Digital Libraries: Beyond Traditional Authority Files*. The Digital Library Federation Council on Library and Information Resources, 2000.

# Towards an Automation of Software Evolution Good Practices

Chouki Tibermacine  
LIRMM, CNRS and Montpellier University, France  
Chouki.Tibermacine@lirmm.fr

Soraya Sakhraoui  
University of Setif, Algeria  
Soraya.Sakhraoui@univ-ubs.fr

Vincent Le Gloahec  
Alkante SAS and VALORIA, University of South-Brittany, France  
v.legloahec@alkante.com

Régis Fleurquin  
IRISA, INRIA Rennes, France  
fleurquin@irisa.fr

Salah Sadou  
VALORIA, University of South Brittany, France  
sadou@univ-ubs.fr

## Abstract

*It is well known that software evolution is an inescapable activity in the software lifecycle. In order to prevent the negative effects of this activity (decreased quality, increased complexity, etc.), some good practices have been recommended in the past. In this paper, we present a method which aims at automating a kind of assistance to software evolution. This assistance makes it possible to guide the developer when applying changes on a given software by making persistent some good practices, which can be considered as some kind of knowledge in the software engineering practice. In this method, a domain metamodel is firstly introduced. A set of constraints formalizing the good practices are then associated to this metamodel. Together, these two elements compose the basis upon which the automatic support for the evolution assistance has been built.*

## 1 Introduction

The study of software evolution, as a first-class phenomenon, started at the end of the sixties. The need for studying software evolution is motivated by the society's increasing dependency on software that implements business processes, which are naturally subject to evolution. This important activity has high costs within a software project budget. These are estimated nowadays for more than 80 % of the global cost [4, 13].

To reduce the evolution cost there have been some proposals for good practices. These "good practices" correspond to an experiment-based methodology which leads a given software evolution to the desired result at a lower cost.

For this purpose, Lehman's works [8, 6] consisted in undertaking several empirical studies on software evolution [7, 8]. From these studies he specified laws and good practices for improving the software evolution process. These good practices, which are considered as units of knowledge acquired through a long experience on evolving existing software systems, have been presented in the form of recommendations addressing different contexts. Examples of good practices include : "the necessity to monitor the number of user-generated fault reports per release in order to check if the fault rate is increasing".

It is interesting to specify a catalog of good practices, but leaving their checking to humans may lead to errors. The causes of errors are multiple. This can be simply the non-use of the rules or their bad interpretation. Indeed, rules defined textually in a natural language can lead to this kind of situation. In order to avoid this, it will be necessary to make an automatic checking of these recommendations. For that aim we have to: i) formalize the description of the good practices; ii) provide a tool able to interpret these descriptions in order to automate the checking of the compliance of the evolution process with the good practices.

In this paper we propose to formalize the description of good practices as a set of rules using meta-models (Section 3) and the OCL language [10] (Section 4). To automate the verification of compliance with good practices, we propose a tool designed as an Eclipse plugin (Section 5). Based on the information system of a software project and the set of rules, this tool checks the compliance of the project with the good practices. Before concluding this paper, we present some related work in Section 6.

In the following section we describe the main principles of our approach.

## 2 Proposed approach

In [14], we proposed to formalize the link between non-functional properties and their corresponding architectural choices, in order to limit the effects of software aging [12]. Using a tool that checks, at every stage of a development, the validity of the architectural choices, it is possible to warn on the non-functional properties potentially affected. By regulating possible actions in response to these warnings, we have built a control system which contributes on the one hand to updating the documentation and on the other hand to checking the non regression of the system. We have shown that this kind of validation significantly increases the chance to reach a well-documented solution, complying with the new requirements, while preserving the quality attributes that should not be tampered with. Thus, we provide a way to induce compliance with the good practice which consists in *"systematically checking, during an evolution process, the consistency of the software's non-functional requirements specification with its existing design"*.

Our aim is to generalize this mode of automatic control to other good practices described in the literature such as those introduced by Lehman's laws or those referenced in models of maturity-level assessing, such as CMMI [3].

### 2.1 General architecture of the framework

Good practices specify a number of actions or measures that improve the effectiveness of the evolution process. Ideally, starting from a formal expression of good practices, we should be able to automatically generate the code needed to check their respect in the tools used to achieve the evolution.

To concretize this goal, we propose the following means to software developers:

1. a language for documenting good practices they wish to see implemented during an evolution process. This documentation must be in a format that is independent from any software project or an IDE (Integrated Development Environment, just like a PIM (Platform-Independent Model) in MDE (Model-Driving Engineering)).
2. a tool able to automatically translate or help in the translation of these good practices in the context of a given project or a tool. This translated documentation corresponds to a PSM (Platform-Specific Model in MDE) of good practices.
3. Finally, a tool able to interpret the PSM good practices and to introspect the current project and the IDE's underlying information system to check if it complies.

### 2.2 Mapping to a particular case

To experiment the proposed approach, we have taken the following steps:

1. We have defined a metamodel, called SEMM (Software Evolution MetaModel), which identifies the concepts manipulated by the good practices defined by Lehman [8] for the software evolution process. This metamodel will be described in the next section. This is the basis for writing rules of good practices regardless of any tool (PIM format).
2. We have described the rules of good practices as OCL constraints on SEMM.
3. We have produced the metamodel of AlkoWeb<sup>1</sup> development tool which is used by our industrial partner. This metamodel defines the concepts handled by the project information system of AlkoWeb (model, view, version, author, date, etc.).
4. We have translated the constraints written on the basis of SEMM to their equivalents in AlkoWeb's metamodel.
5. We have developed an Eclipse plugin, called SEGPE, which checks the compliance with these rules of the evolution of a design defined with AlkoWeb.
6. Finally, we have used SEGPE on a particular project.

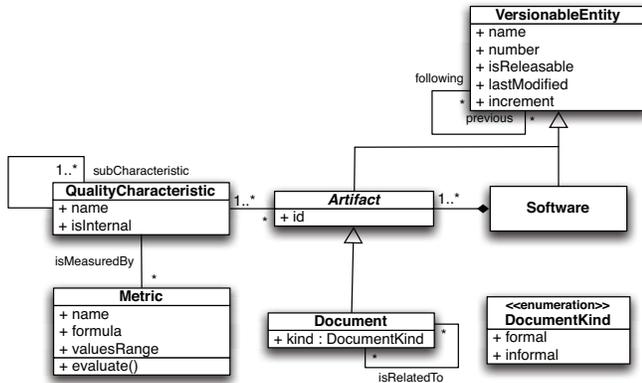
In the following, we present SEMM, examples of constraints corresponding to the rules of good practices written on this metamodel, the tool developed for writing and checking these constraints, and an experimentation made on a concrete project.

## 3 Domain Metamodel

An important step towards the automation of the good practices is to specify the relations between various concepts in software engineering (software, process, activity, quality, etc.). We do this by defining a metamodel (SEMM) for this domain. As described in Section 2, a set of concepts was constructed. These concepts form the entities of the metamodel and are represented as metaclasses. As the concepts were classified according to the context to which they are addressed, the complete metamodel was subdivided in two parts. Each part covers a particular aspect of the domain.

Figures 1 and 2 represent the different parts (aspects discussed above) of the domain metamodel. Assembled, they represent a complete illustration of SEMM. Each aspect is discussed in the following sub-sections.

<sup>1</sup>This tool is briefly described in section 5.



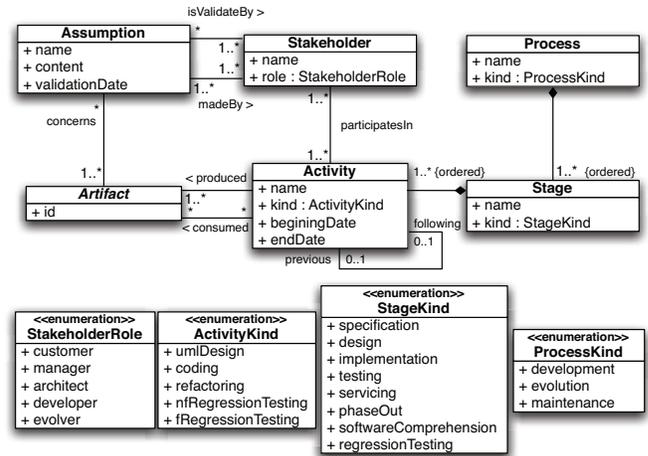
**Figure 1. Domain Metamodel -part 1: software product aspect-**

### 3.1 Software Product Aspect

In figure 1, the metamodel represents a software product as a non-empty set of artifacts. A software and an artifact are considered as versionable entities, having thus a name and a number. A versionable entity can be releasable or not. It becomes a releasable entity when some major changes are made throughout several versions. It has a date meta-attribute (*lastModified*), which corresponds to the date when the software entity has been versioned (the date of a commit in an SVN system, for example). The *increment* represents the modifications made on the software or on one of its artifacts since the last version. Given a versionable entity, we can navigate in this metamodel to the previous or the following versionable entities. In SEMM, *Artifact* is an abstract class. The concrete concept here is *Document*. A document can be formal (formal specification document, architecture design document, source code, ...) or informal (informal non-functional requirements specification, source code documentation, ...).

To a given artifact is associated one or several quality characteristics. These characteristics have the same semantics as in the ISO 9126 standard<sup>2</sup>. Each characteristic (reliability, portability, maintainability, ...) can be externally or internally visible. It has several sub-characteristics. These are measured by the means of metrics. Each named metric is defined by a formula which is used for its evaluation. A values range can be specified for a given metric. It represents the context-free acceptable values for the metric.

<sup>2</sup>Software engineering – Product quality – Part 1: Quality model. The International Organization for Standardization Website: [http://www.iso.org/iso/iso\\_catalogue/catalogue\\_tc/catalogue\\_detail.htm?csnumber=22749](http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=22749)



**Figure 2. Domain Metamodel -part 2: software process aspect-**

### 3.2 Software Process Aspect

Figure 2 shows the software process part of SEMM. The *Artifact* metaclass is the common class between the two parts of the metamodel (the process and the product parts). In this part of the metamodel, this concept is seen as a dynamic entity which is managed within some software actions. In SEMM, we abstract three distinct levels of granularity for actions where artifacts are managed. A *Process* is a set of ordered elements of type *Stage* and a *Stage* is a set of ordered elements of type *Activity*. A process in this metamodel is named and has a kind. It can be a development, a maintenance or an evolution process. A stage (part of a software process) is named and can be of different kinds: specification, design, implementation, testing (for software development process), servicing and phase-out (for evolution process), and software comprehension and regression testing (for maintenance process). An activity is the most atomic action. It represents the low-level operations performed within a software stage (see Figure 2).

Starting from each activity, we can get the eventual previous or following activity. An activity is characterized by a beginning date, where it can consume several artifacts, and an end date, where it produces one or several artifacts. A stakeholder like a project manager, a customer or a developer can participate in one or several activities. During her/his implication in an activity, a stakeholder can make some assumptions. Each assumption is named, has a content and a validation date. Different stakeholders can be responsible for the validation of assumptions made on some artifacts.

## 4 Expression of Constraints

We associated to SEMM a set of constraints. These constraints correspond to formal expressions of the rules representing the good practices of software evolution, usually expressed in the natural language. First, we briefly introduce the constraint language used in this work. We illustrate then two examples of constraints defined with this language upon the metamodel presented previously.

### 4.1 Constraint Language: OCL

As stated at the beginning of this paper, the constraint language used for formalizing software evolution good practices is the OMG's Object Constraint Language. The choice of this language is motivated by its simplicity and the presence of reliable support tools. OCL is a first order logic language which allows, in a declarative way, the description of expressions representing constraints for precisizing the semantics of UML models. In our approach, these constraints have as a context an element in the domain metamodel (SEMM). This means that the constraints will be checked on all the models (software processes, software artifacts, ...) instances of this metamodel.

### 4.2 Examples of PIM rules

In this subsection, we show how two examples of Lehman's evolution good practices are formalized using OCL. The first good practice states that [8]:

"... modifications to the specification, assumptions that accompany them and independent assumptions may be recorded in user or other documentation. ... (assumptions) must be confirmed as not being inconsistent with the specification ... (records of changes to individual specifications record additions and changes to the assumption set that underlie them)."

This good practice, which is associated to the first evolution law (Continuing Change) is specified in OCL as follows:

```
context Assumption inv:
self.concernedArtifacts->forall(art |
art.lastModified < self.lastValidated)->isEmpty()
```

In the context of SEMM, this constraint stipulates that each artifact attached to an assumption should have a date of modification prior to the date of last validation of its associated assumption.

We introduce below another example of a good practice defined in Lehman's evolution laws:

"... monitor the number of user generated fault reports per release to obtain a display of the fault rate trend with time. A fitted trend line can then indicate whether the rate is increasing ..."

In OCL and in a simplified form, we can describe this rule, which is associated to the seventh law (Declining Quality), as follows:

```
context Software inv:
self.artifact->select(a | a.isReleasable)
->oclAsType(Document)->select(doc |
doc.name='ErrorReport')->asSet()->size() <=
self.artifact.previous->select(a | a.isReleasable)
->oclAsType(Document)->select(doc |
doc.name='ErrorReport')->asSet()->size()
```

In this constraint, we query the collection of error (fault) reports associated to two successive versions of releasable artifacts. The size of these collections should decrease or be constant.

### 4.3 PIM to PSM translation

Our choice of OCL was also motivated by the possibility offered by this language to express constraints as models that transformation languages, which are MOF QVT-compliant [11], can handle. Thus, the transformation of PIM constraints to their equivalent PSM ones becomes possible by using ATL<sup>3</sup> or Kermeta<sup>4</sup>, for example.

We have formalized many rules of good practices, but we noticed that some of them defined at PIM level have no equivalent in PSM one. Examples include the following rule which states that *"any documentation created must be clear and comprehensible"*:

```
context Software inv:
self.artifact-> oclAsType(Document)
->forall(d|d.qualityCharacteristic
->exists(q|q.name = "Comprehensibility")
and d.qualityCharacteristic
->exists(q|q.name = "Clarity"))
```

As shown above, this constraint is easily expressed with OCL on SEMM. However, it cannot be easily checked on real-world IDEs, because existing tools' information systems do not provide the necessary information for performing this task (quantifying comprehensibility and clarity). In other words, concepts manipulated by this constraint have no equivalent in the PSM metamodel. However, it is important to keep this kind of constraints at PIM level in case of possible tools evolution in order to incorporate concepts needed for their evaluation (possible definition in the future of some metrics for measuring such quality characteristics that are specific to the targeted tools).

<sup>3</sup>ATLAS Transformation Language: <http://www.eclipse.org/m2m/at/>

<sup>4</sup>Kermeta, Triskell Metamodeling Kernel: <http://www.kermeta.org/>

## 5 SEGPE : a tool to automate Good Practices validation

In order to validate the approach presented in this paper, we have implemented a tool named SEGPE (Software Evolution Good Practices Evaluator). This tool is composed of a set of plugins that are integrated in the Eclipse platform. This platform has been chosen for several reasons : i) the Eclipse community focuses researches concerning software engineering and offers many frameworks for supporting and promoting model-based development; ii) our team has already developed Eclipse plug-ins: AURES, an architecture evolution assistant tool [14], and AlkoWeb-Builder, a tool for component-based Web development [5].

AlkoWeb-Builder, built in a partnership project between VALORIA laboratory (University of South-Brittany) and Alkante<sup>5</sup>, has been linked with SEGPE. SEGPE has been built as a plug-in, using two main frameworks provided by the Eclipse platform : EMF (Eclipse Modeling Framework) and MDT OCL (Model Development Tools - OCL).

### 5.1 SEGPE architecture

The metamodel discussed in this paper has been firstly designed as an EMF Ecore metamodel (Ecore is an implementation of the OMG's MOF (Meta Object Facility)). Then, using this Ecore metamodel, a plugin has been automatically generated, which allows to create and edit instances of our metamodel. The way SEGPE has been designed intends to preserve a low-coupling between the metamodel and the plug-in itself. The metamodel can be extended with no consequences on the plugin.

OCL constraints are defined at the metamodel level and are directly incorporated in the meta-classes (Assumption, Artifact, etc...). At this level, the constraints does only need a syntactic validation, to check if they are grammatically correct. When a developer wants to apply good practices on a real project, the plugin allows to create and edit a model which conforms to SEMM. By creating model objects, which are instances of the meta-classes, the constraints are automatically inherited from the metamodel. At the model level, the context of an OCL constraint is given by the model object the constraint is attached to. Although the constraints are defined at the metamodel level, they are in fact evaluated at the model level. The evaluation of the constraints allows to check compliance of the project's state with the described rules.

The evaluation part of the tool has been designed as another plugin which consists of two views. **Good Practices Contract:** used to view the contract's rules (inherited from

the metamodel) in the context of a model object. The view allows to evaluate the contract (all its OCL constraints). It is also possible to launch an evaluation on the whole model, where all model object contracts will be evaluated. **Good Practices Report:** this view is used to display a summary each time a contract is evaluated.

### 5.2 Integration with AlkoWeb-Builder

In order to validate SEGPE on a real-world project, the tool has been associated to AlkoWeb-Builder. AlkoWeb-Builder allows to model a component-based Web application by graphically composing hierarchical components. To do so, we have translated the rules written using SEMM into their equivalents in AlkoWeb metamodel. This translation was rather easy since the same concepts were present on both metamodels. However, the translation has shown some deficiencies in AlkoWeb metamodel. For example, there were no links between artifacts and assumptions on which they depend. This is important in order to check some rules, such as the one presented in section 4. Therefore, we believe that it is important to keep the rules written with SEMM and then translate them into the targeted metamodels. We can thus check the consistency of the targeted metamodels towards the good practice rules.

## 6 Related Work

The good practices in software engineering aim at producing rationally software, by decreasing for example their complexity. Design patterns are a good example of such practices which have as a goal the capitalization of the programming know-how. Following the same philosophy, a *process pattern* suggests a sequence of proven processes that solves a frequently recurring problem [1]. They represent a way to enforce some good practices. This however needs to be automated to prevent human errors. We preferred a solution based on constraints expressing declaratively good practices because, unlike the process patterns, they can express all the space of acceptable solutions.

The reader can see many similarities between our metamodels and OMG's SPEM (Software Process Engineering Metamodel [9]). Indeed, SPEM is intended to metamodel processes of software engineering (like Rational Unified Process), and this is what we partly aimed to do. The main difference between the two kinds of metamodels is that SPEM is a standard for defining processes to allow building standardized tools for managing (authoring and customizing) processes, while the metamodels we presented aim at formalizing good practices of software evolution. The metamodels presented in this paper are voluntarily simplified in order to facilitate the description of constraints. Many abstract concepts present in SPEM are

<sup>5</sup>Alkante is a company specialized in consulting and engineering information technology (Web, geographical and territorial information systems).

thus not present in these metamodels. This makes easier the navigation in these metamodels for describing constraints. Besides, our metamodels cover some aspects not treated (voluntarily) by SPEM designers. Indeed in SPEM, project management is not metamodeled. Authors of the SPEM specification [9] affirm that this aspect does not meet their concerns, and argue that this will make SPEM more complex, because this standard wants to accommodate a wide range of existing software development processes. In our work, the software project management is an aspect which is widely dealt with in the existing software evolution good practices. Hence, we met the need to integrate it in SEMM.

## 7 Conclusion and Future Work

In this paper we presented a language, a method and a tool to automate a kind of disciplined software evolution. By disciplined evolution, we mean here a software maintenance that complies with some good practices. The presented approach is built upon two technologies whose maturing is ever increasing, MOF metamodeling and the OCL constraint language. OCL language has already proven its usefulness in software maintenance [2]. We are convinced that the precision granted by the formality of OCL comes at a much lower cost than when using other formal specification constructs. In addition MOF has become a *de facto* metamodeling standard, and more people are expected to master OCL and use it currently. The set of the formalized good practices is not exhaustive but the approach is flexible enough to be applied at all the good practices found in the literature and practice. New specifications could be added to the metamodel describing the new determined concepts. The methodology presented in this paper helps: (a) to define a formal specification of the good practices that were only informally defined as well as expressing them in a checkable form, (b) to ensure that there are no ambiguities on the context under which the good practices are defined, (c) to propose a library of good practices specifications in a format that can be used and validated by different stakeholders.

In the literature and the practice, there is no work addressing the formalization of software evolution good practices towards their automation. We are convinced that this is a promising research field, and much work should be done in order to be able to formalize as much as possible (and thus automate the checking of) developers' knowledge and experience acquired while participating in software evolution tasks. The work presented in this paper is a contribution in this direction based on simple and standard languages. It targets the only known catalog of software evolution recommendations: Lehman's evolution laws.

As a perspective to this work on the conceptual level, we plan to study other kinds of good practices and thus to extend the existing domain metamodel and constraint set.

On the tool level, an interesting issue that has not been yet explored is how to make a SEGPE model aware of the underlying information system. For the moment, a model's artifact references a physical data (a Java class for example) by its name, but this reference has to be done manually by the person in charge of editing the model. So, what lacks by now is to make a model's artifact aware of any changes of its referenced data. Even though this issue is not trivial, some features provided by the Eclipse platform could be used to keep a model up-to-date with its associated information system.

## References

- [1] S. W. Ambler and B. Hanscome. *Process Patterns: Building Large-Scale Systems Using Object Technology*. Cambridge University Press, 1998.
- [2] L. C. Briand, Y. Labiche, H. D. Yan, and M. Di Penta. A controlled experiment on the impact of the object constraint language in uml-based maintenance. In *Proc. of ICSM'04*, pages 380–389, Chicago, Illinois, USA, 2004.
- [3] M. B. Chrissis, M. Konrad, and S. Shrum. *CMMI: Guidelines for Process Integration and Product Improvement, 2nd edition*. Addison-Wesley Professional, 2006.
- [4] L. Erlikh. Leveraging legacy system dollars for e-business. *IEEE IT Professional*, 2(3), 2000.
- [5] R. Kadri, C. Tibermacine, and V. Le Gloahec. Building the presentation-tier of rich web applications with hierarchical components. In *Proc. of WISE'07*, pages 123–134, Nancy, France, December 2007. LNCS, Springer-Verlag.
- [6] M. Lehman and J. Ramil. Towards a theory of software evolution - and its practical impact. In *Proc. of ISPSE'00*, pages 2–11. IEEE Computer Society, 2000.
- [7] B W Chatters, M M Lehman, J F Ramil, and P Wernick. Modelling A Software Evolution Process : A Long-term Case Study. In *J. of Softw. Proc.: Improvement and Practice*, issue 2-3, pages 95-102, July 2000.
- [8] M. M. Lehman and J. F. Ramil. Rules and tools for software evolution planning and management. *Annals of Software Engineering*, 11(1):15–44, 2001.
- [9] OMG. Software process engineering metamodel, version 1.1, document formal/2005-01-06. OMG Website: <http://www.omg.org/cgi-bin/doc?formal/2005-01-06>
- [10] OMG. Ocl language specification, version 2.0, document formal/2006-05-01. OMG Website: <http://www.omg.org/cgi-bin/apps/doc?formal/06-05-01.pdf>
- [11] OMG. MOF Query / Views / Transformations. OMG Website: <http://www.omg.org/cgi-bin/doc?ptc/2007-07-07>
- [12] D. L. Parnas. Software aging. In *Proc. of ICSE'94*, pages 279–287, Sorrento, Italy, 1994. IEEE CS Press
- [13] R. C. Seacord, D. Plakosh, and G. A. Lewis. *Modernizing Legacy Systems: Software Technologies, Engineering Processes, and Business Practices*. SEI Series in Software Engineering. Pearson Education, 2003.
- [14] C. Tibermacine, R. Fleurquin, and S. Sadou. On-demand quality-oriented assistance in component-based software evolution. In *Proc. of CBSE'06*, pages 294–309, Vasteras, Sweden, June 2006. Springer LNCS.

# VESTA: A View-based Software Quality Assessment Model for Software Evolution Management

Wei-Chung Hu      Chia Hung Kao      Feng Pu Yang  
Hewijin Christine Jiau      Kuo-Feng Ssu  
Institute of Computer and Communication Engineering,  
National Cheng Kung University, Tainan, Taiwan  
{selain, basara, jopwil}@nature.ee.ncku.edu.tw      jiauhjc@mail.ncku.edu.tw

## Abstract

To ensure software evolution happens in a systematic and controlled manner, several software metric suites and quality models have been developed as tools for various purposes to describe, predict, and manage software quality evolution. However, these tools cannot generate the needed data for stakeholders because they are all from the developer's view. This study introduces a *View-based Software Quality Assessment (VESTA)* model to guide the utilization and integration of existing quality assessment tools from various views of stakeholders. With specified software quality views and quality assessment tools, stakeholders can obtain appropriate measurement data they want. VESTA further provides a multi-languages view with the evaluation result on an open source project.

## 1. Introduction

Software evolution, which is inevitable during software development life cycle, refers to the maintenance and enhancement of software systems over lifetime. As software evolves, the changes made to the software must be carefully managed in order to control the activities. Software quality assessment in different perspectives [1, 2, 3, 4, 5, 6] provides important, quantitative evidence for software evolution management. However, to conduct a customized assessment process exclusively for an assessment context is time consuming and expensive. Therefore stakeholders can only use existing assessment tools to obtain the quality data. Unfortunately, existing assessment tools are usually defined under developers' perspectives to the assessment context. These perspectives are referred as *Views* in this study. Current assessment tools are not suitable for various views from stakeholders. For example, the metrics and the corresponding assessment models usually contribute to software qual-

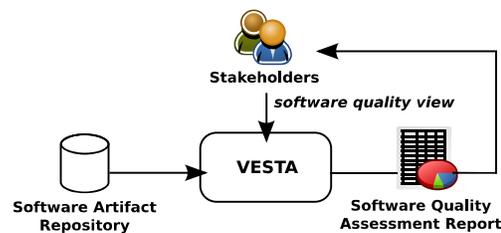
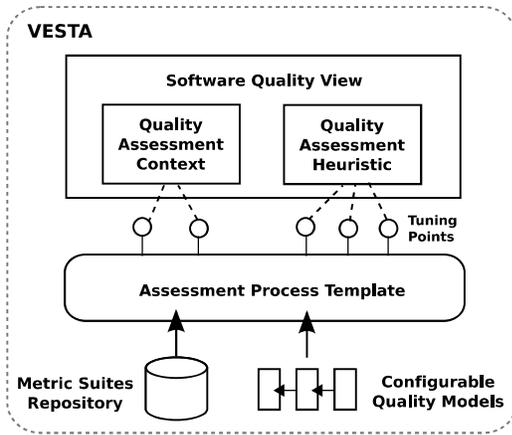


Figure 1. The context diagram of VESTA.

ity assessment in certain “software-specific” levels, such as the whole system, module level, class level, but not the “stakeholder” level.

Mismatches exist between the views of quality assessment tools and the views from stakeholders. The mismatch may result in misunderstanding of quality assessment results, or cause the difficulties in the interpretation of quality assessment results. When stakeholders try to obtain needed quality data for decision making from the measurement results, they are usually confined to the tool views. There are two possible causes to the confinement. One cause is that there is no appropriate assessment process and tool for the stakeholders. To provide the solution, more industry-academia studies are required to invent novel assessment processes and tools. Another cause is that stakeholders have no guidance to utilize and associate existing quality assessment processes and tools to meet their needs. To avoid such shortcomings, the challenge is to associate and adapt existing tools according to stakeholders' views.

For over six years observation on open source software (OSS) project evolution [7], we found that correctly using and associating existing assessment tools can bring extraordinary quality information to stakeholders. We also found there are more and more OSS products built with multiple programming languages. The same product with multiple programming languages can ease stakeholders' effort in software adaptation. The multi-languages view will be



**Figure 2. The internal structure of VESTA.**

needed when stakeholders acquire an OSS product built with multiple programming languages. Unfortunately, current assessment tools cannot be used directly to investigate an OSS product built with multiple programming languages. In this study, a View-based Software Quality Assessment model (VESTA) is proposed to conquer such an issue. In VESTA, the views from stakeholders can be expressed by (1) the quality assessment context and (2) the quality assessment heuristic. Based on the expressed views, the selected metrics, quality models, and quality assessment processes will be associated and configured to a new integral assessment process in VESTA. The integral assessment process will be performed to generate required quality data for stakeholders.

## 2. VESTA

Figure 1 shows the context diagram of VESTA. The inputs of VESTA come from the **Stakeholders** and the **Software Artifacts Repository**. The output of VESTA is the **Software Quality Assessment Report** to the stakeholders according to the given software quality view. To establish the quality assessment context, stakeholders specify the quality assessment context and select quality assessment heuristics in the software quality view. The internal structure of VESTA is shown in Figure 2, which constitutes of four major elements: (1) **Software Quality View**, (2) **Metric Suites Repository**, (3) **Configurable Quality Models**, and (4) **Assessment Process Template**.

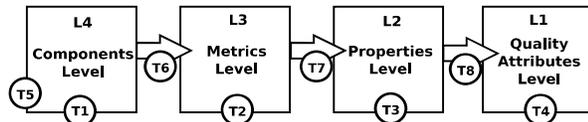
Software quality view includes the quality assessment context and the quality assessment heuristic. The quality assessment context defines the context in which the stakeholders are going to observe software quality evolution. The measurement subjects, granularity, and the weight of diverse measurement subjects will be specified in the quality assessment context. On the other hand, the quality assess-

ment heuristic represents the heuristic configuration of assessment process in the specified quality assessment context. For example, the quality assessment heuristic may include (a) metrics that will be used, (b) quality attributes and models with specific statistics methods, and (c) the metrics measurement results correlated with high-level software qualities. Various software metrics are implemented in the metric suites repository. According to specified software quality views, VESTA will match corresponding metrics from it to perform quality assessment. New metrics could be implemented and added into it to support more software quality assessment tasks. Most commonly used quality models are all implemented in VESTA. Configurable tuning points are identified and exposed during the implementation. Through these tuning points, specific configurations, such as the replacement of metrics and the folk of measurement process, can be made.

The assessment process template defines a series of activities to generate an integral assessment process. First, the quality assessment context provides the context configuration to the assessment process template through tuning points (Figure 2). The tuning points are exposed by the assessment process template to allow configuration from stakeholders' quality views. Second, through the context configuration, the *software artifact processing components* inside the template can be selected and tuned. The responsibility of *software artifact processing components* is to prepare the software artifacts for quality assessment. Third, the quality assessment heuristic also provides the heuristic configurations to the assessment process template. By interpreting the heuristic configuration, metrics and configurable quality models will be selected. Fourth, the selected metrics and quality models will be integrated with *connectors*. The *connectors* are components inside the template, which can control the data flow in integral assessment process. Finally, the integral assessment process will then be generated according to the assessment process template, context configurations and heuristic configurations for quality assessment in VESTA.

### 2.1. Utilization of VESTA under the Multi-Languages View

In this study, the software quality assessment template is instantiated with Bansiya and Davis's hierarchical quality model [8]. The leveling and the identified tuning points are depicted in Figure 3. There are four levels: the component level (L4), the metrics level (L3), the properties level (L2) and the quality attributes level (L1). Figure 4 demonstrates the flow of an assessment process under multi-languages view while assessing the ANTLR open source project (<http://www.antlr.org>). As the developers consider assessing different implementations of ANTLR in Java and



**Figure 3. Selected software quality model and identified tuning points.**

C++ programming languages, the assessment process consists of three fork processes. These three fork processes share the same L4. As shown in Figure 3, T5 represents the selection of target software and its abstract design model. T1 determines the partition of the target software under current software quality view. In the multi-languages view, T5 will be set to ANTLR software and OO design model, and T1 is specified to Java and C++ code partitions. The L4 contains *source code abstraction components* to analyze Java and C++ code into OO design model respectively. The output of L4 is dispatched to three fork processes by a *information flow dispatch connector*.

Each fork process consists of L3, L2 and L1 sequentially. The tuning point T2 in L3 shows the flexibility to select different metrics as needed. Tuning point T6 specifies metrics that are used to measure components. In the multi-languages view, T2 is set to QMOOD metric suite, while T6 specifies the NOP metric that is defined for Java and C++ code. Tuning point T3 in L2 reveals the ability to add other design properties into VESTA according to current software quality view. T7 is the tuning point for the linkage between L3 and L2 to configure metrics that are mapped to design properties. For example, the *Abstraction* design property can be directly mapped by Average Number of Ancestors (ANA) design metric [8]. Tuning point T4 in L1 represents the extensibility on additional quality attributes. The computation equations between quality attributes and properties can be configured by tuning point T8. In the multi-languages view, the configuration of T3, T4, T7 and T8 remain the same as specified in original QMOOD model. Three fork processes are responsible for measuring the Java implementation, C++ implementation, and overall ANTLR product respectively. The measurement results of three fork processes are finally integrated by a *information flow integration connector*, and the integrated quality data is sent to the report generator to form the software quality assessment report.

### 3. Case Study

Figure 5 represents the evolution of QMOOD quality attributes of ANTLR. The QMOOD quality attributes are reusability, understandability, extensibility, functionality,

flexibility and effectiveness. In Figure 5, the horizontal axis represents the version number of the observed software projects and the vertical axis represents the normalized value of each quality attribute. The quality evolution curve of overall assessment is label with “QMOOD” tag. All the quality data in Figure 5 are normalized by the overall assessment of the earliest available version of ANTLR. In ANTLR, two different programming languages are used. C++ is used to implement libraries and interfaces, while Java is used to implement both the interface and main functionality. Figure 5 shows the results of quality assessment from multi-languages view and the original QMOOD assessment. Under QMOOD assessment, the evolution information of whole system qualities are provided, including the downgrade of understandability and upgrade of the other five quality attributes. Developers of ANTLR might take corresponding quality improvement activities, such as understandability improvement. However, the information revealed by QMOOD assessment is not enough.

In this case, the lack of appropriate quality data causes underestimations of extensibility, flexibility, effectiveness of the partition implemented with C++, and also leads to an overestimation of its reusability. For example, the reusability of C++ partition from version 2.4.0 to 2.7.7 evolves from 0.53 to 0.68, while the overall reusability evolves from 1 to 1.49. The reusability of C++ partition is much lower than Java partition. The stakeholders should be aware of this information when they want to acquire the C++ interface and libraries from ANTLR. In addition, the partition implemented with C++ has higher extensibility, flexibility, effectiveness than the partition implemented with Java. For example, the extensibility of Java partition from version 2.4.0 to 2.7.7 evolves from 0.82 to 0.77, while the extensibility of C++ partition from 1.89 to 2.32. This evaluation result suggests that special attention on extensibility, flexibility and effectiveness of Java implementation is needed. The stakeholders who want to acquire Java interface and libraries can therefore negotiate with the developers of ANTLR. The developers of ANTLR can therefore conduct preventive maintenance for Java partition to improve these quality attributes.

### 4. Conclusion

In this study, View-based Software quality Assessment (VESTA) is created to associate and adapt existing tools and processes from the views of stakeholders. VESTA exploits the concept of software quality view, which includes the quality assessment context and heuristic, to model the quality evaluation of software evolution more appropriately. With quality assessment context, contextual factors in the stakeholders’ views can be specified to reflect their concerns to the assessment context. With quality assessment

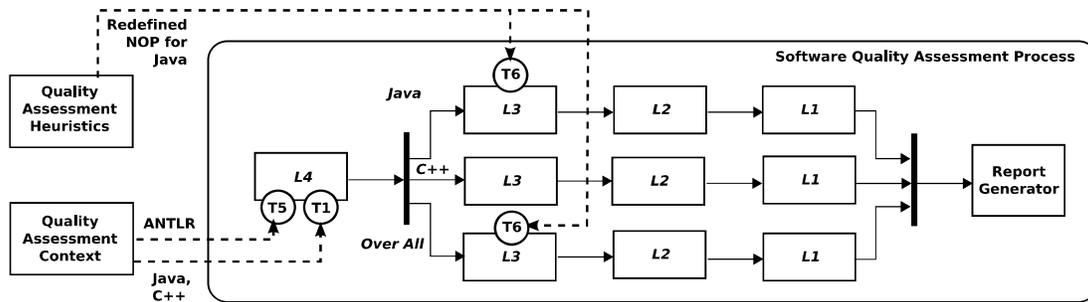


Figure 4. An instantiated software quality assessment process with the multi-languages view.

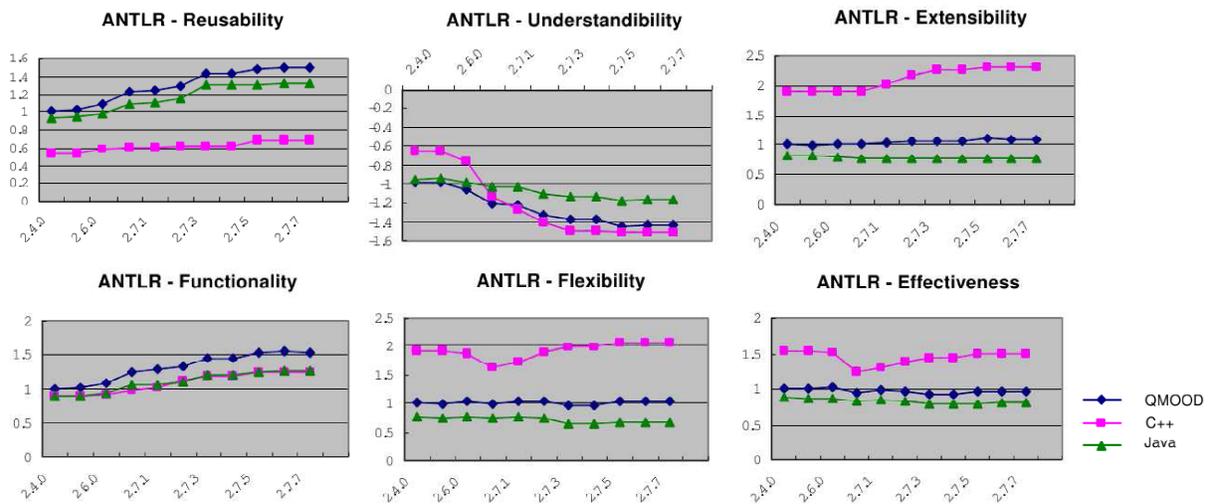


Figure 5. The quality evolution of ANTLR project in the multi-languages view.

heuristic, available quality models and metrics can be selected and adapted to perform the corresponding quality assessment under the specified quality assessment context.

## References

- [1] H. Ogasawara, A. Yamada, and M. Kojo, "Experiences of Software Quality Management Using Metrics through the Life-Cycle," *Proc. 18th Int'l Conf. on Software Engineering*, pp. 179–188, Mar. 1996.
- [2] H. Gall, M. Jazayeri, R. R. Klösch, and G. Trausmuth, "Software Evolution Observations Based on Product Release History," *Proc. 13rd Int'l Conf. on Software Maintenance*, pp. 160–166, Oct. 1997.
- [3] M. W. Godfrey and Q. Tu, "Evolution in Open Source Software: A Case Study," *Proc. 16th Int'l Conf. on Software Maintenance*, pp. 131–142, Oct. 2000.
- [4] X. Franch and J. P. Carvallo, "Using Quality Models in Software Package Selection," *IEEE Software*, vol. 20, no. 1, pp. 34–41, Jan./Feb. 2003.
- [5] G. Robles, J. M. Gonzalez-Barahona, M. Michlmayr, and J. J. Amor, "Mining Large Software Compilations Over Time: Another Perspective of Software Evolution," *Proc. Int'l Workshop on Mining Software Repositories*, pp. 3–9, May 2006.
- [6] F. Deissenboeck, S. Wagner, M. Pizka, S. Teuchert, and J. F. Girard, "An Activity-Based Quality Model for Maintainability," *Proc. 23rd Int'l Conf. on Software Maintenance*, pp. 184–193, Oct. 2007.
- [7] H. C. Jiau and C. H. Kao, "Assessing the Efficacy of User and Developer Activities in Facilitating the Development of OSS Projects," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 21, no. 5, pp. 287–314, Sept./Oct. 2009.
- [8] J. Bansiya and C. G. Davis, "A Hierarchical Model for Object-Oriented Design Quality Assessment," *IEEE Trans. Softw. Eng.*, vol. 28, no. 1, pp. 4–17, Jan. 2002.

# DETECTING EMERGENT BEHAVIOR IN DISTRIBUTED SYSTEMS USING SCENARIO-BASED SPECIFICATIONS

Mohammad Moshirpour

Abdolmajid Mousavi

Behrouz H. Far

Department of Electrical and Computer Engineering  
University of Calgary, Canada  
{mmoshirp, amousavi, far}@ucalgary.ca

## ABSTRACT

Emergent behavior is an important issue in distributed systems' design. Detecting and removing emergent behavior during the design phase will lead to huge savings in deployment costs of such systems. An effective approach for the design of distributed systems is to describe system requirements using scenarios. A scenario, commonly known as a message sequence chart or a sequence diagram, is a temporal sequence of messages sent between system components. However, scenario-based specifications are prone to subtle deficiencies with respect to analysis and validation known as incompleteness and partial description. In this research, a method for detecting emergent behavior of scenario-based specification is proposed. The method is demonstrated and verified using a mine sweeping robot as an example.

**Index Terms** — Distributed systems, Emergent behavior, Message sequence chart.

## 1. INTRODUCTION

Scenario-based specification is one of the approaches to describe a distributed system's behavior. Scenarios are appealing because they allow stakeholders to describe system's functionality by partial stories. There are two main methods for representing scenarios, developed by OMG and ITU, namely, Sequence Diagram (SD) and Message Sequence Chart (MSC) [1, 2]. In spite of the advantages of using scenarios such as expressive power and simplicity, there are several challenges particularly for concurrent systems consisting of multiple autonomous agents (MAS) as well as distributed systems (DS) which consist of multiple system components. For instance, because each scenario gives only a local and partial story of a distributed system's behavior, the challenge is how the behavior of a distributed system can be constructed from a set of scenarios and more importantly whether the derived behavior is acceptable or not.

The model which describes the behavior of each system element (i.e. agent, component or processes) is called *behavioral model*, and the procedure of building the behavioral models for the elements from a scenario-based specification, is called *synthesis of behavioral models*, or simply, *synthesis process*. A widely accepted model for behavioral modeling of individual system elements is the state machine. Several studies have already been conducted to facilitate the procedure of converting a set of scenarios to a behavioral model expressed by state machines [5, 6, 8, 12, 13]. In the synthesis process, one state machine will be built for each system element. The state machine includes all the messages that are received or sent by that element. Then the behavior of the distributed system is described by the product (parallel execution) of all the state machines of the system elements.

One of the challenges during the synthesis process, is *implied scenarios* [3, 4, 10, 11], also known as *emergent behavior*. An implied scenario is a specification of behavior that is in synthesized model of the distributed system and is not explicitly specified in its specification as a scenario. This is best illustrated using an example of a real-life distributed system.

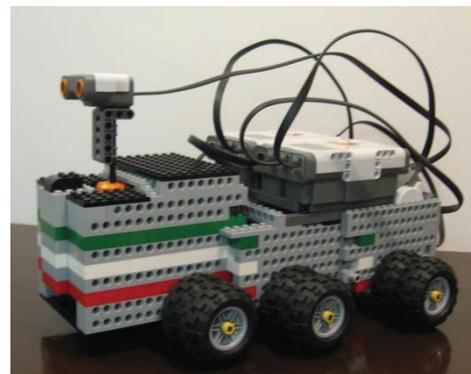


Figure 1 – Prototype of a mine sweeping robot

Let's consider the prototype of an automated mine-sweeping robot shown in Figure 1. The robot's mission is to navigate through a maze-like course, which resembles the

layout of the streets of a city, for which the robot has no map and has to investigate it based on utilizing its sensory information, i.e. ultrasonic and/or GPS data. At the same time, it has to identify and mark the location of mines. For this prototype version it is assumed that mines emit infrared signal which is detectable via the infrared sensor. In order to provide the robot with more computation power and additional control for the motors and different types of sensors, two multi-core CPU units are utilized. The units are built on separate boards connected via a simple but reliable connection protocol. The two CPUs interact using the client-server architecture. As there is no sophisticated operating system in charge of the control and scheduling of the processes and threads, the design of the robot must account for the proper management of all processes and their interactions in a logical and efficient manner.

For the sake of simplicity, let's assume that the robot has only two sets of sensors: an ultra-sonic which is used for navigation purposes, and an infrared sensor to detect mines. Both of these sensors are connected to the client CPU. Thus the client receives signals from sensors, processes the message and sends them to the server CPU to act upon them accordingly. The processes *Client Controller* and *Server Controller*, depicted in Figure 2, are in charge of the motors responsible for the wheels on the left and right sides of the robot. Each process is also responsible for sending and receiving messages to and from the other. The server controller is also in charge of the motor of the mechanism which dispenses a flag on the location in which a mine has been detected. The *ULTS Motor Controller* process is responsible for the motor in charge of the rotation of the ultra-sound sensor which is necessary for optimum navigation through the maze.

Partial behavioral scenarios for this robot are represented by message sequence charts. The message sequence chart 1 (MSC1), shown in Figure 2, illustrates a scenario where the robot is moving forward with no obstacle on its way. MSC2 (Figure 3), shows a scenario where the robot has been halted due to the detection of an obstacle in its path while MSC3 (Figure 4) illustrates a scenario where the robot stops because of the detection of a mine (based on the signal received from the IR sensor) which is a pre-requisite for the mine-flagging operation.

As it can be seen from MSC2 and MSC3 there are two events which cause the robot to halt: (1) detecting an obstacle on the way performed by the ultrasound sensor; and (2) detecting a mine which is done by the infrared sensor. Similarly there are two events which trigger the motion of the robot: (1) detection of a free path (i.e. no obstacles in the way) by the ultrasound sensor; and (2) the completion of the mine-flagging operation. MSC4 shown in Figure 5 illustrates an emergent behavior that might occur as the result.

An important observation to be made is the generalization of messages to indicate their purpose rather

than their specific implementation. Consider message "send signal (some signal)" which is sent from either of the sensors to the client controller process as shown in each of the MSCs (Figures 2-5).

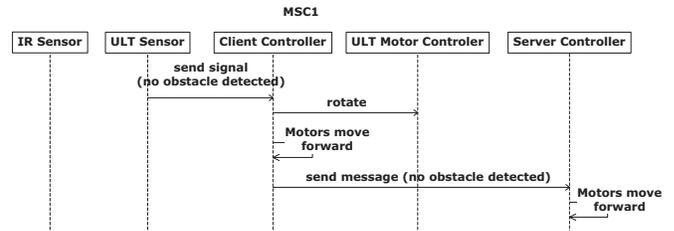


Figure 2 - Robot is moving forward with no obstacle in the way

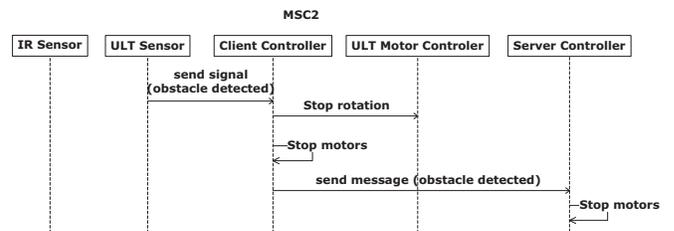


Figure 3 - Robot is halted due to the detection of obstacle in its path

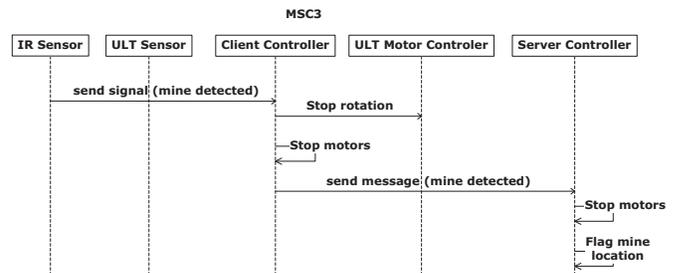


Figure 4 - Robot stops due to the detection of a mine

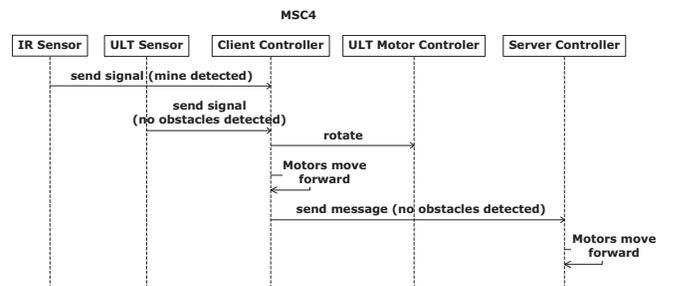


Figure 5 - Client controller receives a no obstacle detection message from the ultra-sound sensor after receiving the mine detection message from the IR sensor which results in missing the mine

For instance in MSC1 (Figure 2) the content of the message sent from the sensor to the client controller is "no obstacles detected" while in MSC3 (Figure 4) the content of

the message is "mine detected". It is important to note that although the content of these messages are different, the purpose of the two messages remains the same. That is, the client controller process expects to receive a message, regardless of the content of the message, from a sensor. Therefore the message sent from either sensor to the client controller process is "send signal" and the content is included in brackets only for clarity.

In [4] an algorithm for safe realizability is proposed but there is no method to directly check whether implied scenarios exist in the first place. In other words, so far there is no formalized and precise representation for the cause of implied scenarios, so that by capturing that cause during the design phase, emergent behavior can be detected and removed. A main contribution of this paper is to give a definition for indeterminism in behavior of distributed systems, so that by identifying indeterminism, one can detect the potential emergent behavior.

The structure of the rest of this paper is as follows: in Section 2 several definitions that are needed for the rest of the paper are presented. In Section 3 system behavior modeling is examined and in Section 4 the detection of indeterminism and emergent behavior in distributed systems is discussed. Conclusions and future works are presented in Section 5.

## 2. DEFINITIONS

In this section, we give some definitions related to MSC notation based on a subset of ITU definitions for MSC [1, 4, 7].

Let  $P$  be a finite set of processes in a distributed system (with the total number of processes  $p \geq 2$ ) and  $C$  be a finite set of message contents (or message labels) that are passed among the processes. Let  $\Sigma_i = \{i!j(c), i?j(c) \mid j \in P \setminus \{i\}, c \in C\}$  be the set of alphabet (i.e. events) for the process  $i \in P$ , where  $i!j(c)$  denotes an event that sends a message from process  $i$  with content  $c$  to process  $j$ , whereas  $i?j(c)$  denotes an event that is received by process  $i$  a message with content  $c$  from process  $j$ . The set of alphabet will be  $\Sigma = \bigcup_{i \in P} \Sigma_i$  and each member of  $\Sigma$  is called a message.

In the following, we try to capture a causal relationship between a message and its predecessors by defining partial Message Sequence Chart (pMSC).

**Definition 1** (partial Message Sequence Chart): A partial Message Sequence Chart (pMSC) over  $P$  and  $C$  is defined to be a tuple  $m = (E, \alpha, \beta, <)$  where:

$E$  is a finite set of events.

$\alpha: E \rightarrow \Sigma$  maps each event with its label. The set of events located on process  $i$  is  $E_i = \alpha^{-1}(\Sigma_i)$ . The set of all send events in the event set  $E$  is denoted by  $E! = \{e \in E \mid \exists i, j \in P, c \in C : \alpha(e) = i!j(c)\}$  and the set of receive events as  $E? = E \setminus E!$ .

$\beta: F! \rightarrow E?$  is a bijection mapping between send and receive events such that whenever  $\beta(e_1) = e_2$  and  $\alpha(e_1) = i!j(c)$ , then  $\alpha(e_2) = j?i(c)$ .

$<$  is a partial order on  $E$  such that for every process  $i \in P$ , the result of  $<$  on  $E_i$  is a total order of its members and the transitive closure of  $\{(e_1, e_2) \mid e_1 < e_2, \exists i \in P: e_1, e_2 \in E_i\} \cup \{(e, \beta(e)) \mid e \in E\}$  is a partial order of the members of  $E$ .

The partial order  $<$  captures casual relationship between the events of a pMSC. This causality basically represents two things. First, a receive event cannot happen without having its corresponding send event happened before. Second, a receive (or send) event, cannot happen until all the previous events, which are causal predecessors of it have already been accomplished. Obviously, if all the send events have their corresponding receive events (i.e. as defined by the function  $\beta$ ), the structure is called a Message Sequence Chart or simply an MSC. In other words, an MSC has the same structural components as a pMSC, except that  $\beta$  is defined for  $F! = E!$ .

**Definition 2** (projection): The projection  $m|_i$  for process  $i$  in MSC  $m$ , is the ordered sequence of messages corresponding to the events for the process  $i$  in the pMSC  $m$ . For  $m|_i$ ,  $\|m|_i\|$  indicates its length, which is equal to the total number of events of  $m$  for the process  $i$ , and  $m|_i[j]$  refers to  $j^{\text{th}}$  element of  $m|_i$ , so that if  $e_j$  is the  $j^{\text{th}}$  interaction event for process  $i$  according to the total order of the events of  $i$  in  $m$ , then  $\alpha_m(e_j) = m|_i[j - 1]$ ,  $0 < j < \|m|_i\|$ . In  $m|_i$ , we call every element  $i!j(c)$ ,  $i, j \in P, c \in C$ , a send message and every element  $i?j(c)$ , a receive message.

For example, the projection for the process client controller in MSC1 in Figure 2 will be "client controller!server controller(send message)".

**Definition 3** (Equivalent Finite State Machine for a projection): For the projection  $m|_i$ , we define the corresponding deterministic finite state machine  $A_i^m = (S^m, \Sigma^m, \delta^m, q_0^m, q_f^m)$  such that:

$S^m$  is a finite set of states labelled by  $q_0^m$  to  $q_{\|m|_i\|}^m$ .

$\Sigma^m$  is the set of alphabet

$q_0^m$  is the initial state

$q_f^m = q_{\|m|_i\|}^m$  is the final state (accepting state)

$\delta^m$  is the transition function for  $A_i^m$  such that  $\delta(q_j^m, m|_i[j]) = q_{j+1}^m, 0 \leq j \leq \|m|_i\| - 1$ . Thus the only word accepted by  $A_i^m$  is  $m|_i$ .

Note that scenarios can be treated as *words* in a formal language, which is defined over send and receive events in MSCs. Then, a *well-formed word* for a process is one that for every receive event there exists a send event in that word, which in fact captures the essence of definition given for a pMSC (Definition 1). On the other hand, a *complete word* for a process is the one that for every send event in it, its corresponding receive event also exists in it. In practice,

a system designer must look for complete and well-formed words for each process, which is not necessarily an easy task. For any MSC  $m$  in the set of MSCs  $M$ , any sequence  $\omega$  of  $m$ , obtained from a sequence of events in  $m$  that respects the partial order of the events defined for  $m$ , is called a linearization of  $m$ , and is a word in the language  $L(M)$  of  $M$ .

### 3. SYSTEM BEHAVIOR MODELING

Scenario-based specification is an efficient and effective way to represent system requirements for distributed systems. However as each scenario only partially describes system's behavior, scenario based specifications are subject to deficiencies such as incompleteness and contradictions. Thus having a methodology which can systematically discover system design errors prior to implementation is most beneficial and will lead to huge savings in time and cost. In this section, the first part of this systematic approach, which is the synthesis of state machines from message sequence charts is described and elaborated using the example of the mine sweeping robot.

The procedure of construction of finite state machines (FSMs) from message sequence charts (MSCs) is referred to as behavior modeling. For any process  $i$  of a partial MSC described in Definition 1, an equivalent finite state machine (Definition 3) can be constructed. For instance, Figure 6 shows the eFSM constructed for the client controller process in MSC1.

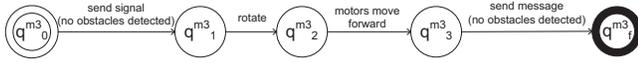


Figure 6 - eFSM for the client controller process in MSC 1

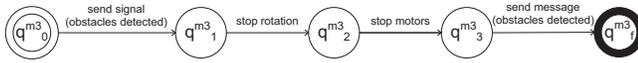


Figure 7 - eFSM for the client controller process in MSC 2

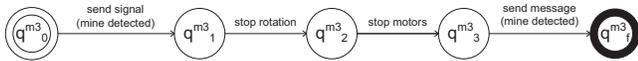


Figure 8 - eFSM for the client controller process in MSC 3

A comprehensive definition for identical states is needed for synthesis of behavior models from scenarios. To achieve this we must first have a clear procedure to assign values to the states of the eFSMs. This is a very important step and is performed differently in various works. For instance, [13] proposes the assignment of global variables to the states of eFSMs by the system designed engineer, referred to as the domain expert in this research. However the outcome of this approach is not always consistent as the global variables chosen by different domain experts would vary. Therefore to achieve consistency in assigning state values, the

approach of [14] which is making use of an invariant property of the system called semantic causality is followed.

**Definition 4** (Semantic causality): A message  $m|i[j]$  is a semantical cause for message  $m|i[k]$  and is denoted by  $m|i[j] \xrightarrow{se} m|i[k]$ , if process  $i$  has to keep the result of the operation of  $m|i[j]$  in order to perform  $m|i[k]$ .

For example, in MSC1 in Figure 2, message "send signal" is a semantic cause for message "rotate". As semantic causality is an invariant property of the system and is part of the system's architecture and the domain knowledge, it is independent of the choices made by the domain experts. In other words, we let the current state of the process to be defined by the messages that the process needs in order to perform the messages that come after its current states. Thus using semantic causality we proceed to build the system's domain theory as defined by Definition 5.

**Definition 5** (Domain theory): The domain theory  $D_i$  for a set of MSCs  $M$  and process  $i \in P$  is defined such that for all  $m \in M$ , if  $m|i[j] \xrightarrow{se} m|i[k]$  then  $(m|i[j], m|i[k]) \in D_i$ .

Following the example above, since the message "send signal" is a semantic cause for message "rotate", both messages are part of the domain theory. Upon building the domain theory based on semantic causality we proceed to assign state values to the states of the constructed eFSM as explained in Defection 6.

**Definition 6** (State value): The state value  $v_i|(q_k^m)$  for the state  $q_k^m$  in eFSM  $A_i^m = (S^m, \Sigma^m, \delta^m, q_0^m, q_f^m)$  is a word over the alphabet  $\Sigma_i \cup \{1\}$  such that  $v_i|(q_f^m) = m|i[f - 1]$ , and for  $0 < k < f$  is defined as follows:

- i)  $v_i|(q_k^m) = m|i[k - 1]v_i|(q_j^m)$ , if there exist some  $j$  and  $l$  such that  $j$  is the maximum index that  $m|i[j - 1] \xrightarrow{se} m|i[l]$ ,  $0 < j < k$ ,  $k \leq l < f$
- ii)  $v_i|(q_k^m) = m|i[k - 1]$  if case i) does not hold but  $m|i[k - 1] \xrightarrow{se} m|i[l]$ , for some  $k \leq l < f$
- iii)  $v_i|(q_k^m) = 1$ , if none of the above cases hold

For instance in order to calculate the state value for state  $q_2^{m1}$  we proceed as follows: from the domain theory of the system (Definition 5) we learn that the maximum index  $j$  for which  $m1|_{client\ controller}[j - 1]$  is a semantical cause for a message in the transitions after  $q_2^{m1}$  is  $j = 1$  for which  $m1|_{client\ controller}[j - 1] = send\ signal$ . That is to say that for example the message "send signal" is a semantic cause for message "motors move forward". Therefore from case (i) of Definition 6 we obtain:  $v_{client\ controller}|(q_2^{m1}) = m1|_{client\ controller}[2 - 1]v_{client\ controller}|(q_1^{m1})$ .

In order to calculate  $v_{client\ controller}|(q_1^{m1})$  we observe that "send signal" is the only semantic cause after  $q_1^{m1}$ , thus case (ii) of Definition 6 holds and we get  $v_{client\ controller}|(q_1^{m1}) = send\ signal$ . Therefore we have  $v_{client\ controller}|(q_2^{m1}) = (rotate)(send\ signal)$ . By

following the same approach we get the state value for  $q_2^{m3}$  to be  $v_{client\ controller}(q_2^{m3}) = (rotate)(send\ signal)$ .

From these examples it can clearly be seen that semantic causality is an invariant property of the system and is not affected by the preferences of the domain expert.

To complete behavior modeling for the system, for each process a final FSM which the union of its corresponding eFSMs from different scenarios is to be built. Figure 9 demonstrates the union of the three eFSMs built from MSCs 1-3. As established in Definition 6 the value of start and end states are defined to be equal to 1. The values for other states are calculated and shown in Table 1.

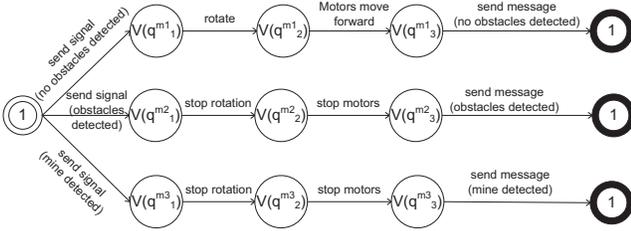


Figure 9 - The union of the eFSMs built from MSCs 1-3

Table 1 - State values for the states of the resulted eFSMs

States	State Values
$v_{client\ controller}(q_1^{m1})$	(send signal)
$v_{client\ controller}(q_2^{m1})$	(rotate)(send signal)
$v_{client\ controller}(q_3^{m1})$	(motors move forward)(send signal)
$v_{client\ controller}(q_1^{m2})$	(send signal)
$v_{client\ controller}(q_2^{m2})$	(stop rotation)(send signal)
$v_{client\ controller}(q_3^{m2})$	(stop motors)(send signal)
$v_{client\ controller}(q_1^{m3})$	(send signal)
$v_{client\ controller}(q_2^{m3})$	(stop rotation)(send signal)
$v_{client\ controller}(q_3^{m3})$	(stop motors)(send signal)

#### 4. DETECTION OF INDETERMINISM AND EMERGENT BEHAVIOR

As demonstrated in the previous section, by assigning state values based on semantic causality, the basis for comparing states and consequently discovering identical states is established. Identical states are defined in Definition 7 as follows.

**Definition 7** (Identical states): Two states  $q_j^m$  and  $q_k^n$  of process  $i$ , ( $m$  and  $n$  could be the same) are identical if one of the following holds:

- i)  $j = k$  for  $0 \leq t < j: m_i[t] = n_i[t]$
- ii)  $v_i(q_j^m) = v_i(q_k^n)$

By considering the union of the eFSMs demonstrated in Figure 9 and the state values presented in Table 1 the

identical states that correspond to case (ii) of Definition 7 are determined. As identical states are possible areas in which the system might get confused over what course of action to take, these states are recorded and presented to the domain expert to be analyzed and reconsidered.

Figure 10 demonstrates the manner by which identical states in the mine sweeping robot example can result in emergent behavior. As mentioned earlier, the message "send signal" sent from either sensor to the "client controller", should be considered as a transition regardless of the content of that message since the client controller process is waiting to take any message that is sent from the sensors. This causes  $q_1^{m1}, q_1^{m2}, q_1^{m3}$  to have identical state values and satisfy case (ii) of Definition 7.

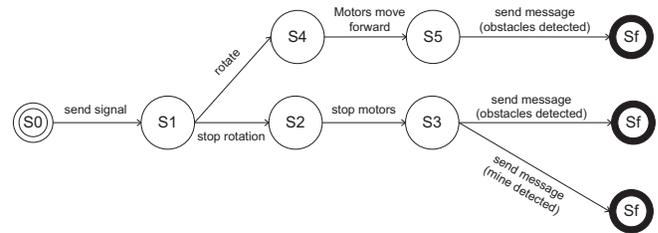


Figure 10 - Resulted DFA after merging identical states

However the content of the message "send signal" will make a difference in the behavior of the robot. Therefore these identical states may result in emergent behavior. As shown in Figure 10 by having the states  $q_1^{m1}, q_1^{m3}$  as identical states the robot gets stuck in a state of confusion between moving forward as it detects no obstacles in its path, and setting a flag that a mine is detected by infrared sensor. This state of deadlock is illustrated by MSC4, in Figure 5.

#### 5. CONCLUSIONS AND FUTURE WORK

Detection of failures and removal of faults during field use of a system is about 20 times more expensive than detection and removal in the requirement and design phase [15]. Some of the failures can be attributed to distributed system design. A goal in this research is to identify possible design flaws that might lead to run time problems in distributed systems by analyzing the system specification expressed by scenarios. Unfortunately, manual review may not efficiently detect all the design flaws due to the scale and complexity of the system. In this research we have provided sound techniques to automate the specification and design review of the distributed system and detect a subset of unwanted run time behaviors, including implied behaviors. These issues have been illustrated using the example of a mine sweeping robot which is designed and implemented as a distributed system.

In this paper, we provided a method to identify the exact cause of implied scenarios, so that by capturing it, implied scenarios can be detected and removed. This method is novel in the sense of formalization of the cause of implied scenarios. We believe that this is the main reason for some shortcomings and conflicts in the current works, as they have been revealed in [9] and [14]. One of our contributions is to show that the concept of indeterminism, which was demonstrated in this paper, is able to address and resolve the problems that are mentioned in [9].

For future work automating the process of resolving the detected emergent behavior is intended. In addition the proposed methodology can be extended to a comprehensive framework for model based analysis and testing of distributed software systems. Furthermore, this technique can be modified to take the UML's sequence diagrams as input and thus incorporate the analysis and design of distributed object oriented systems. These techniques can also be extended to verify the design of multi-agent systems.

## REFERENCES

- [1] Recommendation Z.120: Message Sequence Chart(MSC), Geneva, 1996.
- [2] Unified modeling language specification, version 2. Available from rational software corporation, Cupertino, CA, 2006.
- [3] B. Adsul, M. Mukund, K. N. Kumar, and V. Narayanan. Causal closure for msc languages. In FSTTCS 2006, pp. 335–347. LNCS 3821, 2005.
- [4] R. Alur, K. Etessami, and M. Yannakakis. Inference of message sequence charts. *IEEE Transaction on Software Engineering*, 29(7):623–633, July 2003.
- [5] D. Harel and H. Kugler. Synthesizing state-based object systems from lsc specifications. *International Journal of Foundations of Computer Science*, 13(1):5–51, 2002.
- [6] I. Kruger, R. Grosu, P. Scholz, and M. Broy. From mscs to statecharts, in: Franz j. rammig (ed.): *Distributed and parallel embedded systems*. Kluwer Academic Publishers, 1999.
- [7] M. Lohrey. Safe realizability of high-level message sequence charts. In *CONCUR 2002*, pages 177–192. LNCS 2421, 2002.
- [8] E. Makinen and T. Systa. MAS - an interactive synthesizer to support behavioral modeling in UML. In *ICSE 2001*, pages 15–24, Toronto, May 2001.
- [9] A. J. Mooij, N. Goga, and J. M. T. Romijn. Non-local choice and beyond: Intercacies of MSC choice nodes. In M. Cerioli (ed): *FASE 2005*, LNCS 3442, pages 273–288. Springer, 2005.
- [10] H. Muccini. Detecting implied scenarios analyzing nonlocal branching choices. In *FASE 2003*, Warsaw, Poland, April 2003.
- [11] S. Uchitel, J. Kramer, and J. Magee. Negative scenarios for implied scenario elicitation. In *10th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE'02)*, Charleston, November 2002.
- [12] S. Uchitel, J. Kramer, and J. Magee. Synthesis of behavioral models from scenarios. *IEEE Transaction on Software Engineering*, 29(2):99–115, February 2003.
- [13] J. Whittle and J. Schumann. Generating statecharts designs from scenarios. In *ICSE 2000*, Limerick, Ireland, 2000.
- [14] A. Mousavi and B. Far, "Eliciting Scenarios from Scenarios", In *Proceedings of 20<sup>th</sup> International Conference on Software Engineering and Knowledge Engineering (SEKE 2008)*, San Francisco Bay, USA, July 1-3, 2008.
- [15] Dennis R. Goldenson, Diane L. Gibson, "Demonstrating the Impact and Benefits of CMMI: An Update and Preliminary Results," *CMU/SEI-2003-SR-009*, October 2003.

# MMWA-ae: boosting knowledge from Multimodal Interface Design, Reuse and Usability Evaluation

Americo Talarico Neto, Renata Pontin M. Fortes, Rafael Rossi and Solange Rezende  
Av. Trabalhador Saocarlene,400. São Carlos, SP, Brazil

University of São Paulo

E-mail: {americo@, renata@, solange@, ragero@}icmc.usp.br

## Abstract

*The technological progress designing new devices and the scientific growth in the field of Human-Computer Interaction are enabling new interaction modalities to move from research to commercial products. However, developing multimodal interfaces is still a difficult task due to the lack of tools that consider not only code generation, but usability of such interfaces. In this paper, we present the MultiModal Web Approach and its authoring environment, whose main goal is boosting the dissemination of project knowledge for future developments benefited by the solutions to recurring problems in that multimodal context.*

## 1. Introduction

The support for various interaction modalities has become a mandatory requirement for the next generation interfaces with the growing proliferation of computing devices, the increasing availability of Web services to the worldwide population and due to the great expressive power, naturalness and portability that multimodal interfaces offer the users to perform their daily tasks [9]. However, the use of combined modalities such as voice, touch and graph, raises a number of usability and interaction issues that the project team is faced with, like synchronization and integration requirements

and constraints that should be considered during the design phases[5].

As a result, the designers are exposed to an ever-increasing challenge: despite of learning the novel languages and accessible technologies and applying them to obtain the multimodal application code, they should realize which are the best practices in this research field and how to experiment the designed interfaces with real users with different behaviors in a fast manner prior to the product release. In addition, the project team should be concerned on how to promote reuse of Design Rationale (DR) and application code in order to decrease the cost and effort implementing multimodal interfaces.

In this paper, we endeavor to answer these questions and concentrate on solving such issues by presenting the MultiModal Web Approach (MMWA), which relies on the theoretical and empirical expertise acquired in the design process. Such expertise is documented in the form of Design Rationale (DR), Design Principles and Design Patterns which can be shared and applied by the design team, during multiple iterations to refine the interface or by designers of different applications.

Moreover, based on our experience applying the MMWA in real projects [8] and the observations of the design workflow of MMWA, we have designed an authoring tool that guides the project team, through the MMWA steps and activities. The advantage of this authoring environment, MMWA-ae, is that it suggests design alternatives based on

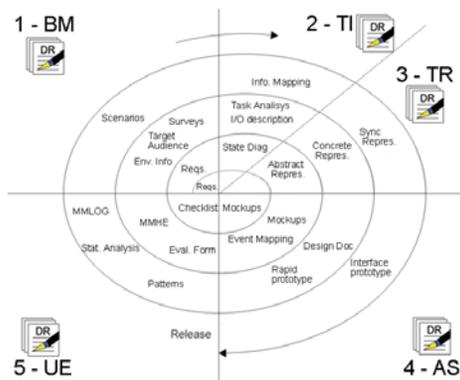


Figure 1. MMWA design process

previous collected DR [8] and the well-known interaction Design Patterns [2] extended to support the multimodal theory and it also implements the previous identified principles and checklists [8] so that the designers are presented with the successful solutions for recurring problems in this context together with their rationale.

This paper is structured as follows: Section 2 details the MMWA. Section 3 describes the case studies that illustrate how MMWA-ae was applied, sharing the most significant results obtained. Section 4 presents a bibliographic review of design methods for multimodal interfaces and their main issues. Conclusions and future work are discussed in Section 5.

## 2. The MultiModal Web Approach and its authoring environment

The MultiModal Web Approach (MMWA) [8], Figure 1, provides designers with a more practical framework to design, develop and evaluate multimodal projects. It follows a spiral process composed of 5 activities that allow the capture and retrieval of DR, which provides successful solutions for recurring problems in the multimodal interaction context, decisions, justifications, alternatives and arguments that led to the final design, documenting them for future reuse. As a result, it optimizes the methodology for developing interfaces

and enables the identification, analysis and discovery of multimodal interaction Design Patterns.

**Behavioral Model (BM)** consists of scenarios, constraints and information on the environment in which the multimodal tasks will be performed. BM aims to assist in identifying the information exchanged between the user and the system during the task execution. The goal is getting the task domain information that is necessary for selecting the appropriate tasks mapping and their multimodal interface representation. All such information and data are documented as part of the multimodal application Requirements Specification.

**Task Identification (TI) and Representation (TR)** identify the tasks to be performed by the user analyzing important task aspects such as: goals, initial states, activities and procedures involved, the problems that may occur, the task environment, the target audience and their experience, and the multimodal input and output. It creates abstract representations employing methods such as those described in [10] or [12]. Elements needed to generate and analyze any kind of multimodal input and output including their pros and cons are documented in a DR form. The idea is representing the result as a list of interfaces elements for each candidate input and output modality and modalities combinations. Concrete representations are used to assign GUI and VUI widgets and the methods required to trigger the transitions as well as the events that can be activated in each interface element.

**Analysis of Solutions (AS)** uses earlier captured DR and filters out the interaction options based on specialists argumentations. It provides the potential modalities to perform the task with their DR. At the end, a design document is obtained and it is evaluated in the next activity.

An elementary step in the development process for any multimodal interface involves **usability evaluation (UE)** of prototypes based on qualitative studies; in MMWA it is represented as the Heuristic Evaluation mechanism (**MMHE**), and authentic user interactions. A main feature of the MMWA is the web-based usability evaluation mechanism that is intended to provide data for problems identification, analysis and correction in early stages of the

project; in MMWA it is represented as a mechanism for automatic logging generation and analysis in remote User Tests (**MMLOG**). A benefit of web-based usability studies is that users can be recruited from all over the world and can interact whenever is most convenient.

The **MMWA authoring environment (MMWA-ae)** was developed to automate the MMWA activities and serve as a framework for developing multimodal Web interfaces using voice, touch and graph. With the aid of MMWA-ae the designer specifies the multimodal interaction using Task Analysis as a foundation. It is possible to make use of Design Patterns and DR to resolve design questions and obtain the rationale to solve recurrent problems with the help of solutions used in previous projects in the same context.

The use of DR can be time-consuming, disruptive and costs excessive. To solve this constraint our approach extracts and captures DR information when it is needed. The MMWA-ae holds an abstract mechanism for capturing and storing DR allowing organizations and developers to use any WIKI systems main features to collaborate in the tasks specification, implementation and during tests activities. This mechanism also makes information organization achievable, so that future projects would benefit from efficient searches, ensuring that precise results are achieved.

Another important feature of MMWA-ae is the ability to generate multimodal interface prototype to be target of usability evaluations. For each task specified by the designer, the tool creates a code snippet that contains voice and graphical interface elements and includes functions to capture interaction events. It is possible to perform remote user testing deploying this interface in a Web environment, therefore evaluating the usability of the designed interface looking for problems that can be corrected before release to a final customer. This functionality also enables the user interaction patterns identification, by analysing charts and graphs, and design errors identification or areas of usability improvement. In addition it covers the main limitations of user testing, which are: the difficulty of recruiting users, time and resources availability to

perform the tests, since the whole process is done remotely.

Finally, MMWA automatically creates the design documentation, given that during the Task Analysis the designer is aided by a wizard, whose main goals are: to perform the analysis of solutions; to obtain the key usability principles to be included in the generated code and to ensure that the checklist available on MMWA is considered in the prototype generation. As future work we intend to include in MMWA a mechanism for the fusion of pre-existing graphical user interfaces with speech interfaces. Thus, the designer would spend less effort in the checklists verification and using the practices compiled in the form of DR during the process of specifying and creating multimodal Web interfaces.

### 3. Overview of Case Studies & Results

Up to now we have performed 3 case studies using the MMWA that permitted us to exhibit expressive results obtained using the approach, including: the validation of the MMWA cycle and its activities; the controlled mechanism to capture and store DR; and the MMHE and MMLOG mechanisms that allow for a better interaction design, reporting the usability problems in a DR format and the underlying rationale to fix them. A new case study was conducted in order to obtain a number of results on the topic of the authoring tool. The objectives were to determine quality metrics, to analyze the reutilization rate and to measure the usability of the generated source code. The same design used in a previous case study, a Car Rental System was used. The artifacts were obtained by two groups of designers: one following the MMWA approach supported by the MMWA-ae and the other group designing, developing and evaluating the interfaces without the support of the authoring tool.

In order to be able to measure **code reutilization** and to confirm that the **reuse** was increased by the use of MMWA-ae we employed the formula:

$$\text{ReutilizationRate} = [LOC_{reused} + (LOC_{WhiteBox}) - (LOC_{NonDesiredMethods})] / LOC_{total}$$

Applying the reutilization rate formula in the

case study we ended up with 83% of overall reutilization.

Moreover, the **time spent to design the interface and to obtain the prototype** for user testing is one of the main advantages of the MMWA-ae, given that the interfaces prototype is obtained right after the design phase is completed. In contrast, when MMWA-ae was not used a large numbers of tasks were performed before the prototype was obtained, like: DR database queries, team discussions to validate implementation, rounds of usability analysis prior to the final delivery, voice user interfaces grammar creation and testing, etc.

MMWA-ae offers the designer a Wizard to fill all the important prompts to be used in each voice interface. Furthermore by choosing one of the Design Patterns, for example the Calendar pattern, the tool automatically creates the graphical and the voice interfaces, the grammar and synchronizes all of them together saving a huge amount of the development time. It also increases the usability of such interfaces because MMWA-ae includes an error recovery strategy mechanism and the well-know voice universal commands, allowing for a more robust multimodal interface [1].

MMWA-ae also promotes the **Design Documentation and Rationale storage and retrieval**. DR recorded in the case studies is intended to be a framework of the reasons behind decisions made when designing artifacts. The understanding of the justification for design decisions made throughout the design process is necessary in order to understand, recreate, reuse or modify the design.

As a result the MMWA-ae enables the use of DR: to verify the design decisions, whether it truly reflected what the designers planned based on the MMWA Behavior Model activity; to evaluate the design alternatives (MMWA Analysis of Solutions activity); to modify the design during the maintenance phases and to promote the Design Reuse; to promote Design Communication among people who are involved in the design process; to document the entire design process.

Another significant outcome obtained was the framework for Design Patterns identification. Based on the DR collected in various projects, we

have developed a mechanism that helps on building a Design Pattern based on repeated issues, recurring positions supported by well-built arguments.

Another important finding is regarding the usability of the MMWA-ae generated interfaces. By performing Heuristic Evaluations we were able to identify that most of the usability issues found during the first case study were not present in the user interface created by MMWA-ae. This could be achievable due to the MMWA-ae inherent DR knowledge and due to the Design Pattern suggestion mechanism, which allows the designers to choose the best design pattern to implement their ideas. Furthermore the usability principle built in the generated code also contributes to the overall usability increase. Comparing data from the three case studies, problems found applying the MMHE decreased from one case study to another due to the use of DR in subsequent projects and due to the use of Heuristic Evaluation prior to the User Tests, validating assumptions made in [6] regarding the complementary nature of those usability evaluation methods.

We also used **Association Rules** in our case studies. An association rule is rule of the type  $A \Rightarrow B$  in which A and B are itemsets on the database, and  $A \cup B = \emptyset$ . The two classical measures to generate association rules are support and confidence. **Support** measures the joint probability of an items set in a database, that is,  $sup(A \Rightarrow B) = n(A \cup B)/N$ , in which  $n(A \cup B)$  is the number of transactions that A and B occur together, and N is the total number of transactions. **Confidence** indicates the occurrence probability of A and B given the occurrence of A, that is,  $conf(A \Rightarrow B) = n(A \cup B)/n(B)$ .

The users interaction with the prototyped interface generated by MMWA-ae generates log files containing the users actions and system events, one containing all the interactions in a row of the database, called BASE-1; and another file containing where a single interaction with an interface element is represented in a row of the database, called BASE-2. There were 70 transactions in the BASE-1 and 1,190 transactions in the BASE-2. Using association rules and extracting modalities and tasks

relationships like occurrence and confidence, we intend to verify which modalities the users choose to perform a task in the system and in which tasks or moments they use a combination of modalities to interact. Besides that, we want to verify the most common errors, in which part of the interface they occur and how the modalities are used to recover from an error.

Some of the extracted rules using the BASE-1 are shown in Table 1. It can be noticed that in most of the interactions the user chooses both voice and graphical models. In 74.6% of the times multimodality was used, in which the use of voice implies the use of graphical interface with 86.9% of confidence, and the use of graphical interface implies the use of voice with 85.5% of confidence.

Rule	Support	Confidence
$\emptyset \Rightarrow$ VOICE	85.9	85.9
$\emptyset \Rightarrow$ GRAPHICAL	87.3	87.3
VOICE $\Rightarrow$ GRAPHICAL	74.6	86.9
GRAPHICAL $\Rightarrow$ VOICE	74.6	85.5
VXML_ERROR $\Rightarrow$ GRAPHICAL	50.7	92.3

**Table 1. Rules from BASE-1**

In order to identify common errors using the voice interface the BASE-2 was used to find association rules. The most common error identified was in the name identification task in the voice interface (Table 2). This error occurred due to the wide variety of names and the high complexity of the name grammar in the voice interface. The association rules and the confidence/occurrence relationship allowed us to verify that users choose to switch modes to recover from the error, after the second frustrated attempt to interact using voice. In 98.1% of the interactions with the names graphical interface we found a previous voice interaction with 71.8% confidence (VOICE  $\Rightarrow$  name GRAPHICAL (71.8, 98.1)).

Rule	Support	Confidence
VXML_ERROR $\Rightarrow$ name	3.3	33.6
Name $\Rightarrow$ VXML_ERROR	3.3	32.8

**Table 2. Rules from BASE-2**

## 4. Related work

A number of tools, methods and frameworks to multimodal interface design have become accessible in recent time. A method for developing Web-based multimodal interfaces was proposed by [12]. In this method a framework splits the interface life cycle into four levels: tasks model, domain model, abstract and concrete interface models, each of them performing transformations on the previous levels until the achievement of the final multimodal interface. All the elements, models and transformations between the levels are specified evenly through a single interface description language, the UsiXML. Thus, the whole project knowledge necessary to lead the changes is explicitly contained in the processing rules, and implementing these rules is granted by a transformation mechanism.

A tool to assist in the development of multiple types of interfaces and different ways to combine graphics and voice in multi-devices environments was proposed by [10]. The interface can be adapted to the interaction resources available to avoid confusing the designer with many details related to devices and programming languages. Bourguet [4] has investigated the creation of a multimodal toolkit in which multimodal scenarios could be modeled using finite state machines. A more theoretical approach is CARE and its component based approach ICARE [3] that has provided inspiration for a comprehensive toolkit: OpenInterface [11].

A multimodal framework, which architecture is based on agents geared toward direct integration into a multimodal application, was designed by [7]. One of the most interesting aspects is the use of a parallel application-independent fusion technique. We could observe that these methods do not consider principles, guidelines, DR or Design Patterns to facilitate the design of multimodal interfaces. We believe such concepts are most important since they guide designers in making consistent decisions through the elements which compose the product and they are efficient techniques to capture, document and communicate the scientifically validated and applied knowledge.

Additional concerns were also identified, given that these tools, methods and frameworks are specific to a particular issue even if multiple interactions are available; there is a lack of systems that convey information using the modalities that are most appropriate to the users and their tasks; there is a lack of experience recording and spreading DR with the employed multimodal interactions. Thus, our aim is to overcome these issues and concerns by providing an approach and a tool as it was explicitly shown in the previous sessions.

## 5. Conclusions

Multimodal interfaces are viewed as a promising opportunity for achieving universal access in the near future. On the other hand the field is still novel and needs further research to build reliable and usable multimodal applications due to a lack of supporting tools for the project team. With these statements in mind, we have created the MultiModal Web Approach and developed its authoring environment that aim to assist designing, implementing and testing multimodal user interfaces, promoting modality integration, error handling, dialog management and DR reuse in a clear manner. Furthermore we have show the significant results applying our approach and our tool in real projects through the accomplishment of case studies.

In summary the tool had: promoted code reuse; decreased the time to create prototype for usability evaluations; promoted design documentation; supported DR storage and retrieval and decreased the usability issues. We have also applied association rules to identify errors and users behaviors interacting with multimodal interfaces since it is a well know tool to discovering interesting relations between variables in large databases. The next step in our research is to increase the Design Pattern support in the MMWA-ae to allow for a more robust interface prototype generation.

## References

[1] D. Bohus and A. Rudnický. Sorry, I Didn't Catch That! - An Investigation of Non-understanding Er-

rors and Recovery Strategies. In *SIGdial*, 2005.

[2] J. Borchers. *A Pattern Approach to Interaction Design*. John Wiley & Sons, Inc., 2001.

[3] J. Bouchet, L. Nigay, and T. Ganille. ICARE Software Components for Rapidly Developing Multimodal Interfaces. In *Conference Proc. of ICMI 2004*, pages 251–258, State College, 2004. ACM Press, New York.

[4] M. Bourguet. A Toolkit for Creating and Testing Multimodal Interface Designs. In *Companion Proc. of UIST 2002*, pages 29–30, Paris, 2002.

[5] J. Coutaz, L. Nigay, D. Salber, A. Blandford, J. May, and R. Young. Four easy pieces for assessing the usability of multimodal interaction: the CARE properties. In *Proc. of Interact'95*, pages 115–120, ACM Press, 1995.

[6] H. Desurvire. *Usability inspection methods*, chapter Faster, cheaper!! Are usability inspection methods as effective as empirical testing?, pages 173–202. John Wiley & Sons, Inc., New York, 1994.

[7] F. Flippo, A. Krebs, and I. Marsic. A framework for rapid development of multimodal interfaces. In *Proc. of the 5th international conference on Multimodal interfaces*, Vancouver, B.C., Canada., 2003.

[8] A. T. Neto, T. J. Bittar, R. P. M. Fortes, and K. Felizardo. Developing and evaluating web multimodal interfaces - a case study with usability principles. In *SAC '09: Proc. of the 2009 ACM symposium on Applied Computing*, pages 116–120, New York, NY, USA, 2009. ACM.

[9] S. Oviatt, R. Lunsford, and R. Coulston. Individual Differences in Multimodal Integration Patterns: What Are They and Why Do They Exist?. In *Proc. of CHI'05*, pages 241–249. ACM Press, 2005.

[10] F. Paterno and F. Giammarino. Multimodal interaction: research papers: Authoring interfaces with combined use of graphics and voice for both stationary and mobile devices. In *Proc. of the working conference on Advanced Visual Interfaces*, 2006.

[11] M. Serrano, L. Nigay, J.-Y. Lawson, A. Ramsay, R. Murray-Smith, and S. Deneff. The OpenInterface framework: a tool for multimodal interaction. In *Adjunct Proc. of CHI 2008*, page 35013506, Florence, Italy, 2008. ACM Press, New York.

[12] A. Stanculescu, Q. Limbourg, J. Vanderdonckt, B. Michotte, and F. Montero. A Transformational Approach for Multimodal Web User Interfaces based on USIXML. In *Proc of the 7th international conference on multimodal interfaces ICMI 2005*, pages 259–266. ACM Press, New York., 2005.

# Human-Computer Interface Design Guidelines: An Expert System

Tiago Cinto, Cecilia Sosa Arias Peixoto  
Methodist University of Piracicaba  
Rodovia do Açúcar, Km 156  
CEP 13400-911, Caixa Postal 68  
Piracicaba – São Paulo, Brazil  
{tcinto, cspeixot}@unimep.br

**Abstract**—The development of user-centered human computer interfaces requires consideration of several design recommendations called guidelines. These guidelines are focused on interface usability and are proposed by several authors. This article proposes the use of an expert system to identify guidelines for a given project more effectively. The expert system developed suggests guidelines from the analysis of a situation described by the designer. Four hundred and thirty guidelines were included. The article also presents the technique used to organize the guidelines of several authors in the system.

**Keywords**—Expert Systems; Human-Computer Interfaces; Guidelines

## I. INTRODUCTION

The user-centered development has attracted designer's attention due to the growing needs to increase user interaction satisfaction.

Aiming at producing more natural and less hostile user interfaces, graphical elements have been introduced into them in order to represent data and tasks. This representation allows the user to directly interact with data and tasks. Actually, these items do not correspond to any data or even tasks; they are just their "signs". Therefore, signs production is a crucial factor which contributes to create a good performance user interface [9].

In order to be able to estimate the success or failure chances of solutions suggested by the designer for a user interface, there are, in the literature, broad sets of recommendations. These recommendations are called guidelines [1].

According to Nielsen [1], Human-computer interface (HCI) design guidelines may be used as background for heuristic evaluations of a user interface which is being developed. Nielsen [1] also describes a heuristic evaluation as a group of designers observing and analyzing the interface in order to identify usability issues and verify the application of guidelines to solve them. Nielsen proposes the use of guidelines in a specific stage of the "usability engineering model". The model is composed by three main stages: Pre-design stage, Design stage e Post-design stage [18]. The following activities compose the usability engineering lifecycle:

Pre-design stage:

- 1) *Know the user*: Study of intended users and use of the product, which includes individual user characteristics, task analysis, functional analysis, and evaluation of the user and the job.
- 2) *Competitive analysis*: Analysis of existing products as best prototypes that can include comparative analysis of competing products if they exist.
- 3) *Setting usability goals*: Setting levels of performance for usability attributes.

Design stage:

- 4) *Participatory design*: This means the involvement of users in the design process.
- 5) *Coordinated design of the total interface*: This step ensures the consistency of the entire user interface.
- 6) *Guidelines and heuristic evaluation*: There are general, category specific, and product specific guidelines that can be used as background for heuristic evaluation.
- 7) *Prototyping*: Fast produced versions of the system for early usability evaluations.
- 8) *Empirical testing*: Evaluation of the interface by user testing.
- 9) *Iterative design*: Production of new interfaces based on the usability problems identified in empirical testing.

Post-design stage:

- 10) *Feedback from field use*: Gathering usability data after the release of the product.

Besides heuristic evaluation, guidelines may also be used in the HCI design [14].

Nielsen's model provides details for the HCI design; however, the designer must be aware of guidelines in order to be able to design a high usability HCI.

Aiming at assisting HCI designers, an expert system was developed. This expert system has the task of recommending HCI design guidelines. These recommended guidelines may be used during the HCI design as well as in expert reviews. The expert system purpose [8] is to provide a fast, easy and

economical way which tells the HCI designer the design guiding principles.

#### A. Guidelines Classification

Nielsen [1] defines three basic categories of guidelines: General guidelines, category-specific guidelines and product-specific guidelines. General guidelines are applicable in all types of user interfaces. Category-specific guidelines are applicable only in a few types of user interfaces. Finally, product-specific guidelines only are applicable in an individual product.

One example of general guideline is proposed by Shneiderman [4]: "Design understandable, predictable and controllable user interfaces".

According to Nielsen [1], there are several extensive collections dedicated to elicit and propose guidelines for the HCI design. Two of them are Brown's collection [2], with a total of three hundred and two guidelines, and Mayhew's collection [3], with a total of two hundred and eighty eight guidelines. Through these numbers we see that guidelines application is not trivial. Therefore, the HCI designer needs to focus on what is needed and deal with design trade-offs [17]

#### B. HCI Design Support Expert Systems

Expert systems are designed to be used in problems which require a considerable amount of human knowledge and expertise [16]. In the HCI context, there is the expert system ACE (A Color Expert) [13] whose purpose is to assist HCI designers whenever they need to choose a color for a specific component of an HCI. According to ACE authors, color guidelines collections contain low quality guidelines. Though these collections contain guidelines aimed at eliciting which color use and which color not to use in a specific HCI, they do not contain general guidelines for the correct components color choosing. The ACE interface consists of some questions regarding components of a specific HCI and their relationship. Through these answers, ACE suggests which color to use in every HCI component.

A multidisciplinary approach to integrate software engineering, HCI design and artificial intelligence is proposed by [19]. The part concerning artificial intelligence is composed by an expert system whose main objective is to provide support and assistance in a HCI design. The expert system outputs are abstract user interfaces which are obtained through the analysis of conceptual models. These conceptual models involve descriptions of various HCI aspects such as HCI tasks and the relationship among these tasks. Some of the benefits obtained through this approach concern the generation of user interfaces adaptable to different types of users and the possibility of performing different types of tasks.

Mobi-D [11] is an interactive environment whose purpose is to represent relevant aspects of the HCI design combined with declarative models. Some of elements which compose this environment are decision support tools whose feature is shown below. The HCI design in Mobi-D starts with user tasks elicitation, a fundamental activity. This activity is performed by the designer and aided by users. From user tasks, the

designer begins to interact with the tool in order to build user tasks and domain model which will be used in the remaining activities. Regarding the decision support tools role, these analyze built models in order to provide presentation and design dialog recommendations such as which interaction object should be used to perform a sub-task.

#### C. The Developed Expert System

The expert system aims at suggesting user interface design guidelines by analyzing a situation described by the designer. In the early development stage, four hundred and thirty guidelines have been selected. All of them have been extracted from these authors: [1], [2], [4], [5], [6], and [14]. In order to allow the guidelines search and selection [8], it was necessary to group them according to the characteristics and goals they had in common. The resulting groups of guidelines are called meta-guidelines [8]. Structuring the database in this way, the guidelines search and selection occurs through the meta-guidelines rather than the analysis of guidelines one by one. The names of these meta-guidelines have been defined by analyzing the common goal which every guideline of a same group owned. For example, there were some guidelines which concerned about providing features to help protect user data, thus the created meta-guideline has been named "data protection".

The guidelines grouping resulted in a total of twenty five distinct meta-guidelines. All of them are shown below:

1. System feedback
  - 1.1. Response time
  - 1.2. System messages
  - 1.3. Response to user actions
  - 1.4. Feedback display
2. Data protection
3. Documentation
4. Accessibility
  - 4.1. Basic accessibility
  - 4.2. Advanced accessibility
5. Data display
  - 5.1. Numeric data display
  - 5.2. Textual data display
  - 5.3. Alphanumeric data display
6. User interface internationalization
7. Colors
  - 7.1. Color application
  - 7.2. Data highlighting
  - 7.3. Assistance to people with disabilities
8. Terminology
9. Design

- 9.1. Textbox design
- 9.2. Button design
- 9.3. Taskbar design
- 9.4. Icon design
- 9.5. Selection list and selector design
- 9.6. Window and message box design
- 9.7. Menu design
- 9.8. Label design
- 9.9. Table and graph design

In addition, since the system database is not restricted to this amount of meta-guidelines, it may be expanded whenever necessary.

## II. SYSTEM ARCHITECTURE

The system was designed according to what was proposed by Peixoto [7]. Therefore, there are four main components which form the whole system. They are: User interface, expert system, knowledge base, and database. The Fig. 1 shows the elements through their organization into layers and modules.

When the system starts, the expert system layer accesses the knowledge base contained in layer three to load stored knowledge rules. The user interface layer, one, gathers some information with the designer through modules one to three. Gathered information is analyzed by the expert system in order to select appropriate meta-guidelines. Finally, as result of this analysis, the system accesses the database (layer three) to retrieve guidelines according to meta-guidelines previously selected.

In addition to user interface modules one to three, there is user interface module four which is aimed at assisting a designer in a HCI evaluation and it is going to be presented with more details in a future section.

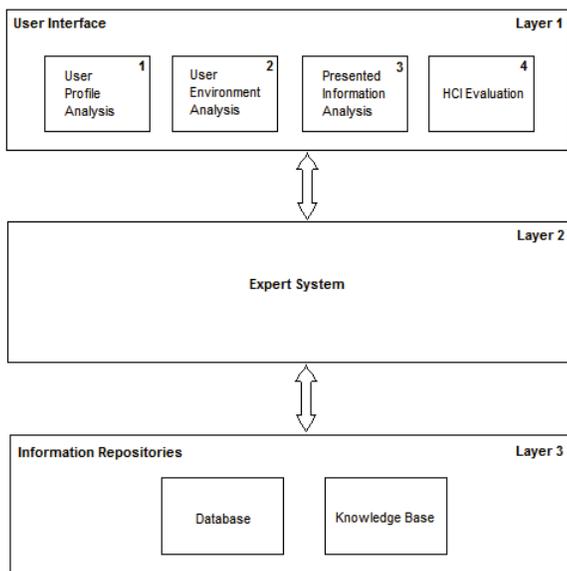


Figure 1. System architecture

### A. User Interface

The user interface performs three types of analysis with the designer. All of them have been made based on proposals [9], [10] and [12].

Netto [9] proposes a user profile analysis based on several questions. Some of them performed by the system are: Computer experience identification, domain knowledge identification (knowledge to perform the same tasks without help of computer) and application user frequency. Complementing Netto's user profile analysis model, there is a question which aims at identifying the existence of users with special needs.

Mandel [12] proposes a user environment analysis based on several aspects. According to him, the designer must gather user environment information regarding the physical environment type, the user location and mobility, and cultural considerations. For this analysis, the user interface asks whether the system under construction will be used in more than one country or region. For example, a system which will be used in more than one country needs to properly deal with conversions among several coin types [1].

Finally, according to Sommerville [10], in a presented information analysis, HCI designer must gather information regarding several aspects. Some of them are: Presented information type, direct manipulation user interface existence for managing such information, the importance of such information, and how fast information changes should be notified to the user.

Thus, the user interface presents one screen for each analysis type:

1. User environment analysis screen: It aims at identifying the existence of an internationalized system. The question the designer has to answer is shown in Fig. 2.
2. Presented information analysis screen: It aims at identifying main characteristics related to the type of information managed by the system under construction such as the information type, textual or numeric. The questions the designer has to answer are shown in Fig. 3.
3. User profile analysis screen: It aims at identifying majority characteristics in the user community such as computer experience and domain knowledge. The questions the designer has to answer are shown in Fig. 4.

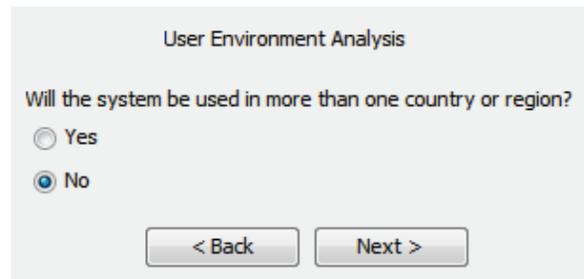


Figure 2. User environment analysis screen

Presented Information Analysis

Presented information type:

Textual

Numeric

Both

Do users need to interact with the displayed information via a direct manipulation interface?

Yes

No

Are relative values of information items important?

Yes

No

Should the change in a value be indicated immediately to the user?

Yes

No

< Back      Next >

Figure 3. Presented information analysis screen

User Profile Analysis

User according to the experience with computers:

Beginner

Intermediate

Experienced

User according to the knowledge to perform the same tasks without help of computer:

Novice

Intermediate

Expert

Application use frequency:

Occasional (Less than twice a month)

Frequent

Are there users with special needs (e.g., color-blindness)?

Yes

No

< Back      Next >

Figure 4. User profile analysis screen

### B. Knowledge base

The knowledge base goal is to store rules used to represent expert knowledge so that the expert system can work with these rules appropriately.

The knowledge base is composed by *when-then* syntax rules that are intended to indicate which are the meta-guidelines that best fit a under construction HCI. All necessary information to system execution is gathered by the user interface.

The system performs three analysis types: User profile analysis, user environment analysis, and presented information analysis. Therefore, the knowledge base has been structured to support all of them.

All the knowledge used to feed the knowledge base was gathered from the literature with the help of a usability expert. One of the books used was [9].

Concerning knowledge base update, a new rule must be inserted manually, since there is not any module to update the knowledge base with new rules yet.

### C. Expert System

The expert system inference engine uses the forward chaining strategy to analyze knowledge rules [15]. Through this strategy, the antecedent part of a rule is analyzed and then, in case of a rule that matches the described situation, the consequent part is executed.

Usually, a knowledge rule used with an expert system is *if-then* type; however, since JBoss Drools Platform [20] has been used to provide some elements of an expert system, the rule syntax used was *when-then* type.

The Fig. 5 presents two when-then syntax rules following Netto's recommendations [9]. For this representation, the rule predicates are *computer\_experience* and *meta-guideline*.

```

R1: When
    computer_experience == beginner
Then
    meta-guideline = data protection

R2: When
    computer_experience == experienced
Then
    meta-guideline = response time

```

Figure 5. User environment analysis screen

### D. Database

The system integrated database aims at storing HCI design guidelines. For each stored guideline, the main information is:

1. Guideline;
2. Guideline example;
3. Guideline justificative;
4. Guideline reference (composed by book, year, page, author, local, and publisher).

Since it depends on the author to inform guideline information, guideline example and justificative may or may not be present.

## III. GUIDELINES SELECTED BY THE SYSTEM

The system output is composed by a set of guidelines presented to the designer. It allows the designer to perform expert reviews or to design a new HCI.

A set of guidelines (such as in Fig. 7 showing results from Fig. 2, 3 and 4) is suitable for design inspiration, as a checklist in heuristic evaluation or can serve as a reference for answering specific design questions [18].

**Guidelines**

1. **Guideline: The button labels should be formatted using system fonts and have the same size.**  
[1]Pg.408
2. **Guideline: Provide a descriptive label to identify the type of information to be typed into a text box.**  
[1]Pg.422
3. **Guideline: Limit the use of fonts for text (up to two types).**  
[2]Pg.95
4. **Guideline: Do not use fonts smaller than 12 points for screens and smaller than 10 points for printed material.**  
[2]Pg.95

**References**

[1] Galitz WO. *The Essential Guide to User Interface Design: An Introduction to GUI Design Principle and Techniques*. New York: John Wiley & Sons, Inc.; 2002  
 [2] Cybis W, Betiol AH, Faust R. *Ergonomia e Usabilidade: Conhecimentos, Métodos e Aplicações*. São Paulo: Novatec; 2007.

Figure 6. Some of the guidelines selected by the expert system

Finally, by using system proposed guidelines, high quality user interfaces will be produced. Nielsen [18] says: “to ensure the usability of interactive computer products, we must actively include usability concerns in the software development process”.

#### IV. HUMAN-COMPUTER INTERFACE EVALUATION

Shneiderman [21] places guidelines in a relation composed by four pillars. This relation demonstrates elements composing a successful HCI design. These elements are: HCI requirements, guidelines, HCI software tools, and usability testing.

Concerning usability testing, there are different types of tests that may be performed. One of them is expert reviews [21]. Expert reviews are those in which experts analyze the user interface in order to find design issues. After that, solutions based on guidelines may be proposed.

Shneiderman [21] also defines six different types of expert reviews: Heuristic evaluation, guidelines review, consistency inspection, cognitive walkthrough, metaphors of human thinking, and formal usability inspection. Heuristic evaluation and guidelines review are those in which HCI guidelines may be used.

Heuristic evaluation is the critique of an HCI regarding a short list of heuristics such as ten usability heuristics from Nielsen [1]. These heuristics are more general than specific HCI guidelines. In this type of review, HCI guidelines may be used as background [1].

In a guidelines review, the interface is checked according to specific HCI guidelines, such those manipulated by the system.

Expert reviews are cheaper than a traditional usability tests such as tests involving user watching and they may be performed at different points in the development process. In addition, it is a test which may take little time or few days to be performed [21].

#### A. HCI Evaluation and the Expert System

Besides suggesting guidelines for a under construction HCI as described before, the expert system may also be used as a way of providing guidelines for an expert review. Aiming at assisting designers in a HCI expert review, it has been developed a module providing only relevant aspects to the designer in this phase. Through one screen, the designer may select some areas that need to be evaluated and select guidelines for those. The Fig. 6 shows all areas available to be chosen.

**HCI Evaluation**

<p><b>A-C</b></p> <ul style="list-style-type: none"> <li><input type="checkbox"/> Advanced Navigation</li> <li><input type="checkbox"/> Alphanumeric Data</li> <li><input type="checkbox"/> Assistance to People with Disabilities</li> <li><input type="checkbox"/> Basic Navigation</li> <li><input type="checkbox"/> Buttons</li> <li><input type="checkbox"/> Colors</li> </ul>	<p><b>D-I</b></p> <ul style="list-style-type: none"> <li><input type="checkbox"/> Data Highlighting</li> <li><input type="checkbox"/> Data Protection</li> <li><input type="checkbox"/> Documentation</li> <li><input type="checkbox"/> Feedback Presentation</li> <li><input type="checkbox"/> Icons</li> <li><input type="checkbox"/> Internationalization</li> </ul>
<p><b>J-S</b></p> <ul style="list-style-type: none"> <li><input type="checkbox"/> Labels</li> <li><input type="checkbox"/> Menus</li> <li><input type="checkbox"/> Numeric Data</li> <li><input type="checkbox"/> Response Time</li> <li><input type="checkbox"/> Response to User Actions</li> <li><input type="checkbox"/> Selection List and Selectors</li> <li><input type="checkbox"/> System Messages</li> </ul>	<p><b>T-Z</b></p> <ul style="list-style-type: none"> <li><input type="checkbox"/> Tables and Graphs</li> <li><input type="checkbox"/> Taskbars</li> <li><input type="checkbox"/> Terminology</li> <li><input type="checkbox"/> Textboxes</li> <li><input type="checkbox"/> Textual Data</li> <li><input type="checkbox"/> Windows and Message Boxes</li> </ul>

Choose which system area needs to be evaluated. After that, press 'OK' so that guidelines will be selected in order to help you with the evaluation.

Figure 7. HCI evaluation screen

After selecting desired areas, some HCI guidelines will be presented to the designer. The presentation of these guidelines is done in the same way as shown before with Fig. 6.

## V. CONCLUSION

This article presented the developed expert system and the technique used to organize the guidelines of several authors inside the system database.

A set of questions have been selected and organized in order to provide the search of guidelines for a specific project. The meta-guidelines concept was introduced to reduce the search space and speed up the choice of guidelines for a specific HCI project.

The system does not have a guidelines acquisition interface yet. However, a module which allows the designer to insert a new guideline with the identification of its meta-guideline will be developed. The system will also be expanded with the acquisition of more guidelines.

The system is going to be used in the first semester of 2010 during the practical lessons of Human-Computer Interfaces subject at Methodist University of Piracicaba. Certainly, by using the system, system instructions will be able to be improved in order to become more efficient in the HCI design support.

## REFERENCES

- [1] J. Nielsen, *Usability Engineering*. Boston: Academic Press, 1993.
- [2] C. M. L. Brown, *Human-Computer Interface Design Guidelines*. Norwood: Ablex Publishing Corp., 1988.
- [3] D. J. Mayhew, *Principles and Guidelines in Software User Interface Design*. New Jersey: Prentice Hall, 1992.
- [4] B. Shneiderman, *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Boston: Addison Wesley Longman, Inc., 1998.
- [5] J. Johnson, *GUI Bloopers 2.0: Common User Interface Design Don'ts and Dos*. San Francisco: Morgan Kaufmann, 2007.
- [6] W. O. Galitz, *The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques*. New York: John Wiley & Sons, Inc., 2002.
- [7] C. S. A. Peixoto, "Human-computer interface expert system for agile methods", *Proceedings of the 31st International Conference on Information Technology Interfaces*, Cavtat, Croatia, June 22-25, 2009, pp. 311-316.
- [8] C. S. A. Peixoto and A. E. A. Silva. "A conceptual knowledge base representation for agile design of human-computer interface", *Proceedings of the 3rd International Conference on Intelligent Information Technology Application*, Nanchang, China, 21-22 November, 2009, pp.156-160.
- [9] A. A. O. Netto, *IHC: Modelagem e Gerência de Interfaces com o Usuário*. Florianópolis: Visual Books, 2004.
- [10] I. Sommerville, *Software Engineering*. Boston: Addison-Wesley Longman Publishing Co., Inc., 2001.
- [11] A. R. Puerta, *A Model-Based Interface Development Environment*, <http://www.arpuerta.com/pdf/iee97.pdf> [02/01/2010].
- [12] T. Mandel, *The Elements of User Interface Design*. New York: John Wiley & Sons, Inc., 1997.
- [13] B. J. Meier, "ACE: A color expert system for user interface design". *Proceedings of the 1st Annual ACM SIGGRAPH Symposium on User Interface Software*, Alberta, Canada, 17-19 October, 1988, pp.117-128.
- [14] W. Cybis, A. H. Betiol, and R. Faust, *Ergonomia e Usabilidade: Conhecimentos, Métodos e Aplicações*. São Paulo: Novatec Editora Ltda, 2007.
- [15] S. Russell and P. Norvig, *Artificial Intelligence*. New Jersey: Prentice Hall Series, 2003.
- [16] S. Rezende, *Sistemas Inteligentes: Fundamentos e Aplicações*. São Paulo: Manole, 2005.
- [17] H. V. Rocha and M. C. Baranauskas, *Design e Avaliação de Interfaces Humano-Computador*. Nied, Unicamp, 2005.
- [18] J. Nielsen, "The usability engineering lifecycle", *IEEE Computer Society Press*, vol. 25, pp. 12-22, March 1992.
- [19] E. Furtado, V. Furtado, K. S. Souza, J. Vanderdonck, and Q. Limbourg, "KnowiXML: A knowledge-based system generating multiple abstract user interfaces in USIXML", *Proceedings of the 3rd Annual Conference on Task Model and Diagram*, Prague, Czech Republic, 15-16 November, 2004, pp.121-128.
- [20] Drools Expert: Community Documentation, [http://downloads.jboss.com/drools/docs/5.0.1.26597.FINAL/drools-expert/html\\_single/index.html#d0e237](http://downloads.jboss.com/drools/docs/5.0.1.26597.FINAL/drools-expert/html_single/index.html#d0e237) [03/05/2010].
- [21] B. Shneiderman, *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 5th ed.. Boston: Addison Wesley Longman, Inc., 2010.

# Assisting Developers to Read Code Help-Documents Efficiently through Discovering Document-section Relationships

Lijie Wang, Leye Wang, Ge Li<sup>†</sup>, Bing Xie

Software Institute, EECS, Peking University, P.R.China  
Key Laboratory of High Confidence Software Technologies,  
Ministry of Education of P.R.China  
{wanglj07, wangly09, lige, xiebing}@sei.pku.edu.cn

**Abstract** — During software development, in order to reuse or maintain some existing codes (e.g. some classes, interfaces and methods), developers often need to figure out how the codes work and how to use them. Whenever this time, developers often need to read a lot of help-documents about the existing codes. However, in some projects, help-documents are very large and the contents related to the target codes are often scattered in different sections or even in different documents. So, developers often have to spend much time and effort searching and reading the documents by jumping from here to there or even from one document to another. In many cases, the reading becomes so tough and time-consuming that it makes many developers abandon their tasks. Especially in some big projects this situation is particularly serious. In this paper, we proposed a document-section relationships discovering method and developed a novel tool named Document Reading Assistant (DRA) to assist developers to read help-documents. In our method, the background knowledge contained in the class diagrams corresponding to the target codes is leveraged to discover the relationships among different document-sections, and then DRA will use these relationships to guide developers to finish their reading efficiently. Our experiment shows that, with our assistant tool, the developers can finish their reading more efficiently and easily.

**Keywords**-software reuse, document reading, document-section relationships discovering, software maintenance;software tools

## I. INTRODUCTION

Software reuse and software maintenance are important activities for software development. During reuse-based software development and software maintenance, developers need to figure out how the target codes (e.g. some classes, interfaces and methods) work and how to use them. In order to reuse or maintain the existing codes, developers must try to understand them and make it clear that how to carry out the task at hand. Besides referring to the source code or example code of the target code, developers often need to read the help-documents such as user manual and design documents about the existing codes.

However, in some projects, the help-documents are very large and usually contain a lot of files. Furthermore, the contents related to the target code are often scattered in different sections or even in different documents. So, developers often have to spend much time and effort searching and reading the documents by jumping from here to there or even from one document to another. In many cases, the reading becomes so tough and time-consuming that it makes many

developers abandon the task. Especially in some big project this situation is particularly serious.

In this paper, we proposed a document-section relationships discovering method and developed a novel tool named Document Reading Assistant (DRA) to assist developers to read help-documents. In this method, the background knowledge contained in the class diagrams corresponding to the target codes is leveraged to discover the relationships among different document-sections. And our tool will show internal relationships among document-sections visually to developers. With this tool, whenever a developer focuses on an interested section in the help-documents as a beginning, DRA will guide the reading process by finding out the most related sections from all the documents and recommend them to the developer. By this means, developers can finish their reading more efficiently and easily.

Differently from the general text similarity matching algorithms such as Vector Space Model (VSM) [9], our document-section relationships discovering method used in DRA aims to discover the internal relationships among the text sections in help-documents according to the background knowledge contained in the class diagrams corresponding to the target codes. In other words, we use class diagrams as an intermediary to discover the relationships between document-sections instead of only calculating text similarity.

Our approach contains the following three steps:

1. Each imported help-document is divided into many sections according to the subtitles. Then, these sections are stored as a tree structure named subtitle tree.
2. Discover the relationships among all the document-sections by our proposed algorithm using the background knowledge in class diagrams corresponding to the target codes.
3. When a section is being read, the related document-sections will be recommended visually to developers according to the discovered relationships.

Because the related document-sections recommended by our tool is discovered according to the class diagrams corresponding to the target codes, but not only according to the general text similarity as many traditional methods did, more valuable relevant document sections can be found, meanwhile more unrelated document sections can be pass over. As a result, developers can read more relevant document sections of the target codes during their reading process with less effort, so they can finish their reading more efficiently and easily.

The rest of this paper is organized as follows: section II presents the overview of our approach. The core algorithm is

<sup>†</sup>Corresponding Author

described in section III. Section IV is the description about our tool—DRA. We conduct an experiment and a case study to evaluate our work in section V. Related work is presented in section VI. Section VII draws the conclusion for our work.

## II. PROPOSED APPROACH

As Figure 1 shows, there are three steps in our approach. We will describe them in detail in the following three subsections, respectively.

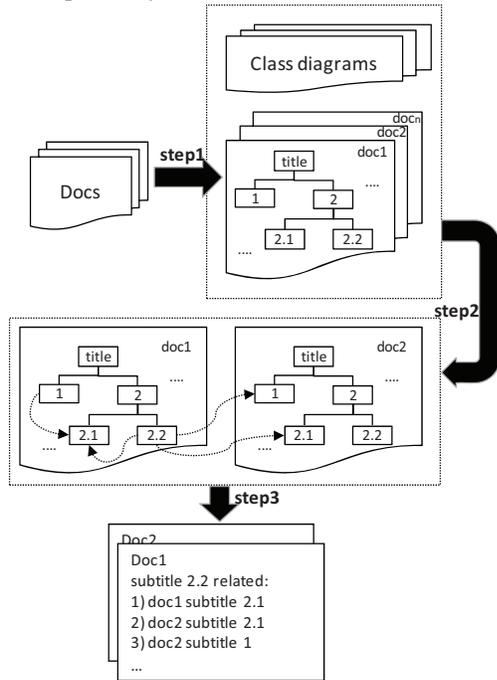


Figure 1. Overview of our approach

### A. Step 1: Document Segmenting

The content of documents is primarily plain text and is usually organized according to some kind of structure. Our first task is to get the document structure. A common method is to use the subtitle tree. The subtitles split the document into many sections, which may have parent-child relationships with each other. For example, the section with subtitle “1.1.” can be seen as the child of the section with subtitle “1.”

For help-documents, all the paragraphs in the same subtitle usually focus on the same topic such as how some part of the code works or how to use some part of the code. So we divide the document in accordance with subtitles, which makes sense in most conditions.

After document segmenting, a help-document is reorganized as a tree. Each node, which has its own subtitle and content, in the tree represents a section of the document, each leaf in the tree represents the smallest subtitles in the document. In the next step, the internal relationships between these sections will be discovered.

### B. Step 2: Relationships Discovering

This step is the core of the whole process. All the document-sections got in the last step and the class diagrams of

the target codes are used as input. In order to make the description simple and easy to understand, we will describe the algorithm separately, the frame of algorithm will be discussed first in this section. The detail of algorithm will be discussed in section III. The frame of our algorithm is shown in Figure 2, it contains four stages.

**Stage 1:** We get all the classes in the class diagrams of codes and all the document-sections in each help-document ready as inputs.

**Stage 2:** For each document section  $s$ , the classes related with  $s$  are obtained by the algorithm named *Algorithm\_s2C*. The algorithm takes  $s$  and all the classes in the class diagrams as input and then calculates the score between  $s$  and each class. The score represents the degree to which  $s$  and the corresponding class are related. If the score between  $s$  and a class  $c$  is greater than zero, we will put  $c$  in the result set  $C$ . *Algorithm\_s2C* will be described in section III-A in detail. For each  $c$  in  $C$ , the relationship between  $s$  and  $c$  can be shown as:

$$\langle s, c, score_{s2c} \rangle,$$

where  $score_{s2c}$  represents the relativity degree between section  $s$  and class  $c$ .

**Stage 3:** Secondly, for each class  $c$  in the result set  $C$  get in stage 2, the sections related with  $c$  are obtained by the algorithm named *Algorithm\_c2T*. Similarly to *Algorithm\_s2C*, this algorithm takes  $c$  and all the document-sections as input and then calculates the score between  $c$  and each section. The score represents the degree to which  $c$  and the corresponding section are related. If the score between  $c$  and a section  $t$  is greater than zero, we will put  $t$  in the result set  $T$ . *Algorithm\_c2T* will be described in section III-B in detail. For each  $t$  in  $T$ , The relationship between  $c$  and  $t$  can be shown as:

$$\langle c, t, score_{c2t} \rangle,$$

where  $score_{c2t}$  represents the relativity degree between class  $c$  and the section  $t$ .

Taking into account the relationship  $\langle s, c, score_{s2c} \rangle$  obtained in Stage 2, we know  $T$  contains the sections related with  $s$  through  $c$ . For a section  $t$  in  $T$ ,  $s$  and  $t$  have the relationship:  $\langle s, t, c, score_{s2c}, score_{c2t} \rangle$ .

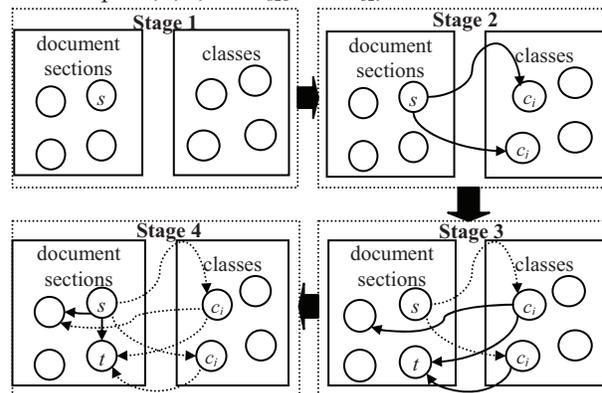


Figure 2. Four stages in relationships discovering

**Stage 4:** Stage 3 is repeated for each  $c$  in  $C$ . Then a lot of  $T$ s are produced. We use  $T_i$  to represent the  $T$  related with  $c_i$ . The final result set  $T_{final}$  can be obtained by putting  $T_i$  together. For a  $t$  in  $T_{final}$ , it may be present in different  $T_i$  at the same

time. For example, if a section  $t$  is present in both  $T_i$  and  $T_j$ , then  $s$  and  $t$  have these relationships simultaneously:

$$\langle s, t, c_i, score_{s2c_i}, score_{c_i2t} \rangle$$

$$\langle s, t, c_j, score_{s2c_j}, score_{c_j2t} \rangle$$

As a result, the relations discovered between sections here are actually multi-dimensional. Each dimension is based on a class in the target codes.

For each document-section, the four stages will be performed until all the relationships are discovered.

### C. Step 3: Recommendation

After the relationships are discovered, some work should be performed to convert relationships to recommendations:

1. The relationships discovered are multi-dimensional, which are inappropriate to display directly. So we need to convert the multi-dimensional relations into a one-dimensional relation. The following method is used:

For  $s$  and  $t$ , if they have relationships like this:

$$\langle s, t, c_i, score_{s2c_i}, score_{c_i2t} \rangle$$

$$\langle s, t, c_j, score_{s2c_j}, score_{c_j2t} \rangle$$

$$\langle s, t, c_k, score_{s2c_k}, score_{c_k2t} \rangle \dots$$

Convert the multi-dimensional relationships to a one-dimensional relationship:

$$\langle s, t, C_i, score_{s2t} \rangle, \text{ where } C_i: \{c_i, c_j, c_k, \dots\} \text{ and}$$

$$score_{s2t} = \sum_{c \in C_i} score_{s2c} * score_{c2t}$$

Here,  $C_i$  can be seen as the background knowledge of the relationship between  $s$  and  $t$ , which may help developers decide whether  $t$  is worth reading.

2. It is worth nothing that the tree structure of the document should be taken into consideration for relationship discovering. After the relationships are changed to one-dimensional ones, the effect of the tree structure can be involved. The main principle here is that the relationships between section  $s$  and the direct children sections of a section  $t$  should contribute to the relationship between  $s$  and  $t$ .

$$finalscore_{s2t} = score_{s2t} + \sum_{i=1}^n finalscore_{s2ct_i} / n$$

$n$ : number of direct children of  $t$  in the subtitle tree

$ct_i$ : direct child of  $t$  in the subtitle tree,  $i$  is from 1 to  $n$

3. Sorted by the  $finalscore$ , related sections can be recommended to the developer when section  $s$  is being read.

## III. ALGORITHMS IN RELATIONSHIPS DISCOVERING

This section describes the two algorithms mentioned in section II-B in detail.

### A. Algorithm\_s2C

Algorithm\_s2C is used in Stage 2 of Step 2, by which we can find the related classes for a source section  $s$ . The algorithm aims to calculate  $score_{s2c}$ . Overall, it uses the

frequency of  $c$ 's class name appearing in the title and content of  $s$  to determine  $score_{s2c}$ . Following is the pseudo-code:

```

Input:
  s : the section being read.
  allClasses : the set composed of all the classes.
Output:
  C : the result set which contains the related classes.
Begin:
  for each c in allClasses
    float score_s2c = 0;
    if (c.name appears in s.subtitle)
      score_s2c += weight_subtitle;
    end if
    int freq = countFreq(c.name, s.content);
    score_s2c += countScore(freq);
    if (score_s2c > 0)
      put c in the result set C
      s and c has a relation: <s, c, score_s2c>
    end if
  end for
End

```

In the code above, we distinguish two kinds of positions where  $c$ 's name appears in section  $s$  to calculate  $score_{s2c}$ .

1. When  $c$ 's name appears in  $s$ 's subtitle, regardless of frequency, we add  $weight_{subtitle}$  to  $score_{s2c}$ . In our implementation, we set  $weight_{subtitle}$  to 1.0.

2. When  $c$ 's name appears in  $s$ 's content, the amount added to  $score_{s2c}$  is calculated by  $countScore(freq)$ , which is implemented as follows:

```

Input:
  freq : frequency of c's name appearing in s's content.
Output:
  score : a part of score_s2c which is calculated based on freq.
Begin:
  float score = 0;
  if (freq > 0)
    score = 0.5 + 0.05*(freq-1);
  end if
  if (score > 1)
    score = 1;
  end if
  return score;
End

```

This implementation makes a great difference in  $score$  when  $freq$  is 0 and 1. The  $score$  rises with the increase of  $freq$ . Considering that the coarse-grained sections would be in the ascendant, we set a maximum value for the returned  $score$ , i.e. 1.0 in this paper which takes good effect in practice.

### B. Algorithm\_c2T

Algorithm\_c2T is used in Stage 3 of Step 2, by which we can find the related sections for class  $c$ . The algorithm aims to calculate  $score_{c2t}$ , which mainly uses term frequency and inverse document frequency to count the score.

The score formula is:

$$w(term, doc, D) = tf_{term, doc} / |doc| * \log(|D| / df_{term})$$

$tf_{term, doc}$ : frequency of term in doc

$|doc|$ : number of all the terms in doc

$D$ : set of all docs

$|D|$ : number of docs

$d_{f_{term}}$ : frequency of docs that contain term

The  $score_{c_{2t}}$  can be got by the following formula:

$$score_{c_{2t}} = w(c.name, t.subtitle, Allsubtitles) * \alpha + w(c.name, t.content, Allcontents) * \beta$$

$Allsubtitles$ : set of the subtitles of all the sections

$Allcontents$ : set of the contents of all the sections

$\alpha$  and  $\beta$  represent the importance of subtitle and content in the calculation, respectively. In our implementation,  $\alpha$  is set to 0.4 and  $\beta$  is set to 0.6.

#### IV. OVERVIEW OF THE TOOL: DRA

Determining how to display the recommendations to developers is also an important issue, which will make a great impact on users. If the recommendations are not well displayed, it will confuse developers and takes little effect.

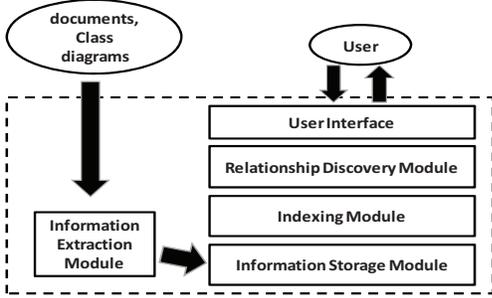


Figure 3. The architecture of DRA

Based on our approach, we developed a tool which is called Document Reading Assistant (DRA) to help developers read help-documents. The architecture of DRA is shown in Figure 3.

So far, our tool supports Microsoft word file with the extension ‘doc’ and the class diagram file generated by EclipseUML [8]. EclipseUML is a plug-in for Eclipse, which could be used to create UML diagrams.

DRA contains five parts:

1. The *information extraction module* is responsible for extracting information from documents and class diagram files.

2. The *information storage module* stores the information extracted from files. The document segmenting in Step 1 of our approach is carried out in this module.

3. The *indexing module* is built upon the information storage module. It creates the indices for both the subtitles and contents of document-sections. These indices provide support for the relation discovering module.

4. The *relationship discovery module* is the most important module. It deals with the major steps in our approach, including discovering relationships, recording relationships, etc.

5. The *user interface* can significantly impact on user experience. We designed the interface by reference to the interfaces of some outstanding IDEs, such as Eclipse and Visual C++, in order to make DRA user-friendly.

The snapshot of the main feature of DRA is shown in Figure 4. DRA shows the section being read and the related sections at the same time in order to bring the user convenience

in studying a software project. As we can see, as users reading a document section, the sections with the closest relationships with the section being read would be listed beside the source section. Users can jump to a section directly by double-clicking the corresponding subtitle.

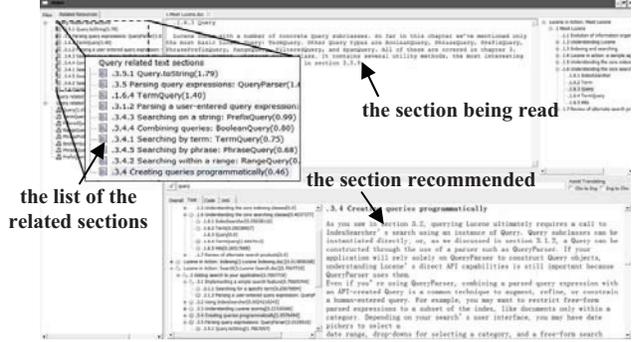


Figure 4. Recommending related sections

In addition to the main feature, DRA also provides some other features. For instance, DRA can show users the class diagram visually, and can find the sections related with a class in the class diagram (the algorithm in section III-B is reused here). Due to the limited space, please refer to [18] for further information about DRA.

#### V. EVALUATION

To evaluate our approach and the implemented tool, we conducted an experiment and a case study, which are presented in the next subsections, respectively.

##### A. The Experiment

In this section, we conduct an experiment to assess our approach using two widely accepted IR metrics: *recall* and *precision*. Recall is the ratio of relevant document sections retrieved for a query (i.e. the document-section a developer is reading) over the total number of relevant document-sections for the query. Precision is the ratio of relevant document sections retrieved for a query over the total number of retrieved document sections for the query. For a given document-section  $d_i$ , we assume that there are  $R_i$  document-sections relevant to it, and there are  $K_i$  relevant document sections among the total  $N_i$  retrieved document-sections. The recall and precision in this case are defined by the following formulas:

$$Recall = \frac{K_i}{R_i} = \frac{\# relevant_i \wedge retrieved_i}{\# relevant_i} \%$$

$$Precision = \frac{K_i}{N_i} = \frac{\# relevant_i \wedge retrieved_i}{\# retrieved_i} \%$$

Obviously, high recall and high precision are what we should pursue. However, the consequence of higher precision is generally a lower recall (and vice versa). For the entire system, the average recall and precision are defined as follows, where  $i$  ranges over the entire query document-sections set:

$$Recall = \frac{\sum_i K_i}{\sum_i R_i} = \frac{\sum_i \# relevant_i \wedge retrieved_i}{\sum_i \# relevant_i} \%$$

$$Precision = \frac{\sum_i \mathcal{K}_i}{\sum_i \mathcal{N}_i} = \frac{\sum_i \#relevant_i \wedge retrieved_i}{\sum_i \#retrieved_i} \%$$

This metric is also widely used by researches on traceability links recovery between software artifacts to evaluate the recovery results [10, 11].

We conduct the experiment on Lucene [7], a well-known open source information retrieval library containing two main functions: text indexing and searching. The version of Lucene we use in the experiment is release 1.4 consisting of 231 classes. We use ‘Lucene in action’ [6] as the help-document, which is a very popular book introducing Lucene 1.4 comprehensively; it consists of 10 chapters, 421 pages. It is unpractical to label all of the contents for the experiment; we select the first four chapters of this book to conduct the experiment. Chapter 1 is an overview about Lucene; Chapter 2 introduces the indexing functionality in detail, Chapter 3 describes the searching functionality, and Chapter 4 introduces analysis, which is an important factor for indexing and searching. There are total 113 sections, 148 pages in these 4 chapters. From the 113 sections, we randomly select 50 sections for which we labeled their relevant sections manually. Among these 50 sections, 4 have one relevant section, 5 have two relevant sections, 12 have three relevant sections, 17 have four relevant sections, 4 have five relevant sections, and 8 have more than five relevant sections. There are no sections which do not have any relevant segments. Then we use our approach to recommend sections for these 50 sections from the 113 sections, and calculate the average recall and precision according to different cut values. Result is shown in Figure 5.

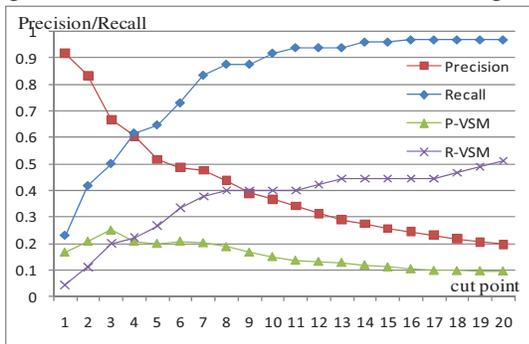


Figure 5. Recall and precision values for experiment with our approach (i.e. Precision and Recall) and experiment with VSM (i.e. P-VSM and R-VSM) using Lucene 1.4 and Lucene in action.

From Figure 5, we can see that at cut point 1, the average precision is over 90% while recall is 23%. Recall reaches over 90% when cut point is 10, where the precision is 37%. Recall reaches 96% at cut point 16. It is worth nothing that it is unpractical to find all of the traceability links only using automatic method. Human interaction should be leveraged to achieve this goal. Generally, it is considered by many researches that a good compromise for using IR methods in traceability links recovery would be getting on average a good recall (typically at least 70%), with an acceptable precision (typically at least 30%) [17]. From this point of view, the result of our approach is acceptable. The result also indicates that our approach can restrict the document space to read at the level of

10 to ensure that a developer can find the actually relevant segments roughly with precision being 40%.

Note that, instead of simply using textual similarity, our approach compute the relativity between document sections based on class diagram. To compare our approach with textual similarity based approach, we carried out the experiment again by computing the relativity between document sections according to their textual similarity calculated with Vector Space Model (VSM). The precision and recall values for this experiment are shown in Figure 5 as P-VSM and R-VSM, respectively. Obviously, our approach outperforms VSM based approach significantly. With cut point below 20, the greatest precision of VSM based approach is only 25%, while the greatest recall is 51%. The poor performance of textual similarity based approach might because there is too much useless information (terms) which will influence the result negatively.

However, our ultimate goal is to help software developer learn software project, and further, to reuse or maintain the software project better. To verify whether our tool can really save the effort a developer need spend in learning a software project, we conduct a case study in the next subsection based on Lucene.

## B. The Case Study

In this study, we assign some tasks to the subjects to answer some questions about Lucene. Some subjects are provided with our tool and the other subjects can only learn the target software manually. By comparing the time spent by different subjects giving right answers, we evaluate the effect of our tool. As mentioned, Lucene consists of two main functions: indexing and searching. In the case study, we designed a questionnaire which contains two parts of problems. Part 1 is about indexing and Part 2 is about searching. Eight senior students were selected to participate in the experiment. They all major in computer science, and they do not know Lucene before the case study took place. These students were divided into three groups. Group 1 and Group 2 could read the help-document from paper, while Group 3 could read them on computer.

In Lucene in action, section 1.5 and section 1.6 are the overview of indexing and searching, respectively. The difference between Group 1 and Group 2 is that Group 1 was given the recommendations of section 1.5, while Group 2 was given those of section 1.6. Group 3 used DRA to solve the problems. So the recommendations of both section 1.5 and 1.6 are available to the subjects of Group 3.

The answers of the questions in the questionnaire are distributed in many sections. The recommendations of section 1.5 (the overview of indexing) might help solve the problems of Part 1 (the problems about indexing), while the recommendations of section 1.6 (the overview of searching) might help solve the problems of Part 2 (the problems about searching). The experiment aimed to evaluate the effect of these recommendations.

The time each subject consumed was recorded. Furthermore, we asked the subjects about their experience in accomplishing the task after they finished the experiment.

### 1) Case Study Result

The time each subject consumed is shown in Table 1. The results with bold font are the time consumed by the subjects with recommendations.

Firstly, we discuss the groups which read the documents from paper. We can see that Group 1 with the recommendations related with indexing spent less time than Group 2 apparently in Part 1. While in Part 2, Group 2 with the recommendations related with searching does not show significant advantage and even spent a little more time in the mass. The following may be the reasons leading to the condition:

TABLE I. THE TIME EACH SUBJECT CONSUMED

Environment	Group	No.	Part 1 (min.)	Part 2 (min.)
Paper	1	1	<b>25</b>	11
		2	<b>14</b>	10
		3	<b>20</b>	15
	2	4	34	<b>13</b>
		5	32	<b>14</b>
		6	27	<b>10</b>
Computer (DRA)	3	7	<b>20</b>	<b>12</b>
		8	<b>20</b>	<b>13</b>

1. The problems of Part 1 are more difficult than those of Part 2. As the students must spend much time finding out the location where the section related with the problems is, the section recommended plays an important role. While the problems of Part 2 are easier, it's not difficult for the subjects of Group 1 to find the related section without the recommendations.

2. If the recommendations are not meaningful for the problem, which may occur occasionally, the subject will waste some time consulting the recommendations and then refer back to the structure of the documents such as the catalog to solve the problem. This kind of wasted time can be ignored in the difficult task like Part 1. But in the easier task like Part 2, it can't be ignored.

Next, let's see the result of Group 3. The time spent in Part 2 is still similar to all the other subjects'. The possible reasons have been discussed above. Then we focus on the time spent in Part 1. It is similar to that of Group 1. It seems that using our tool has no advantage over the recommendations from paper. Some experiments [5] have shown that reading paper is more comfortable and efficient than on-line documents. Taking into account the fact, we can conclude that our tool does improve the efficiency in reading those on-line documents.

### 2) Case Study Result Analysis

In contrast to the results above, all the subjects, including those in Group 2, think that the recommendations are useful and make the tasks easier more or less. This shows that even the recommendations do not take effect in increasing efficiency in some cases; it is certainly effective to improve the user experience.

Some subjects (3 of 6) in Group 1 and 2 complained that the recommendations on paper were hard to use, especially at the beginning of the experiment. They had to spend some time getting familiar with the use of the recommendations. But they still utilized the recommendations to finish the task in the end and approved the effect of the recommendations.

The subjects in Group 3 had no such complaints. In fact, while using DRA, they were more dependent upon the recommendations. And the use of those recommendations did not confuse them. The tool could display both the section being read and the sections recommended in one screen, which brought the subject convenience.

From the result, we can see that our approach and the tool improve the efficiency in reading the help-documents to finish a task, although the easy task in this case cannot show the advantage significantly. Considering the difficulty in learning a real software project, we believe that this tool could help the developers in the studying process. On the other hand, the experiment shows that the tool does improve user experience in reading the documents, which is another important reason to confirm the meaning of the tool.

In addition, it is notable that the help-document we used during the evaluation, i.e. 'Lucene in action', is a published book. This means that its content has been arranged carefully before being published to make it more readable. But in real life, the help-documents for many software projects, especially some open source projects, are generally not well organized. Developers will run into many problems during trying to learn these projects. We believe that the improving effectiveness of our approach and tool for developers to learn software projects will be better for the software projects whose help-documents are not well organized. In the future, we plan to carry out some further experiments and case studies on more various kinds of projects to evaluate the effect of our approach and tool further.

## VI. RELATED WORK

### A. Traceability Link Recovery

There are many artifacts produced during software development. How to manage or recover the traceability links between artifacts has been received much attention in the literature. Some of them focus on managing traceability links among various artifacts during software development phase, such as TOOR in [12], IBIS in [13], and REMAP in [14]. But they either rely on naming conventions or assigning the links manually by developers. Some other researches recover traceability links between design and source code by leveraging a specific notation as intermediate representation, such as software reflexion model proposed by Murphy *et al.* in [15], and Abstract Object Language (AOL) in [16]. The most closely related work is the IR based traceability links recovery. The first research using IR method to recover traceability links between software artifacts is proposed by Antonio *et al.* in [10]. They applied and compared two different IR models, probabilistic model and vector space model, to recover the traceability links between source code and free text documentation. Further, Marcus *et al.* [11] used another IR model, i.e. Latent Semantic Indexing (LSI), to conduct traceability links recovery, and the result indicates that LSI outperforms the two models applied by [10]. Both of the two works extract information, e.g. identifiers, comments, from source code thus to treat source code as a special kind of free text document. Then the traceability links are recovered by computing the textual similarity between source code and other documents using different IR model. They provide a promising starting point to recover the traceability links between software

artifacts using IR methods, though the results seems not so encouraging due to the limitation of IR technology. The difference between them and our work is that we do not simply apply IR method to recover the traceability links between source code and documentation. Using class diagrams as intermediary, we compute the relativity between different document-sections. This ensures that the relativity reflects the relationship between document sections more in the aspect of implementation, rather than textual similarity. Information in class diagrams and the structure of documents are taken into account during the computation. In addition, based on the proposed approach, we implement a tool to assist developer to read help-documents to learn software project. The document-sections related to the section being read by a developer will be recommended to the developer. Developers can also use our tool to view class diagram and source code when they read documents.

### B. Recommendation System

There is an increased interest in recommending system which can help users find books, web pages, news that they are fond of [1, 2, 3]. Regardless of the intentions of these systems, there are three main methods to make the recommendations: content-based filter [2], collaborative filter [1] and hybrid methods [3].

Software reuse and maintenance are important activities for software development, though we still face a lot of difficulties in practice. The idea of developing a tool that can help the developers carry out successful reuse and maintenance is also an important reason that leads to this work. To the best of our knowledge, there is no proceeding work which focuses on recommending the related sections in the software project help-documents to assist the developers to learn the software project. Our approach can be seen as using content-based method. But differently from the normal content-based method which only takes the item the user has read into consideration, our approach also involves the class diagrams of the software in the recommendation algorithm. It doesn't get information only from the document-section itself.

## VII. CONCLUSION

During software reuse or software maintenance, developers often face a lot of help-documents in order to learn the project. In this paper, we provided an approach to finding document-sections related with the section being read in these documents. We also developed a tool named DRA based on the approach, which can assist the developers to read the help-documents of a software project.

Our approach uses class diagrams of the software as background knowledge in the recommendation algorithm. As a result, the section being read and the sections recommended are always about the similar parts of the software project. Furthermore, the developers can be informed of which parts of the project the section recommended focuses on, so that they can decide whether to read the section. Experimental results

show that this approach can help the developers improve both efficiency and experience in studying a software project.

### ACKNOWLEDGMENT

This research was sponsored by the National Basic Research Program of China (973) under Grant No. 2009CB320703; the High-Tech Research and Development Program of China under Grant 2009AA010307; the Science Fund for Creative Research Groups of China under Grant No. 60821003, 60803010, 60803011.

### REFERENCES

- [1] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl, GroupLens: an open architecture for collaborative filtering of netnews, in Proceedings of the ACM conference on Computer Supported Cooperative Work, pp. 175-186, Oct. 1994.
- [2] R. J. Mooney and L. Roy, Content-based book recommending using learning for text categorization, in Proceedings of the Fifth ACM Conference on Digital Libraries, pp. 195-204, 2000.
- [3] Mark Claypool, Anuja Gokhale *et al.*, Combing content-based and collaborative filters in an online newspaper, ACM SIGIR'99 Workshop on Recommender Systems: Algorithms and Evaluation, Aug. 1999.
- [4] C.W. Krueger, Software reuse, in ACM Computing Surveys, vol. 24, no. 2, pp. 131-184, June 1992.
- [5] Kenton O'Hara , Abigail Sellen, A comparison of reading paper and on-line documents, in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp.335-342, March, 1997.
- [6] E. Hatcher, O. Gospodnetic, Lucene in action, Manning Publications, 2005.
- [7] Lucene, <http://lucene.apache.org/>.
- [8] EclipseUML, <http://www.omondo.com/>.
- [9] Salton, G., Wong, A., and Yang, C., A vector space model for automatic indexing, in Communications of the ACM 18, 11, pp.613-620, 1975.
- [10] G. Antoniol, G. Canfora, G. Casazza, A. D. Lucia, and E. Merlo, Recovering traceability links between code and documentation, in Transactions on Software Engineering, vol 28, No. 10, pp. 970-983, 2002
- [11] A. Marcus, J. I. Maletic, Recovering documentation-to-source-code traceability links using latent semantic indexing, in Proceedings of International Conference on Software Engineering, pp. 125-135, 2003
- [12] F.A.C. Pinhero and J.A. Goguen, An object-oriented tool for tracing requirements, IEEE Software, vol. 13, no. 2, pp. 52-64, Mar. 1996.
- [13] J. Konclin and M. Bergen, gIBIS: a hypertext tool for exploratory policy discussion, ACM Transactions on Office Information Systems, vol. 6, no. 4, pp. 303-331, Oct. 1988.
- [14] B. Ramesh and V. Dhar, Supporting systems development using knowledge captured during requirements engineering, in IEEE Transactions on Software Engineering, vol. 9, no. 2, pp. 498-510, June 1992.
- [15] G.C. Murphy, D. Notkin, and K. Sullivan, Software reflexion models: bridging the gap between source and high-level models, in Proceedings of the third ACM symposium on Foundations of Software Engineering, pp. 18-28, 1995.
- [16] G. Antoniol, B. Caprile, A. Potrich, and P. Tonella, Design-code traceability for object oriented systems, The Annals of Software Engineering, vol. 9, pp. 35-58, 2000.
- [17] A. D. Lucia, F. Fasano, R. Oliveto, and G. Tortora, Recovering traceability links in software artifact management systems using information retrieval methods, in ACM Transactions on Software Engineering and Methodology, Vol. 16, No. 13, 2007
- [18] [http://sei.pku.edu.cn/~wanglj07/DRA\\_introduction.htm](http://sei.pku.edu.cn/~wanglj07/DRA_introduction.htm)

# Validity Threats in Empirical Software Engineering Research - An Initial Survey

Robert Feldt, Ana Magazinius  
Dept. of Computer Science and Engineering  
Chalmers University of Technology  
SE-412 96 Gothenburg, Sweden  
Email: robert.feldt@chalmers.se

**Abstract**—In judging the quality of a research study it is very important to consider threats to the validity of the study and the results. This is particularly important for empirical research where there is often a multitude of possible threats. With a growing focus on empirical research methods in software engineering it is important that there is a consensus in the community on this importance, that validity analysis is done by every researcher and that there is common terminology and support on how to do and report it. Even though there are previous relevant results they have primarily focused on quantitative research methods and in particular experiments. Here we look at the existing advice and guidelines and then perform a review of 43 papers published in the ESEM conference in 2009 and analyse the validity analysis they include and which threats and strategies for overcoming them that were given by the authors. Based on this analysis we then discuss what is working well and less well in validity analysis of empirical software engineering research and present recommendations on how to better support validity analysis in the future.

**Keywords**-Validity threats, Empirical research, Software engineering, Research methodology

## I. INTRODUCTION

Empirical research in software engineering (ESE) is increasingly important to advance the state-of-the-art in a scientific manner [1]. A critical element of any empirical research study is to analyze and mitigate threats to the validity of the results. A number of summaries, models and lists of validity threats (VTs) have been presented to help a researcher in analyzing validity and mitigate threats [1, 2]. Many of these results focus on VT analysis for quantitative research such as experiments [1] while other consider VT analysis for qualitative research [2].

For a researcher in ESE it may not be clear if the existing checklists are consistent or which one applies to a particular study. In particular for qualitative ESE research this can be a problem since few studies with results specific to software engineering has been presented. Furthermore, several recent results have pointed to problems in how ESE research is conducted or reported in sub-areas [3, 4, 5]. In this paper, we want to analyse in particular how validity threats are analysed and mitigated in ESE research studies. Ultimately this will lead to additional support and guidelines for how to more consistently perform such analysis and mitigation and thus increase the validity of future ESE results.

In this study we survey how validity threats are analyzed in recent papers published in ESE and discuss the current state-of-the-art and what may have caused it. We then propose ways that can help in improving the analysis of validity threats. In particular we address the research questions:

- RQ1. What are the existing advice on and checklists for VTs in ESE?
- RQ2. What is the current practice in discussing and handling validity threats in research studies in ESE?
- RQ3. Is the current practice of sufficient quality and, if not, how can it be improved?

To limit the scope the survey presented here has been constrained to only cover one year of a major ESE publication venue. Because of this our results are only preliminary.

Section II below describes existing advice on validity threats in software engineering. In Section III we describe the methodology for the survey we have performed on VT analysis and mitigation in ESE. Section IV gives the results of the survey and analyses them. Finally, Section V discusses the VTs of this study itself and future work, while Section VI concludes.

## II. VALIDITY IN SOFTWARE ENGINEERING RESEARCH

Validity of research is concerned with the question of how the conclusions might be wrong, i.e. the relationship between conclusions and reality [2]. This is distinct from the larger question of how a piece of research might have low quality since quality has more aspects than validity alone, for example relevance and replicability. In non-technical terms validity is concerned with ‘How the results might be wrong?’ not with the larger questions of ‘How this research might be bad?’. In many cases they overlap, though.

Validity is a goal, not something that can be proven or assured with the use of specific procedures. A *validity threat* (VT) is a specific way in which you might be wrong [2]. In a *validity analysis* (VA) you identify possible threats and discuss and decide how to address them. A specific choice or action used to increase validity by addressing a specific threat we call a *mitigation strategy*.

For quantitative research in software engineering, such as experiments, specific advice on validity analysis and threats was given by Wohlin et al [1] structured according to previous

results from Cook et al [6]. Wohlin et al discuss four main types of validity threats: conclusion, internal, construct and external.

*Conclusion validity* focus on how sure we can be that the treatment we used in an experiment really is related to the actual outcome we observed. Typically this concerns if there is a statistically significant effect on the outcome.

If there is a statistically significant relationship the *Internal validity* focus on how sure we can be that the treatment actually caused the outcome. There can be other factors that have caused the outcome, factors that we do not have control over or have not measured.

*Construct validity* focus on the relation between the theory behind the experiment and the observation(s). Even if we have established that there is a casual relationship between the treatment of our experiment and the observed outcome, the treatment might not correspond to the cause we think we have controlled and altered. Similarly, the observed outcome might not correspond to the effect we think we are measuring.

Finally, the *External validity* is concerned with whether we can generalize the results outside the scope of our study. Even if we have established a statistically significant casual relation between a treatment and an outcome and they correspond to the casue and effect we set out to investigate the results are of little use if the cause and effect we have established does not hold in other situations.

For qualitative research methods there are not software engineering specific results and we have to turn to other fields of study. Epistemologically qualitative research stretches between positivism as one extreme and interpretivism as the other [7]. The two similar notions of subtle realism and anti-realism have also been used by some authors [8].

The fundamental difference between the two resides in the interepretivists belief that the humans differ from the subjects/objects of study in the natural sciences since social reality has a meaning for humans. That meaning is the basis both for human actions and the meanings that humans attribute to their own actions as well as the actions of others [7]. Positivists on the other hand view humans as any other data source that can be sensed, measured and positively verified [7].

Both positivistic and interpretivistic scientists are interested in assessing whether they are observing, measuring or identifying what they think and say they are [9]. However, since their research questions, methods and views on reality differ, so do the methods to assess the quality of their work. Positivists usually use natural sciences criteria: internal and external validity, reliability and objectivity [7]. The interpretivists use a set of alternative criteria: credibility, transferability, dependability and confirmability [10]. No matter the criteria, there are threats to the validity of the research. Lincoln and Guba suggest reactivity (researchers presence might influence the setting and subjects behaviour), respondent bias (where subjects knowingly or unknowingly

provide invalid information) and researcher bias (assumptions and personal beliefs of the researcher might affect the study) as main threats. Maxwell focuses on the researcher and lists description (of the collected data), interpretation (of the collected data) and theory (not considering alternative explanations) as the main threats [2].

Table I below summarizes the main validity threats that have been discussed above.

### III. REVIEW METHODOLOGY

Below we describe how we selected the papers to be included in the review, the overall process used and the forms used in the analysis of each paper.

#### A. Selection of papers

To limit the scope of our initial review we wanted to select only recent papers published in ESE. This is also motivated by the fact that ESE has been maturing and if we would include too many years back these papers might not be representative of current practice. Based on this we selected all full research papers published in the Empirical Software Engineering and Measurement (ESEM) conference in 2009. Even though this is a limited selection our assumption is that ESEM is the state-of-the-art conference on empirical software engineering research and as such can be expected to have higher expectations, standards and experience of empirical research.

A total of 43 full papers was published in ESEM 2009 and are thus included in our results and analysis below.

#### B. Process for Review

For each of the papers we read and filled in one form summarizing the type of and empirical content of the paper (Paper form). For each validity threat or issue discussed in the paper we then filled in a form with detailed information about each threat and any mitigation strategies mentioned for it (VT form). The forms used are presented in Tables II and III respectively and described in more detail below.

The forms themselves were created in discussions between the researchers based on the review of the existing literature on validity threats and analysis presented above. These initial forms were then piloted on a random selection of 4 papers which was reviewed independently by both authors. The pilot test did not uncover any discrepancies between the analyses of the two authors. It only prompted clarification of where to record what and the exclusion of a redundant question. Since we used the same papers in this step this also validated that we had the same interpretation of the forms and used them in the same way. We also discarded the idea of classifying each identified threat since there is no clear unified model for how this can be done.

The remaining 39 papers was then reviewed by one of the authors and the corresponding forms filled in. During this

Table I: Main types of Validity threats discussed in the literature.

Validity threat type	Example of typical questions to be answered
Conclusion validity	Does the treatment/change we introduced have a statistically significant effect on the outcome we measure?
Internal validity	Did the treatment/change we introduced cause the effect on the outcome? Can other factors also have had an effect?
Construct validity	Does the treatment correspond to the actual cause we are interested in? Does the outcome correspond to the effect we are interested in?
External validity, Transferability	Is the cause and effect relationship we have shown valid in other situations? Can we generalize our results? Do the results apply in other contexts?
Credibility	Are we confident that the findings are true? Why?
Dependability	Are the findings consistent? Can they be repeated?
Confirmability	Are the findings shaped by the respondents and not by the researcher?

review work, whenever some aspect of a paper were unclear the paper were discussed jointly by the authors.

In total this resulted in 43 Paper forms and a total of 136 VT Issue forms filled out. One paper is excluded from our analysis since it was a theoretical paper without any empirical content or any analysis of validity. Descriptive statistics, aggregation and analysis of the forms was then done cooperatively between the researchers.

### C. Paper form

Table II presents the paper form that was filled out for each of the reviewed papers.

The questions on the form aimed to characterize the overall characteristics of the paper (Questions numbered 1-3), the research methods used (Q4-5), the type and amount of empirical material (Q6) and additional comments (Q7). For each of the VTs identified we then filled in a VT form detailed below. We also noted a derived measure based on whether there was zero (noted value ‘No’) or at least one (‘Yes’) VT forms filled out for each paper. All the sheets and the answers to each question were collected, sorted per paper, in an excel sheet for further analysis.

### D. Validity threat form

For each validity issue discussed in one of the papers we used the form summarized in Table III to describe it.

The first two questions on the form (Q1-2) asked the reviewer to describe the validity threat and if the authors gave the threat any ‘formal’ name according to Table II above. Then two questions (Q3-4) were filled out for each mitigation strategy associated with a validity threat. One questions focused on describing the mitigation strategy itself (in the authors own words) while the other asked the reviewer to estimate which part of the study had been affected by the mitigation. The included parts to be judged were the design, the collected data or the analysis of the results. If the mitigation strategy had not actually been used but was just noted as a possible future strategy it was marked as affecting the ‘Future Work’.

The total time needed for reviewing each paper was also noted in the excel sheet that collected all information.

Table II: Summary of ‘Paper form’ with questions for each analysed paper.

Question	Alternatives
1. Software Engineering Perspective?	Business, Architecture/Technology, Process, Organisation, <i>Other</i>
2. Software Engineering Areas?	Requirements Engineering, Design or Architecture, Implementation, Testing or V&V, Management, Metrics or Measurement, People issues or Human factors, Economics, Methodology, <i>Other</i>
3. Sub-area(s) within main area?	<i>List main keywords in order of decreasing importance</i>
4. Type of methodology or outlook?	Quantitative, Qualitative, Both, Unclear
5. Research Methodology(ies)?	Controlled Experiment, Experiment, Survey, Case study (comparative), Case study (exploratory), Multiple Case Study, Design of Solution, Improvement of Solution, Design of Education, Design of Methodology, Observation study, Grounded Theory Study
6. Data sources and collection methods?	<i>Describe type as well as number of subjects/groups/companies etc</i>
7. Comments?	<i>Free form for any important additional comments</i>

Table III: Summary of ‘Validity threat form’ with questions for each validity issue.

Question	Alternatives
1. Summary of validity threat?	<i>Summarize based on how threat is described in the paper</i>
2. Classification or naming of threat by author?	<i>Note according to validity threats listed in Table I</i>
3. Mitigation strategy for threat?	<i>Summarize strategy based on how described in the paper</i>
4. Mitigation strategy affected which part of study?	Design of study, Implementation / Collected data, Analysis of results, Future work

Table IV: Num. of papers for different types of research methodology.

Type of Method	Number	Percentage
Quantitative	30	71.4%
Qualitative	4	9.5%
Both	7	16.7%
Unclear	1	2.4%
<b>TOTAL</b>	<b>42</b>	<b>100%</b>

#### IV. RESULTS AND ANALYSIS

##### A. Summary of quantitative data

From the 42 paper forms included in the analysis only one focused on the Business perspective, 11 on the Architecture/Technology perspective, 28 on the Process perspective and four on the Organisational perspective<sup>1</sup>. One paper was focused on Education.

Table IV shows the distribution of papers for the different types of research methodology (Q4 on 'Paper form'). Unfortunately the distribution is very skewed and have too few Qualitative papers for us to be able to compare the two different methodologies.

Table V summarizes the number of papers and their average number of validity threats for each research methodology employed. Overall 34 papers (79.1%) discussed at least one validity threat while 9 papers (20.9%) lacked any such discussion. We can see that interestingly enough there seems to be a difference in the number of validity threats that are discussed in quantitative studies compared to qualitative ditto. If we consider the experiments, controlled experiments and surveys as quantitative methods and consider case studies, multitude case studies, observation study and action research as qualitative (since the latter used grounded theory it seems plausible), the former group considers an average of 5.46 validity threats while the latter considers 3.23. Since we cannot assume the number of validity threats follow a normal distribution and since they are few, we compared these differences in mean with the Wilcoxon rank sum non-parametric statistical test. The difference between them are statistically significant at the  $\alpha = 0.05$  level ( $p = 0.029$ ).

Figure 1 shows a boxplot of the number of validity threats for the papers employing qualitative or quantitative research methods, respectively. We can see that the difference is quite clear as confirmed by the statistical test.

The number of mitigation strategies (69) was much lower than the number of validity threats (136) with an average of only 1.60 mitigation strategies per paper and 0.49 per validity threat. So on average only for about half of the validity threats that were discussed did the paper authors also discuss a way to overcome the threat.

Table VI lists the number of mitigation strategies deemed to affect a certain phase of a study. We note that for 5 of the

<sup>1</sup>Note that a few papers had more than one perspective

Table V: Num. of papers for different research methodologies.

Method	Num. papers	Percentage	Avg. VTs
Experiment	9	21.4%	5.22
Design of Solution	8	19.0%	1.38
Case studies	7	16.7%	4.14
Multiple case studies	4	9.5%	2.25
Design of Improvement	4	9.5%	1.25
Survey	3	7.1%	4.33
Design of Methodology	3	7.1%	1.67
Controlled experiment	1	2.4%	11
Design of education	1	2.4%	3
Observation study	1	2.4%	2
Grounded theory	1	2.4%	2
<b>TOTAL</b>	<b>42</b>	<b>100%</b>	<b>3.26</b>

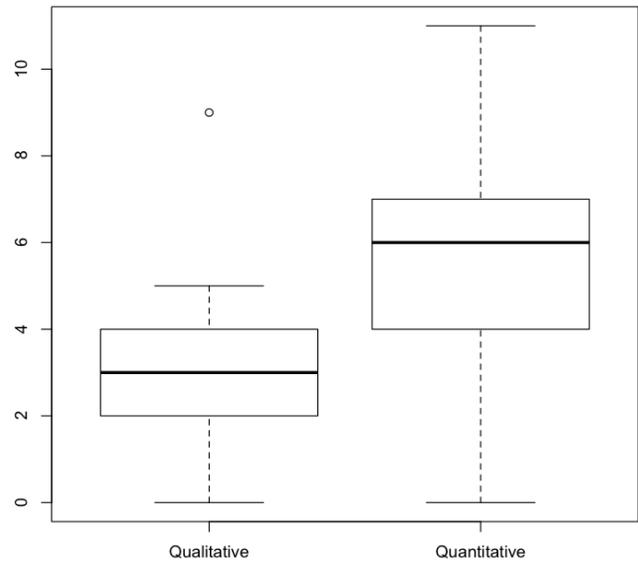


Figure 1: Boxplot of num. of validity threats for different research methods

69 identified strategies it was not possible to judge which phase they affected. For the rest there is no clear trend, other than the fact that many mitigation strategies (26.5%) even if they are discussed are just mentioned as future work and have not affected the results in the reviewed study at all.

Table VI: Num. of mitigation strategies that affects each phase.

Affected phase	Num. strategies	Percentage
Design	20	31.2%
Future Work	17	26.5%
Analysis	14	21.9%
Implementation / Data collection	13	20.3%
<b>TOTAL</b>	<b>64</b>	<b>100%</b>

### B. Analysis of validity threats in reviewed papers

Almost no authors had classified their threats based on any of the standard methods or names as presented in Section II. The only widespread name was ‘Generalizability’ which is the same as External validity.

A fairly common threat was the questioning of the ‘quality of the raw data’, ‘limited sample size’ and ‘convenience sampling’. Quite common was also the use of different types of biases to describe threats. Example of such biases that were mentioned are the ‘researcher intervention bias’, ‘mono-operation bias’, ‘mono-method bias’, ‘researcher bias’, and ‘vested interest might cause bias’.

Overall it was not clear how to map many of the discussed threats into the formal nomenclature. Most authors used their own naming and did not try to relate to the published literature with lists of threats to be checked. Many authors seemed to consider the validity threats mostly as a post-research walkthrough of limitations with only limited actual effect on the study. However, we need to study the data in more detail, and find a unified framework in which to analyse the many different threats stated before we can investigate this in more detail.

We also investigated how easy it was to find the validity discussion in the reviewed papers. Only 25 of the investigated 43 papers had an explicitly named section discussing validity threats.

## V. DISCUSSION

Overall more than 20% of the reviewed papers lacked any analysis of validity threats. This is an alarmingly high figure. Empirical researchers should know that there is always several threats to the validity of any research study and they should be examined both before and while designing the study as well as throughout the other phases of a study.

The average number of validity threats discussed was 3.26 overall, but higher for quantitative studies (5.46) and slightly lower for qualitative studies (3.23) and the lowest for studies that designed a new method, approach or solution. We think this can be explained by the fact that the existing guidelines for validity analysis of software engineering research focus on experiments. It thus becomes more likely that researchers doing quantitative research in the area would eventually get reviewer comments or take preparatory courses etc where they are introduced to these guidelines. The same is not as likely for qualitative researchers which would have to go outside of the software engineering field to find relevant guidelines and support.

It is apparent from our study that the existing terminology for analysing validity threats is not used. Only in a few cases was the terms Internal, Conclusion, Construct or External validity used, even in experiments and other quantitative studies. We propose that one explanation can be the fact that these terms are not more directly linked to the actual elements of the studies, what has been done etc. Rather

many researchers seem to describe threats based on their within-study terminology and understanding. The current terminology is more abstract which may limit its uptake and use and thus the level of support it gives.

To remedy this situation we proposed that a new and simpler terminology is introduced that is more directly linked to the actual elements of the study and is adapted to the type of method, data source and analysis method employed. Ideally such a framework should support both quantitative and qualitative methods. We see little reason that this should not be possible as long as variation is allowed along the dimensions discussed above. As a starting point a common and abstract process model of most research studies can be used along the lines presented for case studies in [11] or for experiments in [1]. These two models both have five steps with the same meaning and only slightly varying names.

That the number of mitigation strategies is low compared to the number of threats might seem alarming. However, many threats are stated just as limitations for which there is no way the author could have mitigated the it. A typical example is that the sample size is too small so the results might not be statistically significant. However, the reason often seem to be one of convenience sampling, i.e. the researchers interviewed or performed experiments on the subjects that were available in the studied organisation; they could not get access to any additional subjects.

There also seem to be some room for more extensive validity analysis. For the 25 reviewed papers that had a validity discussion they covered 5.44 threats on average per paper. Since there are typically several threats to each part and phase of a study this number seems low and it seems that there may be a lot of omission errors. However, it can also be the case that because of length restrictions the researchers have considered more threats but only includes the most severe ones in the actual paper. For these reasons it might be useful to have more complete validity analysis available on home pages for respective papers or similar. Standardized forms for this type of analysis and reporting might be valuable to promote this practice. However, we first need to extend the analysis in this paper to better understand the quality of validity analysis today.

### A. Validity threats

There are several validity threats to the design of this study. Our choice of venue is limited to a single venue and a single year. In extending this work we should of course include more venues and more years. This would both give more data and allow better and more detailed statistical analysis as well as allowing a broader coverage of the field of software engineering. Care should be taken to ensure that also quality studies are found and included.

During data collection we mostly used a single researcher to review each paper. Although we tried to mitigate this threat by noting any unclear issues and discuss them together there

is still a higher risk that a single reviewer can be biased and consistently extract the wrong information. For the future we hope to have the resources to have at least two reviewers for each paper. The pilot with co-review of four papers should have helped to create a common understanding though.

Another threat to the data collection is that our chosen categories did not always fit the papers, threats and/or mitigation strategies. Based on the experience from this initial survey we will update the form further to ensure consistent collection and analysis of results.

We see few threats to the numerical analysis, however there are several threats to the analysis of the threat and strategy summaries. Since we had no unified framework with which to classify the threats or strategies our analysis at this stage is only indicative; without statistics on which threats are common and not the analysis is unreliable.

## VI. CONCLUSIONS

To investigate the current state-of-practice in analysing validity threats among researchers in empirical software engineering we have reviewed all full research papers from the ESEM conference in 2009. More than 20% of these papers contains no discussion of validity threats and the ones that do discuss on average only 5.44 threats. For only half of the discussed threats are any strategy to overcome or mitigate the threat discussed by the paper authors of the reviewed papers. And more than 25% of these mitigation strategies mentioned have not been used in the studies but are just discussed as future work. Furthermore, the situation seems to be worse for qualitative studies than for quantitative, possibly because the little advice there is for validity analysis in software engineering has been presented for experiments. However, even the existing advice seem to have found little use, in particular, no common terminology seemed to be used for validity analysis; researchers use their own terms or use no specific terms at all.

We propose that a common model for the process of conducting empirical research in software engineering is created and that a simpler terminology for validity threats and analysis is adapted to this model. Specific guidelines for different research methods, data sources and collection methods can then be attached to the model and allow adaptation to the specifics of each study. Such a model would also allow for more detailed analysis of which threats are currently analysed and which have been missed. In future work we will develop such a model and framework and apply it to more research papers to validate it as well as deepen our knowledge of the state-of-the-art in validity analysis.

## REFERENCES

[1] C. Wohlin, M. Höst, P. Runeson, M. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software*

*engineering: an introduction*. Kluwer Academic Publishers, 2000.

- [2] J. Maxwell, *Qualitative research design: An interactive approach*. Sage Publications Inc., 2004.
- [3] D. Šmite, C. Wohlin, T. Gorschek, and R. Feldt, “Empirical evidence in global software engineering: a systematic review,” *Empirical Software Engineering*, vol. 15, no. 1, pp. 91–118, February 2010. [Online]. Available: <http://dx.doi.org/10.1007/s10664-009-9123-y>
- [4] S. Ali, L. Briand, H. Hemmati, and R. Panesar-Walawege, “A Systematic Review of the Application and Empirical Investigation of Search-based Test-Case Generation,” Technical Report Simula. SE. 293. Simula Research Laboratory, Norway, Tech. Rep., 2009.
- [5] K. Petersen and C. Wohlin, “Context in industrial software engineering research,” in *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*. IEEE Computer Society, 2009, pp. 401–404.
- [6] T. Cook, D. Campbell, and G. Fankhauser, *Quasi-experimentation: Design & analysis issues for field settings*. Houghton Mifflin Boston, 1979.
- [7] A. Bryman and E. Bell, *Business research methods*. Oxford University Press, USA, 2007.
- [8] N. Mays and C. Pope, “Qualitative research in health care: Assessing quality in qualitative research,” *British Medical Journal*, vol. 320, pp. 50–52, 2000.
- [9] J. Mason, *Qualitative researching*. Sage Publications Ltd, 2002.
- [10] Y. Lincoln and E. Guba, *Naturalistic inquiry*. Sage Publications Inc., 1985.
- [11] P. Runeson and M. Höst, “Guidelines for conducting and reporting case study research in software engineering,” *Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, 2009.

# A Comparative Study of Attribute Weighting Techniques for Software Defect Prediction Using Case-based Reasoning

Elham Paikari  
*Department of Electrical and  
Computer Engineering  
University of Calgary, Canada  
epaikari@ucalgary.ca*

Michael M. Richter,  
*Department of Computer  
Science  
University of Calgary, Canada  
mrichter@ucalgary.ca*

Guenter Ruhe  
*Dep. of Computer Science and  
Department of Electrical and  
Computer Engineering,  
University of Calgary, Canada  
ruhe@ucalgary.ca*

## Abstract

*Software defect prediction is an acknowledged approach used to achieve better product quality and to better utilize resources needed for that. One known method for predicting the number of defects is applying case-based reasoning (CBR). In this paper, different attribute weighting techniques for CBR-based defect prediction are analyzed. Two weighting techniques are compared with the case of applying uniform weights. The first one called SANN is based on sensitivity analysis of the impact of attributes as part of neural network analysis. The second one is multi-linear regression called MLR.*

*Evaluation of the accuracy of the overall method for applying the three different weighting techniques is done over five data sets comprising in total about 5000 modules from NASA. Two quality measures are applied: Average absolute error AAE and average relative error ARE. In addition to the variation of weighting techniques, the impact of varying the number of nearest neighbours is studied.*

*The three main results of the empirical analysis are: (i) In the majority of cases, SANN achieves the most accurate results, (ii) uniform weighting performs better than the MLR-based weighting heuristic, and (iii) there is no significant preference pattern for the different number of similar objects applied in CBR.*

**Keywords:** *Defect prediction, case-based reasoning, neural networks, sensitivity analysis, empirical evaluation.*

## 1. Introduction

Software metrics-based quality prediction models can be effective tool for identifying the number of defects of the modules. The use of such models prior to each release of the system can considerably improve

the system quality. A timely prediction of which modules need more effort to remove the defects is an important input for allocating quality assurance resources.

Method-level metrics are widely used for software defect prediction problem [1]. In our model, inputs are software metrics collected from past development process. The output is a prediction of the number of defects per module.

The evaluative studies of case-based reasoning (CBR) models have been promising. The most notably advantage of a CBR in this context is that the detailed characterization of the similar cases can help one interpret the automatic results.. CBR can be instantiated in different ways by varying its parameters. However, it is not clear which combination of parameters provides the best performance [1].

In this paper, different attribute weighting techniques for CBR-based defect prediction are proposed and empirically evaluated. Existing related work is discussed in Section 2. Based upon the problem statement made in Section 3, the CBR-based solution approach is described in Section 4. A comprehensive empirical evaluation is the content of Section 5. Finally, conclusions and an outlook to future research are given in Section 6.

## 2. Related work

Different methods and techniques are known for prediction of quality in general, and the number of defects in particular. Approaches such as genetic programming, neural networks, case-based reasoning, fuzzy logic, decision tree, Naive Bayes, and logistic regression have been applied successfully [1].

The software metric-based models [3], [4], [5] can be divided in two groups. The First group is predicting whether the given segment of the code, such as a method or module, is defected or not, which is referred

as software quality classification [6], [7]. The second group is predicting the number of possible defects in the module [8].

This paper focuses on the prediction of the number of defects. It has been argued that CBR is in a class of modeling techniques more suited to the analysis of software engineering data (compared with classical statistical techniques, such as least squares regression) [9], [10]. The CBR methodology has also been used for other problems in the software engineering field, including software cost estimation [11], software reuse [12], software design [13], effort prediction [14], and software quality [1], [15].

CBR has been applied to the quality modeling and for example, one empirical evaluation found CBR to be competitive to a discriminated analysis model [16]. Another study that considered models for predicting the number of faults found a CBR system to have superior predictive performance compared to an ordinary least squares regression model [14]. In [17] an empirical study investigates the effects of using different similarity functions, different solution algorithms, and different numbers of nearest neighbour cases on the prediction accuracy of CBR system. In [1][2] the performance of a CBR model with different parameters (distance measures, standardization techniques, use or non-use of weights and the number of nearest neighbours to use for the prediction) are evaluated. They concluded that there is no difference in prediction performance when using any combination of parameters.

The weight assignment for attributes in similarity measurement used by the CBR system play an important part in the obtained defect prediction accuracy. There are some studies about utilizing heuristics [18] or rough set analysis [14] to assign the weights for attributes when using CBR for effort prediction in software engineering. But in defect prediction, using CBR, there is a lack of employing additional techniques other than regression used in [2] and [16]. Their evaluation helps to determine which technique works well under which conditions and which one does not.

### 3. Problem statement

The fundamental setup for applying CBR is (i) to have a set of known cases from the past, (ii) to have a new case for being predicted, and (iii) trying to predict the new case by adapting a set of most similar cases from the past. Any instantiation of this CBR framework requires the specification of a number of parameters:

1. Similarity (distance) function
2. Attribute weighting technique

3. Number of similar cases used for the prediction
4. Solution algorithm (how the new case is predicted from the known old ones)
5. Accuracy measurement of prediction

In this paper, we provide an empirical investigation of the impact of incorporating different attribute weighting techniques (uniform weights, weights obtained from sensitivity analysis based on neural networks (SANN)) and weight obtained from applying multi-linear regression). In addition, the impact of varying the numbers of similar objects is studied. For all that, two measures of the accuracy of prediction are applied.

## 4. Solution approach

### 4.1. Overview

An overview of the proposed solution approach is given in Figure 1. Input data represents the attributes (software metrics) for each module. A case is a pair (problem, error prediction) for which we have a case base (a repository). The problem description uses features as e.g. *Cyclomatic Complexity* and *Lines of Code*.

The accurateness of the prediction depends on the chosen attributes and the attribute weight vector. The higher a weight, the more influence it has on the prediction. The use of weighted sums also requires independent attributes. For this reason the CBR approach has to be extended by two techniques that are concerned with removing dependent attributes and learning the weights.

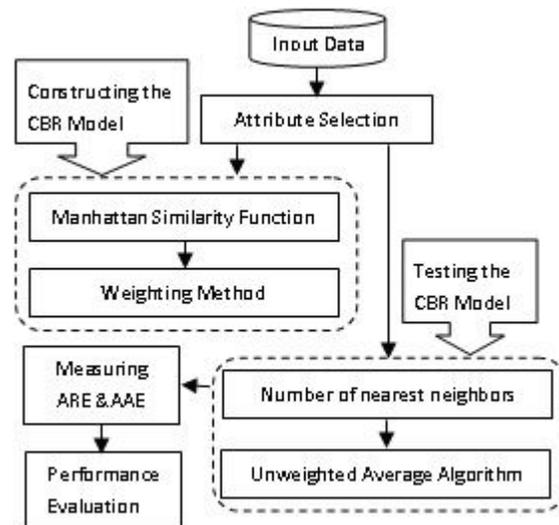


Figure 1. Proposed approach

Since the input data in each dataset has correlated values, analysis is carried out for transforming the correlated variables into uncorrelated ones. And a new dataset is obtained with fewer attributes. For the attributes selection, WEKA 3.4.14 [26] is used.

#### 4.1.1. Similarity measure

For measuring the similarity between two objects  $q$  and  $p$ , we apply the local-global principle [22]. According to that, the (global) similarity is defined as the weighted sum of local similarities related to attribute  $i$  as follows:

$$\text{sim}(q, p) = \sum w_i \text{sim}_i(q_i, p_i) \quad (1)$$

$\text{sim}_i$ , is the local measure defined on the ranges of the attributes and  $w = \{w_1, \dots, w_n\}$  is the weighting vector.

One of the most popular and straightforward distance measures for each attribute is Manhattan Distance function. The Manhattan distance between a current case,  $q$ , in the test data set, and a past case,  $p$ , in the case base, is:

$$\text{Manhattan} = \sum_i w_i |q_i - p_i| \quad (2)$$

#### 4.1.2. Weighting attributes

The question for attribute weighting is which attributes influence the prediction in which way? Some previous studies with CBR used non-equal weights [19], while others assumed equal weights [15]. A weighting method based on Multiple Linear Regression applied in [19]. For prediction of being defected or not, another weighting based on logistic regression model is applied for attribute weighting in [1].

In this study, two methods non-uniform weighting attributes were applied and compared to the uniform one. The first one uses the neural network based NeuroXL Forecaster Tool [25] and assign the weights as a result of sensitivity analysis.

The second one uses WEKA 3.4.14, and makes a multiple linear regression model to assign the weights. In this method, the weights will be the coefficients of a MLR. All attributes in the data sets,  $C_i$  and the corresponded Number of Defects, ND, would be used for finding the coefficients,  $w_i$  in MLR, given by (3).

$$ND = w_0 + w_1 c_1 + \dots + w_i c_i \quad (3)$$

The coefficients from this model are used for attribute weighting.

#### 4.1.3. Number of nearest neighbours and solution algorithm

In CBR model, considering the similarity function, the different numbers of nearest neighbours (most similar cases) may affect the accuracy. Some previous software engineering studies with CBR systems [15], [19] used the single nearest neighbour as the basis for prediction. Some other ones, such as [1], [17], evaluated different numbers of nearest neighbours.

In this paper, the effect of varying the number of nearest neighbours is investigated for all different methods of weighting.

#### 4.1.4. Solution algorithm

For the solution algorithm, used to predict the number of defects in the target module, the cases with the biggest value of similarity will be selected. A set of nearest neighbours, in CBR, is the set of these cases. And it means, these nearest neighbours are the most similar ones to the new case, and their numbers can be varied for exploring its effect on the prediction accuracy. Once *the number of nearest neighbors* is selected, a solution algorithm would be used to predict the number of defects.

There are different types of solution algorithm and here, the *unweighted average* is employed. This algorithm estimates ND by calculating the average of the number of defects of the most similar modules, (their number is equal to the number of nearest neighbours,  $n_N$ ) [17]. The predicted value is therefore given by:

$$ND = \frac{1}{n_N} \sum c_k \quad (4)$$

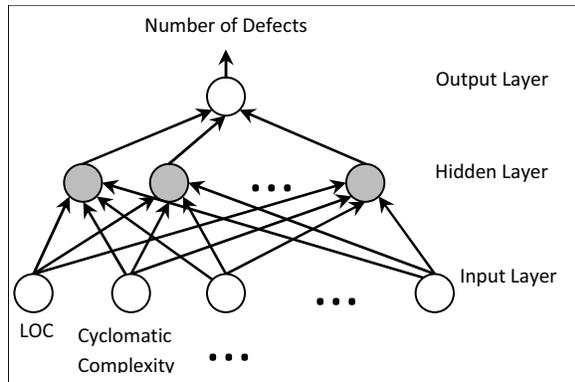
The CBR tool used in this paper is myCBR. myCBR is a case-based reasoning tool developed at the DFKI [24] as a Protégé plug-in. It is open source and developed under the GPL license.

## 4.2. Sensitivity analysis based on neural network (SANN)

A neural network (NN) consists of a network of nodes arranged in layers. A typical NN consists of three or more layers of processing nodes: an input layer that receives external inputs, one or more hidden layers, and an output layer which produces the results. There is no computation involved in the input layer. When data is presented at the input layer, the network nodes perform calculations in the successive layers until an output value is obtained at the output nodes.

The NN used in this study consists of three layers as shown in Figure 2. The input layer has 14 nodes which correspond to the 14 attributes extracted from a

software module in data sets. The output layer has one node to indicate the number of defects in the module. To build the NN, here, we have used prepared datasets for CBR.



**Figure 2. The architecture of the neural network**

Based upon the NN, sensitivity analysis is performed. The intention of this analysis is to measure the relative influence of each attribute. This analysis has three steps [20].

The sensitivity analysis will find that varying which attribute from minimum to maximum, whilst the other ones have their mean as their value, will have a greater effect on the network output. After constructing the neural network, it can be used for forecast which is needed in the sensitivity analysis to give the output value. This sensitivity analysis is deliberated in [21].

The sensitivity analysis is implemented as an algorithm here in Figure 3. In this algorithm, first, a matrix with 28 rows (two rows for each attribute) and 14 columns (one column corresponded to each attribute), will be built. In the nested loop, the value of each column would be assigned to the mean value of each corresponded attribute. The Mean function gets one attribute and returns its mean value from the data set.

In the next loop, Min and Max functions, get the one attribute and returns its minimum and maximum value from the data set. Then, first column of the first row in the matrix will have the minimum value of the first attribute, and the first column of the second row will have the maximum value of the first attribute. After that, second column will have the minimum value of the second attribute in the third row and the maximum value of the second attribute in the fourth row. And it continues for all attributes. The NN\_Forecaster function will get the values of the current row (all values of the 14 attributes on the current row) and will return the output from the neural network. The provided neural network for this analysis gets the value of all the input attributes and will give the number of

defects corresponding to the inputs, as the output. The output\_Min[j] and Output\_Max[j] contain the output from the neural network corresponding to the situation that all input attributes have their mean value, but the jth attribute has its minimum and maximum value, respectively.

The difference between the output values of the minimum and maximum of each attribute (Output\_Min[j] - Output\_Max[j]) is used as attribute weighting heuristic.

```

For i from 1 to (2*#of Attributes)
//i is the number of rows
{
  For j from 1 to (# of Attributes)
  //j is the number of columns
  {
    Value [ij] ← Mean (att[j])
    // Assign the Mean value of the existing data for the jth
    attribute to the element of the ith row and jth column of
    the matrix
  }
}
i←0 For j from 1 to (#of Attributes)
{
  i←i+1;
  Value[ij] ← Min (att[j])
  //Assign the Minimum value of the existing data for the jth
  attribute to the element of the ith row and jth column of the
  matrix

  Output_Min[j] ←NN_Forecaster(Value[i,j←1..14])
  //Assign the output of Neural Network with the value of all
  attributes of the ith row of the matrix

  i←i+1;
  Value[ij] ← Max (att[j])
  //Assign the Maximum value of the existing data for the jth
  attribute to the element of the ith row and jth column of the
  matrix

  Output_Max[j] ←NN_Forecaster (Value[i,j←1..14])
  //Assign the output of Neural Network with the value of
  all attributes of the ith row of the matrix

  Weight_NN [j] ← Output_Min [j] - Output_Max [j]
}

```

**Figure 3. SANN algorithm**

## 5. Empirical evaluation

### 5.1. Data sources and pre-processing

The MDP (**M**etrics **D**ata **P**rogram) is one of the most comprehensive Data Repositories and databases that store problem data, product data and metrics data [23].

In this study, construction, training, testing and evaluation of the CBR Model are based on 5 data sets including more than 5000 modules from NASA-MDP. The summary information of these data sets is shown in Table 1.

**Table 1. Characterization of data sets**

Data Set	# Modules	% Defective	Code	LOC
CM1	505	9.6	C	17K
KC3	458	9.39	Java	8K
PC1	1107	6.87	C	26K
PC3	1563	10.24	C	36K
PC4	1458	12.21	C	30K

## 5.2. Evaluation criteria

For each data set, the test data set that is independent of the training data set, and includes 34% percent of the whole data set, is prepared. The accuracy of the model is assessed using these data sets for cross validation.

The *Average Absolute Error (AAE)* and *Average Relative Error (ARE)* are two common quantities, used to measure the accuracy of the predictions. The AAE and ARE are given by:

$$AAE = \frac{1}{n} \sum_{i=1}^n |p_i - y_i| = \frac{1}{n} \sum_{i=1}^n |e_i| \quad (5)$$

$$ARE = \frac{1}{n} \sum_{i=1}^n \left| \frac{p_i - y_i}{y_i + 1} \right| \quad (6)$$

ARE is an average of the absolute errors  $e_i = p_i - y_i$ , where  $p_i$  is the prediction and  $y_i$  is the true value. In ARE, the absolute error is divided by the true value to give the relative error. The denominator in ARE has a '1' added to avoid division by zero as suggested in [17]. Here, n is the number of the modules in test data sets.

## 5.3. Comparison of the prediction results

Considering the same attributes for all data sets, we compare the results from the different prediction models.

In the first comparison, the results from two models are shown in Table 2. For the first model the weights of the attributes are equally assigned and for the second one the SANN is applied. In the second comparison, the results from other two models are shown in Table 3. For the first model the weights of the attributes come from the MLR method and for the second one, the SANN method is applied.

The prediction performance measurements, AAE and ARE for all five different numbers of nearest

neighbours and different weighted models, for these models in Table 2 and Table 3, doesn't show significant differences.

**Table 2. AAE and ARE for SANN and Equal Weighting with Five Different Numbers of Nearest Neighbors for Five Data Sets**

Data Sets	nn	Weighting with Sensitivity Analysis based on NN		No Weighting (Equal Weights)	
		AAE	ARE	AAE	ARE
CM1	K=1	0.259	0.119	0.265	0.127
	K=2	0.265	0.129	0.271	0.137
	K=3	0.249	0.114	0.257	0.125
	K=4	<b>0.251</b>	<b>0.115</b>	<b>0.269</b>	<b>0.133</b>
	K=5	0.258	0.121	0.267	0.131
KC3	K=1	<b>0.295</b>	<b>0.162</b>	<b>0.365</b>	<b>0.228</b>
	K=2	0.314	0.193	0.321	0.193
	K=3	0.316	0.210	0.357	0.237
	K=4	0.332	0.219	0.353	0.245
	K=5	0.322	0.212	0.342	0.233
PC1	K=1	<b>0.162</b>	<b>0.094</b>	<b>0.178</b>	<b>0.106</b>
	K=2	0.186	0.116	0.177	0.109
	K=3	0.191	0.124	0.196	0.127
	K=4	0.183	0.116	0.197	0.128
	K=5	0.184	0.118	0.197	0.128
PC3	K=1	0.145	0.084	0.158	0.097
	K=2	0.151	0.097	0.167	0.112
	K=3	0.159	0.104	0.179	0.122
	K=4	0.158	0.103	0.175	0.119
	K=5	<b>0.164</b>	<b>0.110</b>	<b>0.186</b>	<b>0.131</b>
PC4	K=1	0.167	0.167	0.196	0.196
	K=2	<b>0.155</b>	<b>0.155</b>	<b>0.182</b>	<b>0.182</b>
	K=3	0.151	0.151	0.167	0.167
	K=4	0.152	0.152	0.158	0.158
	K=5	0.151	0.151	0.156	0.156

There are two points to highlight in Table 2. First, besides the small changes, the SANN is performing better, almost always. The only case is in the PC1 data set, when we are considering two as the number of the nearest neighbours. The best result for each data set, regarding AAE and ARE, is shown by different font.

The AAE and ARE in Table 3 are showing bigger differences. Again, almost all of AAE and ARE values from SANN are smaller in comparison with AAE and ARE from MLR. The exceptions here are three cases. Two of them are in PC4 data sets, while the numbers of nearest neighbours are one and two, and one of them

is in PC1data set, while the number of nearest neighbour is three.

The point here is about the bigger differences. It means that if we compare MLR with equal weighting, in the most of the cases, equal weighting is more accurate than MLR.

**Table 3. AAE and ARE for SANN and MLR with Five Different Numbers of Nearest Neighbors for Five Data Sets**

Data Sets	n <sub>N</sub>	Weighting with Sensitivity Analysis based on NN		Weighting with Multiple Linear Regression	
		AAE	ARE	AAE	ARE
CM1	K=1	<i>0.259</i>	<i>0.119</i>	<i>0.306</i>	<i>0.182</i>
	K=2	0.265	0.129	0.285	0.157
	K=3	0.249	0.114	0.273	0.141
	K=4	0.251	0.115	0.266	0.135
	K=5	0.258	0.121	0.261	0.132
KC3	K=1	<b>0.295</b>	<b>0.162</b>	<b>0.449</b>	<b>0.303</b>
	K=2	0.314	0.193	0.433	0.305
	K=3	0.316	0.210	0.380	0.256
	K=4	0.332	0.219	0.346	0.238
	K=5	0.322	0.212	0.344	0.239
PC1	K=1	<b>0.162</b>	<b>0.094</b>	<b>0.205</b>	<b>0.139</b>
	K=2	0.186	0.116	0.198	0.133
	K=3	0.191	0.124	0.189	0.122
	K=4	0.183	0.116	0.186	0.117
	K=5	0.184	0.118	0.187	0.119
PC3	K=1	0.145	0.084	0.181	0.122
	K=2	0.151	0.097	0.187	0.132
	K=3	0.159	0.104	0.212	0.154
	K=4	<b>0.158</b>	<b>0.103</b>	<b>0.214</b>	<b>0.158</b>
	K=5	0.164	0.110	0.214	0.157
PC4	K=1	0.167	0.167	0.127	0.127
	K=2	0.155	0.155	0.148	0.148
	K=3	0.151	0.151	0.156	0.156
	K=4	0.152	0.152	0.162	0.162
	K=5	<b>0.151</b>	<b>0.151</b>	<b>0.162</b>	<b>0.162</b>

Again, the best results, regarding AAE and ARE, are shown by different fonts in Table 3. We observe that the best result can come from sizes of the set of nearest neighbours.

The best results from both Table 2 and Table 3, all show the improvement of applying SANN, in comparison with equal weighting and MLR methods

for feature weighting in this CBR model, and all of them are shown in Table 4.

**Table 4. Relative improvements in comparing attribute weighting techniques.**

Data Sets	n <sub>N</sub>	Relative improvement of SANN in comparison with unweighted		Relative improvement of SANN in comparison with MLR	
		AAE	ARE	AAE	ARE
CM1	K=4	7	16	18	53
KC3	K=1	24	41	52	57
PC1	K=1	10	13	26	48
PC3	K=5	13	19	36	54
PC4	K=2	18	18	7	7

## 6. Conclusions and future research

In this paper, an empirical evaluation of three weighting techniques used for defect prediction following case-based reasoning has been conducted. Based on more than 5000 modules included in the study, the three main findings are: (i) In the majority of cases, SANN achieves the most accurate results, (ii) uniform weighting performs better than the MLR-based weighting heuristic, and (iii) there is no significant preference pattern for the different number of similar objects applied in CBR.

While the results are promising, we are aware of a number of limitations which need to be addressed as part of future research. First of all, further data sets other than the ones coming from NASA, need to be considered to increase the external validity of the results.

A second limitation is the selection of the similarity function and the definition of prediction accuracy. While Manhattan similarity measure and the AAE and ARE are acknowledged measures, the questions remains open to compare the performance of techniques against other measures as well. This would help to increase the construct validity of the results.

As part of the sensitivity analysis in SANN, another measure for impact of attributes can be applied. Finally, other weighting heuristics could be evaluated in addition to SANN and MLR.

### ACKNOWLEDGEMENT

One of the authors would like to thank the Canadian NSERC Natural Sciences and Engineering Research Council (Discovery Grant 250343-07) for the financial support of this research.

## 7. References

- [1] C. Catal and B. Diri, "A Systematic Review of Software Fault Prediction Studies", *Expert Systems with Applications*, vol. 36, 2008, pp. 7346-7354.
- [2] K. El Emam, S. Benlarbi, N. Goel, and S. N. Rai, "Comparing Case-based Reasoning Classifiers for Predicting High Risk Software Components", *The Journal of Systems and Software*, vol. 55, 2001, pp. 301-320.
- [3] V.R. Basili, L.C. Briand, and W.L. Melo, "A Validation of Object-Oriented Design Metrics as Quality Indicators", *IEEE Transactions on Software Engineering*, vol. 22(10), 1996, pp. 751-761.
- [4] N.F. Schneidewind, "Investigation of Logistic Regression as a Discriminant of Software Quality", 7<sup>th</sup> Proceedings of the International Software Metrics Symposium, 2001, pp. 328-337.
- [5] T.M. Khoshgoftaar, E. B. Allen, and J.C. Busboom, "Modeling Software Quality: The Software Measurement Analysis and Reliability Toolkit", 12<sup>th</sup> Proceedings of the International Conference on Tools with Artificial Intelligence, 2000, pp. 54-61.
- [6] T. Menzies, J. Greenwald, and A. Frank, "Data Mining Static Code Attributes to Learn Defect Predictors", *IEEE Transactions on Software Engineering*, vol. 33 (1), 2007, pp. 2-13.
- [7] M. Shepperd and G. Kadoda, "Comparing Software Prediction Techniques Using Simulation", *IEEE Transactions on Software Engineering*, vol. 27(11), 2001, pp. 1014-1022.
- [8] M.P. Evett, T.M. Khoshgoftaar, P.D. Chien, and E.B. Allen, "GP-based Software Quality Prediction", 3<sup>rd</sup> Proceedings of the Conference of Genetic Programming, 1998, pp. 60-65.
- [9] A.R. Gray and S.G. MacDonell, "Software Metrics Data Analysis: Exploring the Relative Performance of Some Commonly Used Modeling Techniques", *Empirical Software Engineering*, vol. 4, 1999, pp. 297-316.
- [10] T.M. Khoshgoftaar and N. Seliya, "Analogy-based Practical Classification Rules for Software Quality Estimation", *Empirical Software Engineering*, vol. 8, 2003, pp. 325-350.
- [11] A. Idri, A. Abran, T.M. Khoshgoftaar, "Estimating Software Project Effort by Analogy Based on Linguistic Values", 8<sup>th</sup> Proceeding of the International Software Metrics Symposium, 2002, pp. 21-30.
- [12] C.V. Ramamoorthy, C. Chandra, S. Ishihara, and Y. Ng, "Knowledge-based Tools for Risk Assessment in Software Development and Reuse", 5<sup>th</sup> Proceedings of the International Conference on Tools with Artificial Intelligence, 1993, pp. 364-371.
- [13] B. Bartsch-Spoerl, "Toward the Integration of Case-based, Schema-based, and Model-based Reasoning for Supporting Complex Design Tasks", 1<sup>st</sup> Proceeding of the International Conference on Case-based Reasoning, 1995, pp. 145-156.
- [14] J. Li and G. Ruhe, "Software Effort Estimation by Analogy Using Attribute Selection Based on Rough Set Analysis", *International Journal of Software Engineering and Knowledge Engineering*, vol. 18 (1), 2008, pp. 1-23.
- [15] K. Ganesan, T.M. Khoshgoftaar, and E.B. Allen, "Case-based Software Quality Prediction", *International Journal of Software Engineering and Knowledge Engineering*, vol. 10(2), 2000, pp. 139-152.
- [16] T.M. Khoshgoftaar, K. Ganesan, E.B. Allen, F.D. Ross, R. Munikoti, N. Goel, and A. Nandi, "Predicting Fault-Prone Modules with Case-based Reasoning", 8<sup>th</sup> proceeding of the International Symposium on Software Reliability Engineering, 1997, pp. 27-35.
- [17] T.M. Khoshgoftaar, N. Seliya, and N. Sundaresh, "An Empirical Study of Predicting Software Faults with Case-based Reasoning", *Software Quality Journal*, vol. 14, 2006, pp. 85-111.
- [18] A. Tosun, B. Turhan, and A. B. Bener, "Feature Weighting Heuristics for Analogy-based Effort Estimation Models", *Expert Systems with Applications*, vol. 36, 2009, pp. 10325-10333.
- [19] T.M. Khoshgoftaar, K. Ganesan, E. Allen, F. Ross, R. Munikoti, N. Goel, A. Nandi, "Predicting Fault-prone Modules with Case-based Reasoning", 8<sup>th</sup> Proceedings of the International Symposium on Software Reliability Engineering, 1997, pp. 27-35.
- [20] Larose D.T., "Discovering Knowledge in Data; an Introduction to Data Mining", John Wiley & Sons, New Jersey, USA, 2005.
- [21] Paikari E., "Analogy Based Defect Prediction Model, Software Engineering Decision Support, Technical Report 086/2009, December 9, 2009, available at <http://people.ucalgary.ca/~epaikari>.
- [22] M. M. Richter, "Similarity in Case-Based Reasoning for Signals and Imaging", ed. Petra Perner. Springer Verlag, 2007.
- [23] Metrics Data Program, NASA Independent Verification and Validation facility, <http://mdp.ivv.nasa.gov>, last accessed on 12/03/2010.
- [24] Available at: <http://mycbr-project.net/index.html>, Last Accessed on 12/03/2010.
- [25] Available at: <http://www.neuroxl.com>, Last Accessed on 12/03/2010.
- [26] Available at: <http://www.cs.waikato.ac.nz/ml/weka>, Last Accessed on 12/03/2010.

# Software Project Portfolio Selection

## A Modern Portfolio Theory Based Technique

Hélio R. Costa  
COPPE / UFRJ  
Caixa Postal 68511  
Rio de Janeiro, RJ Brazil

Márcio O. Barros  
PPGI / UNIRIO  
Av. Pasteur, 458  
Rio de Janeiro, RJ Brazil

Ana Regina Rocha  
COPPE/UFRJ  
Caixa Postal 68511  
Rio de Janeiro, RJ Brazil

**Abstract** – The task of choosing which projects should compose a portfolio is not easy, especially because of the uncertainties involved in this decision. Software development organizations face this problem more intensely, since technologies and project requirements are constantly changing, clients are always demanding more quality and requirements are often being changed. In this paper we present a technique to create a balanced software project portfolio based on the Modern Portfolio Theory, Risk Management, Project Viability, and Monte Carlo Simulation. The approach is supported by the dynamical analysis of the intrinsic correlation among the candidate projects, what allows the study of the influence of the projects on the portfolio.

**Key Words:** Project Portfolio Selection, Portfolio Balance, Project Correlations.

### I. INTRODUCTION

Many authors [3] [17] have introduced the notion that software development is a value-driven activity. In this sense, the construction of software systems can be analyzed in the same way as many other investment activities. Thus, there must be clear justifications for using the available resources in projects, based on their expected returns and risks [6].

Project Portfolio Management (PPM) has gained attention in recent years as organizations have become increasingly project, program, and portfolio oriented [11]. Levine [12] defines PPM as the management of the project portfolio, aiming to maximize the contribution of projects to the overall welfare and success of the enterprise. The concept of portfolio emerged in the early 1950s, when the term was coined as a group of assets held by an investor [14].

Cooper et al [5] outline three main goals for PPM: (i) maximize the value of the portfolio; (ii) select the right balance of projects; and (iii) link the project portfolio to the business strategy. In their extensive research, and according to a set of metrics created to evaluate the achievements of enterprises adopting PPM, the authors have found that selecting the best projects available to meet portfolio balance is the least precisely defined goal and the one with the lowest performance.

Suppose a software development organization facing the following typical situations: (i) a certain amount of money is available to be invested in a set of projects, though the

resources at disposal are not sufficient for all of them; (ii) a project portfolio is already being developed and one or more projects are about to be chosen to take part of the portfolio; and (iii) one or more projects belonging to an ongoing portfolio are not performing as planned and the organization is trying to find out if it is valid to keep, cancel or reprioritize them. Assuming that the goal of such an organization is to maximize the value of its project portfolio, we address the question of which projects should compose its portfolio.

A great number of approaches used to select and balance project portfolios can be observed in the systematic reviews presented by [5], [8] and [11]. These approaches can be classified into three main categories: (i) scoring and ranking; (ii) mapping models; and (iii) financial methods [4]. According to the authors, financial methods are the most commonly used approaches and more suitable to deal with the project selection problem, because methods from the two first categories are usually based on subjective and independent assessment of the projects.

Among the financial methods, the Discounted Cash Flow Models, Return on Investment, Internal Rate of Return, and Payback Analysis are the first alternatives for most organizations, since they are relatively easy to understand and execute. More complex approaches, such as Productivity Index, Expected Commercial Value, Expected Monetary Value, and Real Options are also encountered, but their acceptance in organization is limited.

Since a project selection process will compose a portfolio from which the revenues of a company will be derived, one important aspect to be addressed by such a process is the interdependencies among the available projects. This issue is not directly addressed by the former techniques. Thus, in this paper we propose a project selection technique that is based on the Modern Portfolio Theory (MPT) [14], a financial approach that takes into account not only the joint expected returns, but also the risks incurred by the assets, and their interdependencies.

Besides this introduction, this paper is organized in five more sections. Next, we introduce the basic concepts of Modern Portfolio Theory. The following section discusses the differences between assets on financial domain and projects. On Section V we present our approach. Examples

of the approach are displayed on section VI. Conclusions are made at the last section

## II. MODERN PORTFOLIO THEORY

MPT is a disciplined approach to support decisions related to portfolio investments. In MPT, a portfolio is a weighted combination of assets, where the weight of each asset is proportional to the amount of capital invested in that asset. The purpose of MPT is to define in which assets and in which proportion an investor should allocate his capital in order to maximize his return and minimize his risk.

The *Expected Return of the Portfolio* ( $ER_p$ ) is the weighted return of its assets. Supposing a portfolio composed only by assets  $I$  and  $J$ ,  $R_p$  is represented by (1).

$$ER_p = w \cdot \mu_I + (1 - w) \cdot \mu_J \quad (1)$$

where  $w$  is the percentage of money invested in asset  $I$ ,  $(1 - w)$  is the percentage of money invested in asset  $J$ .  $\mu_I$  and  $\mu_J$  are the average returns of assets  $I$ , and  $J$ , respectively.

The Risk of the Portfolio ( $\sigma_p$ ) is a function of the risks of the assets presented in the portfolio, the weight of each asset, and the interdependencies among them. The risks of the assets ( $\sigma$ ) are calculated by the standard deviation of their observed returns overtime. The weights are the proportion of investment in each asset, and the interdependencies are calculated by means of the pair-wise correlation ( $\rho$ ) among them, taking the historical returns as the basis for this calculation. It is important to highlight that in MPT, the correlations shows how an asset affects and is affected by the presence of other asset in a given portfolio.

The correlation is a measure of the strength and direction of the relationship between two assets. It is a number in the  $[-1, +1]$  interval, where  $-1$  represents two assets that move in opposite ways with the same strength, while  $+1$  represents two assets that tend to move in the same direction with same strength.

Given the weights, the correlations, and the risks of its assets, the risk of a portfolio composed by two assets ( $I$  and  $J$ ) is calculated by (2).

$$\sigma_p^2 = w^2 \cdot \sigma_I^2 + (1-w)^2 \cdot \sigma_J^2 + 2 \cdot w \cdot (1-w) \cdot \rho_{IJ} \cdot \sigma_I \cdot \sigma_J \quad (2)$$

Given a set of portfolios, their Expected Returns ( $ER_p$ ), and their risks ( $\sigma_p$ ), an Efficient Frontier (EF) can be drawn. The EF is formed by the uppermost points of a cloud plotted in a chart that presents risk on its horizontal axis, and expected return on its vertical axis. Each point (risk x return) represents a portfolio formed by combinations of candidate projects. A typical EF can be observed on Fig. 1.

By indicating how much risk the investor is willing to accept, the frontier shows the portfolio with the greatest expected return. On the other hand, for a given expected return, there is only one portfolio with the minimum risk. Thus, choosing portfolios on the frontier is the rational decision for an investor.

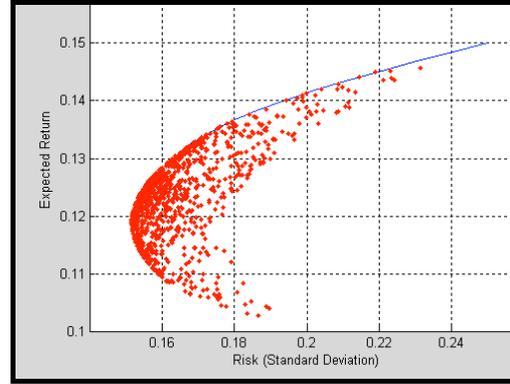


Fig. 1. Typical Efficient Frontier.

The composition of these optimum portfolios embeds what MPT calls diversification, which is the combination of assets negatively correlated. This artifice allows the creation of less risky portfolios, because a negative impact on an asset is compensated by a positive impact on another asset, since they tend to move in opposite directions.

## III. FINANCIAL ASSETS AND PROJECTS

If MPT is applied outside of its traditional financial context, some differences must be considered [4] and [10].

The first difference is that assets of financial portfolios are usually liquid, in the sense that they can be bought or sold at an investor's discretion.

On the other hand, projects composing a project portfolio that have already been initiated cannot always be interrupted at any time, at the discretion of those investing in the project. Two situations arise in a project portfolio context: projects performing as planned should be kept at the portfolio, while projects with low performance might be re-evaluated. The latter can either (i) be abandoned (if possible) accepting the injuries and allocating the resources made available on other projects; (ii) be kept in the portfolio (if the project cannot be interrupted), considering the money formerly invested as sunk cost and reassessing the remaining cash flows provided by the project as a new investment.

Another difference is that the proportion of capital invested in each financial asset is determined by the investor's will, and this proportion can be changed at any time. Conversely, in a project context, the proportion is defined by the amount of money necessary to develop the projects and, most of the times, this proportion cannot be changed.

We do not consider that a restriction to adapt MPT to the project context, because the problem addressed will be simplified to the selection of the assets (projects), since the proportion of the investment is already defined.

Finally, financial assets often have available data about their formerly observed returns to be analyzed by investors. Information required to select the best portfolio using MPT (expected returns, risks, and correlations) can be calculated

from past data. Projects, for their turn, are unique by definition and historical series of past returns are not available. Thus, their expected returns, risks, and the interdependencies must be addressed in other ways. Our project selection technique collects this information from a systematic analysis of common risks incurred by candidate projects (see section IV).

Despite of many differences, MPT has been applied to different knowledge areas, such as Oil and Gas [1], Pharmaceutical Industry [2], and R&D companies [15]. Each of these approaches applied the general MPT model with a set of constraints that would not be commonly used in financial portfolios [10]. In IT project portfolios, MPT was first suggested by [13] and, since then, other approaches such as [9] can be found in literature. However, a measure of interdependency among projects is still an open issue.

#### IV. THE PROJECT SELECTION APPROACH

The basic input information for our software project portfolio selection process is a group of candidate projects prone to take part in a portfolio. For each candidate project, there must exist information about its costs, expected return, and the risks (probability of occurrence, impact on project, possible responses and their correlated costs) faced by the project under analysis.

The approach is composed by eight steps: (i) calculate the NPV of all candidate projects; (ii) perform risk management for all candidate projects; (iii) calculate the risks scenarios; (iv) calculate the expected values for the projects under such scenarios; (v) calculate the correlations among the projects; (vi) mount all possible portfolios composed by subsets of the pre-selected projects; (vii) calculate the variables necessary to determine the best portfolio; and (viii) plot the Efficient Frontier.

##### A. Calculate the Net Present Value

The Net Present Value (NPV) of a project is defined as the difference between the sum of the discounted cash flows that will be generated by the project and the amount of money that was invested. So, the NPV is a number that expresses how much value an investment will result in. Detailed procedures to calculate the NPV of a project can be found in [16].

One advantage of choosing this technique in detriment of other traditional project valuation methods, such as ROI, IRR or Payback, is that it considers the cash flow of a project in a time horizon and corrects the value of the money overtime through a given discount rate. Another advantage is that, regardless the duration of the projects, they can all be compared, since all their cash flows are brought to the present moment.

On the other hand, the usage of NPV as the only source of decision support information may lead to incorrect decisions. Used in isolation, this model cannot account for uncertainties that may affect future cash flows [7]. This is a typical situation faced by software development

organizations, due to the risks incurred by the projects. Thus, risks must be addressed, which is the second step of our approach.

##### B. Perform Risk Management

Based on the NPV (Expected Return) of each candidate project under analysis, our Risk Management approach is composed by the following steps: (i) identify the risks (threats and opportunities) that may affect the projects; (ii) estimate their probability of occurrence and impact (positive or negative) on the projects; (iii) plan responses for these risks; (iv) recalculate the new probabilities of occurrence and/or impacts of the risks after the responses; and (v) recalculate the Expected Returns of the projects due to the eventual costs involved on the risk response plan and the changes on the risk exposures.

In such risk management process, the risks that may affect more than one project must receive special attention. This is necessary to verify how distinct projects behave while facing the same uncertainties. Such behavior, observed in the following step of our proposed technique, provides the necessary data to estimate the interdependency between the projects. Risks that affect a single project will not provide much insight on project dependencies. In software projects, examples of such generic risks include creeping user requirements, effects of new technologies, human resources issues, inadequate cost estimation, lack of support from high management, and low productivity.

##### C. Creation of Risk Scenarios

Based on the  $n$  risks that were identified on the previous step, we create all possible combinations of risks that can occur, leading to  $2^n$  risk scenarios. The scenarios represent a group of different uncertain condition that the projects may be exposed to, and may vary from no risk to all the risks happening at the same time. Depending on the risk scenario that may occur, the projects will be affected in a different manner. We assume that the risks faced by the project are independent. Thus, the probability of occurrence for each scenario is given by the product of the probability of occurrence of each risk that compose the scenario times one minus the probability of occurrence of each risk that does not take part of the scenario. Table I shows an example of the possible risks scenarios that can be formed by five risks.

TABLE I  
RISK SCENARIOS

Scenario Number	Risks				
	R1	R2	R3	R4	R5
#1	0	0	0	0	0
#2	1	0	0	0	0
#3	1	1	0	0	0
#4	1	1	1	0	0
.	.	.	.	.	.
#32	1	1	1	1	1

#### D. Series of Expected Values

This step refers to the creation of a series of expected values for all projects. The goal is to generate historical data that enables the calculation of the risks and the correlations among the projects, which are two of the missing variables not encountered on portfolio balance approaches encountered on the literature.

Based on the Expected Returns of the projects obtained after the first two steps of our approach, as well as the probabilities of occurrence of the scenarios, it is possible to calculate a series of Expected Returns of the projects by means of the following steps: (i) through a Monte Carlo Simulation process, one risk scenario is randomly drawn, and each candidate project is analyzed in relation to the risk scenario; (ii) if the project is affected by one or more risks belonging to the scenario, the impacts are added (positive impact) or subtracted (negative impact) to the Expected Return of the project, generating a new Expected Return; (iii) projects that are not affected by the scenario must maintain their Expected Return in the current round; and (iv) a number  $n$  of scenarios are drawn, in order to generate the series of expected values of the projects.

A comparison with the stock market can be made to justify this procedure. Given a set of financial assets, a certain number of risk scenarios that may affect them, and a time horizon, the occurrence of a scenario in a day, for example, will affect the prices of the assets subjected to this risk in a certain way and will not cause any effect to the assets that are not exposed to risks of this scenario. As the days, months and year passes, a series of values of is formed.

But in projects, the perspective is different, and it is not possible to get historical information, since, by definition, a project is a unique event. Therefore, the Monte Carlo Simulation replaces the days, months or years, generating the series of Expected Returns overtime, accordingly to the risk scenarios to which a project might be subjected.

#### E. Correlations

This is the central point and the differential of our approach. According to our research and the systematic reviews achieved by [5], [8] and [11] no technique presented, so far to balance portfolios calculates the correlations among projects in a formal manner. Only adaptations, guessing or estimations are made.

Some of these artifices used to balance portfolios involve the selection of short and long term projects, different categories of projects or projects placed in distinct quadrants on a bubble diagram. All these attempts, in a certain way, try to diversify investments in projects, which is the basis of the MPT. But, the problem concerning these approaches is that MPT diversifies investments by choosing assets based on quantitative interdependencies among them, while the former approaches handle interdependencies in a subjective fashion.

Subjective evaluations may carry the bias of opinion and personal preferences, usually having less value to decision makers than evaluations carried upon a formal, systematic approach. Therefore, if correlations between assets cannot be calculated, the use of the concepts of MPT cannot be properly applied and an efficient diversification or balance among projects cannot be achieved.

Based on the series of expected values, the correlations are calculated with the aid of any statistical package or even a spreadsheet. Aiming to reach reliable results, we suggest at least 30 expected values of each project.

#### F. Possible Portfolios

On this step, a matrix of all possible portfolios that can be formed by the prospective projects is created. Given a number  $n$  of candidate projects,  $2^n$  possible portfolios can be formed. This step is important to have an idea of all possibilities that can be taken by a decision maker.

Each portfolio created will have a risk level and an expected return. Thus, the core question of the problem to be solved is precisely the choice of which portfolio must be selected.

This must be made, finding the portfolio that best fits the organization's resources and maximizes the risk x return relation.

#### G. Variables of the Portfolios

Some variables are necessary to choose the best portfolio. The first one is the cost necessary to implement each portfolio ( $C_p$ ), which is represented by the sum of the costs to implement all projects composing a given portfolio.

The second variable is the Expected Return ( $ER_p$ ) of the portfolio, which is calculated, by the sum of the expected return of all the projects of a given portfolio.

Based on the series of Expected Returns of the projects, the risk of each project can be calculated. The risks are calculated by the standard deviation of the series of Expected Returns. According to [14] the standard deviation of an asset describes its risk level, since it demonstrates the variability around the Expected Returns average.

By means of the variables generated so far, it is possible to calculate the risk of the portfolios. According to [14] this is made by the (2). For the sake of exemplification, on (3), we will demonstrate the risk of a portfolio composed only by projects  $I$  and  $J$ .

$$\sigma_{IJ}^2 = w_I^2 \sigma_I^2 + w_J^2 \sigma_J^2 + 2.w_I w_J \rho_{IJ} \sigma_I \sigma_J \quad (3)$$

The fourth variable that can be obtained is an index to determine the relation between the return of the portfolio and its risk level. This index demonstrates the amount of return of a portfolio per risk unit, what is a theoretical measure of portfolio efficiency. The greater the value for this index, the more efficient is the portfolio. This is the essence of the portfolio balance, i.e., join a group of assets (projects) that maximize the returns and minimize the risk.

We name this index as *Portfolio Efficiency Index (PEI<sub>p</sub>)*, which is represented by (4).

$$PEI_p = ER_p / \sigma_p^2 \quad (4)$$

#### H. Efficient Frontier

The final step of our approach is the formation of the Efficient Frontier, that is, the chart plotting the tuple  $ER_p$  and  $\sigma_p^2$  of each portfolio. Thus, it is possible to visualize portfolios which are suitable to be chosen, i.e., only those composing the frontier, since they represent the best risk return relation. Portfolios represented by points below the frontier represent suboptimal project portfolios, since they provide an intermediate expected return for a given level of risk. Thus, a rational investor should never choose a portfolio below the frontier.

#### V. EXAMPLES OF THE APPROACH

Suppose a software development organization trying to decide in which projects (A, B, C, D, and E) it should invest. Table II shows the five candidate projects and their series of Expected Values, calculated under 30 different risk scenarios. Due to space restrictions, only the three first the last scenario values are displayed. The three rows on the bottom show the Average Expected Return of each project, their Risks (Std Dev) and the costs needed to implement them. It can be observed that, to implement all projects, it would be required \$196,000, but let us suppose the organization has only \$160,000.

TABLE II  
ERs SERIES

Rounds	Projects ERs				
	A	B	C	D	E
1	10,000	25,000	30,000	6,800	15,000
2	12,000	15,000	25,000	9,000	17,000
3	8,000	20,000	31,000	5,800	18,000
.	.	.	.	.	.
30	14,000	18,000	28,000	9,000	19,500
AVG ER	9,566	20,416	28,211	9,343	24,166
Std Dev	2,836	3,058	2,427	2,460	4,504
Cost	26,000	30,000	50,000	50,000	40,000

After calculating the correlation among the projects a matrix like Table III will be formed. It is possible to observe that some projects are positively correlated, while others are negatively correlated. The correlations represents the interdependency among projects.

TABLE III  
CORRELATIONS

	A	B	C	D	E
A	1.000	0.308	-0.017	-0.255	0.226
B	0.308	1.000	-0.011	0.130	0.029
C	-0.017	-0.011	1.000	0.135	-0.096
D	-0.255	0.130	0.135	1.000	0.032
E	0.226	0.029	-0.096	0.032	1.000

In MPT, better portfolios are those which combine assets negatively correlated. Some of the thirty-two portfolios (2n) that can be formed from the projects are represented on Table IV.

TABLE IV  
POSSIBLE PORTFOLIOS

Portfolio Number	Projects				
	A	B	C	D	E
#1	0	0	0	0	0
#2	1	0	0	0	0
#3	1	1	0	0	0
#4	1	1	1	0	0
#5	1	1	1	1	0
.	.	.	.	.	.
#32	1	1	1	1	1

Table V shows five candidate projects and their associated variables ( $ER_p$ ,  $\sigma_p^2$ ,  $C_p$ , and  $PEI_p$ ). It can be noted that the portfolio with the best PEI is the #32, but the amount of money required to implement it is greater than the amount of resources available (\$196,000 > \$160,000). Since portfolio #32 is unfeasible, the next best option is portfolio #5, which has the best PEI under the available resources to be invested.

TABLE V  
PORTFOLIOS' VARIABLES

Portfolio	$ER_p$	$\sigma_p^2$	$C_p$	$PEI_p$
#1	0	0	0	0
#2	9,566	2,346.45	26,000	4.076
#3	29,982	4,345.87	56,000	6.898
#4	58,193	4,922.39	106,000	11.822
#5	67,536	5,533.92	156,000	12.204
.	.	.	.	.
#32	91,702	7,374.23	196,000	12.435

Fig. 2 shows the Efficient Frontier plotting the 32 portfolios. Each portfolio is presented as a point. Note the line formed by the uppermost part of the "cloud" of points. Every portfolio placed on it is an optimum portfolio. Selections under the frontier would lead to unnecessary risk, given a certain return, or low return, given a certain risk.

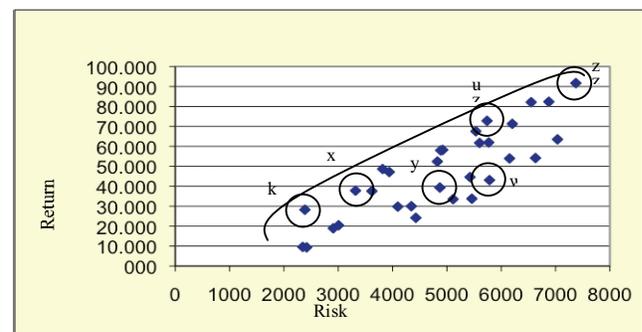


Fig. 2. Efficient Frontier

For instance, portfolios k, x, u, and z are optimum. Portfolios x and y have about the same expected return, but portfolio y has higher risk. Portfolios u and w have almost the same risk, but w has a lower expected return. Therefore, choosing portfolios y and w is not a rational decision. Portfolio z is #32 on Table V (Expected Return: 91.702; Risk: 7.374) - the one with the best PEI, but also requiring more investment.

Thus, the decision, at this point, is to find out how much money is available to invest on the optimums portfolios, or how much risk someone is willing to take to get its correspondent return.

Let us suppose, now, a situation where the portfolio #3 on Table IV (composed by projects A and B) is already being developed and a decision maker wants to decide about the possibility to introduce Project C on the portfolio (leading to portfolio #4). We can observe on Table V that investing \$50,000 (cost of Project C), the return will double, the PEI will almost double, and the risk will be increased only by 15%. Thus, this seems to be a good decision. On the other hand, if we analyze the introduction of projects C and D (Portfolio #5), the cost will be three times higher, the return will not be tripled, the PEI will only double, and the risk will be increased by 25%. Therefore, this is not a rational decision.

This analysis can be corroborated by the study of the correlations among the projects (Table 2). Based on Portfolio #3 (Projects A and B), the addition of project C would be beneficial, because projects A and C are negatively correlated, as well as B and C. However, if we try to introduce project D to the portfolio would be detrimental, because project B and D are positively correlated, as well as C and D. This is a proof of how diversification works on MPT and the adherence of our approach to the foundations of this theory.

## VI. CONCLUSION

In this paper, we presented a technique to select software project portfolios based on the aggregation of many sound theories and techniques. Besides, we have shown that is possible to calculate, and study the interdependency of a group of projects in a formal manner, which was a challenge to be solved. Besides, we highlight some other advantages of our approach such as including risk response plan before the project selection, the possibility to have an overall view of the portfolios, the creation of a single parameter (PEI) on which the possible portfolios can be compared and the Efficient Frontier. We believe the approach is a powerful instrument to analyze portfolios and help decision makers in their job of creating and increasing value to their business, choosing the right projects to compose their portfolio.

## ACKNOWLEDGMENTS

The authors would like to thank and acknowledge the support from the Brazilian Research Council (CNPq) and the Foundation for Scientific Development of Rio de Janeiro State (FAPERJ).

## REFERENCES

- [1] Ball, B. C. and Savage, S. L. 1999. *A New Era in Petroleum Exploration and Production Management*. Notes on Exploration and Production Portfolio Optimization.
- [2] Blau, G. E., Penky, J. F., Varma, V. A., and Bunch, P. R. 2004. "Managing a Portfolio of Independent New Product Candidates in the Pharmaceutical Industry". In *Journal of Product Innovation Management*, No. 21, 227-245.
- [3] Boehm, B.W., Sullivan, K., 2000. "Software Economics: A Roadmap, in The Future of Software Engineering". In 22nd *International Conference on Software Engineering*.
- [4] Boehm, B., Biffl, S., Aurum, A., Grömbacher, P., 2006. *Value-Based Software Engineering*. Springer-Verlag, Germany.
- [5] Cooper, R.G., Edgett, S.J and. Kleinschmidt, E.J. 2001. *Portfolio Management for New Products*. 2nd edition, Cambridge, MS, USA.
- [6] Costa, H.R., Barros, M.O, Travassos, G.H. 2007. "Assessing Software Projects Risks". In *Journal of Systems and Software*, January.
- [7] Damodaran, A. 2002. *Investment Valuation: Tools and Techniques for Determining the Value of Any Asset*. 2nd edition, John Wiley & Sons. NJ, USA.
- [8] Dye, L. D., Pennypacker, J. S., 2003. *Project Portfolio Management: Selecting and Prioritizing Projects for Competitive Advantage*. Glen Mills, PA. Center for Business Practices.
- [9] GAO - General Accounting Office. 1994. "Improving Mission Performance Through Strategic Information Management: Learning from Leading Organizations", GAO/AIMD-94-115, Washington DC.
- [10] Hubbard, D. 207. *How to Measure Anything. Finding the Value of Intangibles in Business*. John Wiley & Sons. Hoboken, NJ, USA.
- [11] Killen, C.P., Hunt, R.A., Kleinschmidt, E.J. 2007. "Managing the New Product Development Project Portfolio: A Review of the Literature and Empirical Evidence". In *PICMET Proceedings*, Portland, Oregon, USA, August.
- [12] Levine, H.A., 2005. *Project portfolio management: A practical guide to selecting projects, managing portfolios, and maximizing benefits*. John Wiley, San Francisco, CA, USA.
- [13] McFarlan, F. W. 1981. *Portfolio Approach to Information Systems*. IEEE Press, Piscataway, NJ, USA.
- [14] Markowitz, H.M. 1952. "Portfolio Selection". In *Journal of Finance*, Vol. 7, No 1, pp. 77-91.
- [15] Ringuest, L., Graves, S.B., Case, R. "Formulating R&D Portfolios that Account for Risk". *Research-Technology Management, Research Institute Inc*. 1999.
- [16] Ross, S.A, Jordan, B.D, Westerfield, R.W. 2008. *Essentials of Corporate Finance*. 6th Ed. , McGraw- Hill
- [17] Sullivan, K., Chalasani, P., Jha, S., Sazawal, V., 1999. "Software Design as an Investment Activity: A Real Options Perspective". In *Real Options and Business Strategy: Application to Decision Making*, L. Trigeorgis, Consulting Editor, Risk Books.

# Runtime Constraint Checking Approaches for OCL, A Critical Comparison

Carmen Avila  
Dept. of Computer Science  
University of Texas at El Paso  
El Paso, TX 79968  
ceavila3@miners.utep.edu

Amritam Sarcar  
Microsoft Corporation  
One Microsoft Way  
Redmond, WA 98052  
amritams@microsoft.com

Yoonsik Cheon and Cesar Yeep  
Dept. of Computer Science  
University of Texas at El Paso  
El Paso, TX 79968  
{ycheon@, ceyeep@miners}.utep.edu

**Abstract**—There are many benefits of checking design constraints at runtime—for example, automatic detection of design drift or corrosion. However, there is no comparative analysis of different approaches although such an analysis could provide a sound basis for determining the appropriateness of one approach over the others. In this paper we conduct a comparative analysis and evaluation of different constraint checking approaches possible for the Object Constraint Language (OCL). We compare several approaches including (1) direct translation to implementation languages, (2) use of executable assertion languages, and (3) use of aspect-oriented programming languages. Our comparison includes both quantitative metrics such as runtime performance and qualitative metrics such as maintainability of constraint checking code. We found that the implementation language-based approaches perform better in terms of memory footprint and runtime overheads but the other approaches are more appealing in terms of maintainability.

**Keywords**—pre and postconditions; runtime constraint checking; AspectJ; JML; OCL

## I. INTRODUCTION

A recent trend in software development is a shift of focus from writing code to building models [1]. The idea is to systematically generate an implementation from a model through a series of transformations. A key requirement of this model-driven development is the availability of a precise model to generate working code from it. A formal notation such as the Object Constraint Language (OCL) [2] can play an important role to build such a precise model because the most popular modeling language, UML [3], lacks sufficient precision to enable complete code generation; OCL is a textual, declarative notation to specify constraints or rules that apply to UML models. Modeling and specifying design constraints explicitly is also said to improve reasoning of software architectures and thus their qualities.

Besides static reasoning, formally specified design constraints such as OCL constraints can be checked at runtime, and there are many benefits of checking design constraints at runtime. For example, it can detect when an implementation deviates from its design, often called *design corrosion* or *drift* [4] [5]. Design corrosion is said to be proportional to the development and maintenance time and occurs when the initial design of software gets modified to accommodate new or changed requirements or to correct defects. It also occurs as

the result of code hacks and workarounds, a common practice of software maintenance. Runtime constraint checking can also facilitate automating program or conformance testing by allowing constraints to be used as test oracles [6].

Several different approaches are possible for checking design constraints such as OCL constraints against implementations at runtime. The most common approach is to translate constraints directly to an implementation language by coding a constraint checker in that language and making it part of the implementation [7]. Constraints can also be translated to executable assertions if the implementation language provides an assertion facility such as an `assert` statement or if it has a separate assertion languages [8] [9]. Yet another possibility is to apply aspect-oriented programming to modularize constraint checking code by implementing constraint checking as a crosscutting concern (see Section III-C for details) [10] [11].

We expect that each of the aforementioned approaches have its strengths and weaknesses. In this paper we conduct a comparative analysis of these approaches in order to determine appropriateness of one approach over the others. In our study we use OCL as the constraint specification language and Java as the implementation language. Our analysis will involve applying different approaches to a common set of OCL constraints and recording a set of metrics from each application. For the comparison we will use both quantitative metrics such as runtime speed and memory usage and qualitative metrics such as maintainability of constraint checking code. We will consider various types of OCL constraints such as class invariants, operation pre and postconditions.

The remainder of this paper is structured as follows. In Section II we briefly explain OCL by introducing example constraints to be used throughout this paper. In Section III we describe in detail four different constraint checking approaches by applying them to the example constraints. In Section IV we first describe the case study that we performed to compare the approaches and then analyze the results from the case study. In Section V we conclude this paper with a summary of our findings.

## II. BACKGROUND ON OCL

The Object Constraint Language (OCL) [2] is a textual, declarative notation to specify constraints or rules that apply to

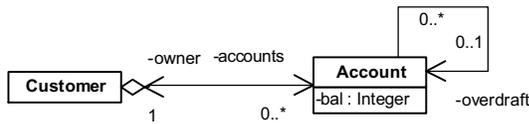


Fig. 1. UML class diagram

UML models. OCL can play an important role in model-driven software development because UML lacks sufficient precision to enable the transformation of a UML model to complete code. In fact, it is a key component of OMG’s standard for model transformation for the model-driven architecture [12].

A UML diagram alone cannot express a rich semantics of and all relevant information about an application. The diagram shown in Figure 1, for example, is a UML class diagram for bank accounts. A customer can own several accounts, and an account can be linked to another account for overdraft protection. However, the class diagram doesn’t express the fact that an account cannot be linked to itself for overdraft protection. It is very likely that a system built based only on diagrams alone will be incorrect. OCL allows to precisely describe this kind of additional constraints on the objects and entities present in a UML model. It is based on mathematical set theory and predicate logic and supplements UML by providing expressions that have neither the ambiguities of natural language nor the inherent difficulty of using complex mathematics. The above-mentioned fact, for example, can be expressed in OCL as follows.

```

context Account
  inv: self <> overdraft
  
```

This constraint, called an *invariant*, states a fact that should be always true in the model. The keyword `self` denotes the object being constrained by an OCL expression, called a *contextual instance*; in this case it is an instance of the `Account` class. The invariant says that an account cannot be equal to its overdraft protection account. It is also possible to specify the behavior of an operation in OCL. For example, the following OCL constraints specifies the behavior of an operation `Customer::addAccount` by writing a pair of predicates called *pre* and *postconditions*.

```

context Customer::addAccount(acc: Account): void
  pre: not accounts->includes(acc)
  post: accounts = accounts@pre->including(acc)
  
```

The pre and postconditions states that, given an account not already owned by a customer, the operation should insert the account to the set of accounts owned by the customer. The postfix operator `@pre` denotes the value of a property (`accounts`) in the pre-state, i.e., just before an operation invocation. The constraints are written using OCL collection operations such as `includes` and `including`; the `includes` operation tests whether an object is contained in a collection, and the `including` operation adds an element to a collection.

### III. RUNTIME CONSTRAINT CHECKING

Several different approaches are possible for checking design constraints against implementations at runtime. For example, Froihofer et al. reviewed and evaluated different constraint

validation approaches for Java [13]. They discussed hand-crafted approaches, code instrumentation using OCL and JML [14], aspect-oriented programming, proxy implementations, CORBA, and EJBs. Each approach has its own advantages and disadvantages such as runtime overhead that ranges from a factor of two to more than one hundred. In this paper we focus on approaches available for OCL by considering OCL-specific features and consider only those approaches that do not require external or separate constraint checking monitors. We study the following three approaches classified by the target language to which OCL constraints are translated or in which the checking code is written.

- *Implementation languages.* This is the most widely-used approach and maps OCL constraints to an implementation language in that a constraint checker is written in that language and becomes part of the implementation (see for example [7]). If the implementation language supports an assertion facility such as the `assert` macro or statement, constraints can also be translated to executable assertions.
- *Assertion languages.* Some programming languages such as Eiffel support class invariants and operation pre and postconditions as built-in language constructs called *design-by-contract* [15]. Design-by-contract is not a formal part of Java, but there are several extensions or tools to support it for Java [14] [16]. OCL constraints can be translated to design-by-contract assertions [8] [9].
- *Aspect-oriented languages.* If an implementation language has an aspect-oriented extension, e.g. AspectJ for Java, it can be used to implement constraint checking code [10] [11] [17]. Constraint checking is viewed as a crosscutting concern and implemented as a so-called *aspect* that resides in a separate module and *advises* the implementation code (see Section III-C below).

In the following subsections we explain each of the aforementioned approaches in detail using the OCL examples introduced in Section II.

#### A. Translating to Implementation Languages

1) *Using Programming Language Statements:* In this approach one injects hand-crafted checking code to an implementation. For each OCL constraint, one has to decide appropriate checking points and then translate the constraint to programming language statements. For example, a class invariant should be checked at the end of a constructor execution and before and after the execution of every method because a constructor has to establish the class invariant and every method has to preserve it. As an example, let us consider the `addAccount` operation of the `Customer` class introduced in Section II. Here is a possible implementation of the operation in Java.

```

public void addAccount(Account acc) {
  // check invariant at pre-state if any
  // check precondition
  if (accounts.contains(acc))
    throw new OclError("Precondition_violation");
  // calculate accounts@pre
  Set<Account> accsPre = new HashSet<Account>(accounts);
  
```

```

accounts.add(acc);
acc.setOwner(this);

// check postcondition
accsPre.add(acc);
if (!accounts.equals(accsPre))
    throw new OclError("Postcondition_violation");
// check invariant at post-state if any
}

```

As shown, the method body is wrapped with constraint checking code that checks the pre and postconditions and the class invariant as well in the pre and post-states.

The main shortcoming of this approach is that there are a lot of manual work involved, such as translating OCL constraints to programming language statements and implementing the supporting infrastructure (e.g., one for constraint inheritance). Additionally, the resulting code may not be maintainable (refer to Section IV for an analysis).

2) *Using Assertion Facilities:* This approach is similar to the previous one except that OCL constraints are now translated to executable assertions of the implementation language. For example, the following is the `addAccount` method with OCL constraints translated to Java `assert` statements.

```

public void addAccount(Account acc) {
    assert !accounts.contains(acc) : "Precondition";
    Set<Account> accsPre = new HashSet<Account>(accounts);

    accounts.add(acc);
    acc.setOwner(this);

    accsPre.add(acc);
    assert accounts.equals(accsPre) : "Postcondition";
}

```

As in the previous approach, one has to determine appropriate constraint checking points and manually translate the constraints. However, one advantage of this approach is its ability to selectively enable or disable assertions; in Java, for example, one can control assertions at various granularities by using command-line switches.

## B. Using Assertion Languages

In this approach, OCL constraints are translated to executable assertions such as design-by-contract. There are several extensions to Java that support design-by-contract [13]. For example, the Java Modeling Language (JML) [14] [16] is a formal interface specification language for Java to document the behavior of Java classes and interfaces, and a significant subset of JML is executable. The following code shows the `addAccount` method annotated with JML specifications.

```

/*@ public model JMLObjectSet accSet;
   @ private represents accSet
   @ = JMLObjectSet.convertFrom(accounts);
   @*/

/*@ requires !accSet.has(acc);
   @ ensures accSet.equals(\old(accSet.insert(acc)));
   @*/
public void addAccount(Account acc) { /* ... */ }

```

As shown, JML annotations are enclosed in special comments such as `/*@ ... @*/` and precede the Java declarations such as method declarations that are being annotated. Method pre and postconditions follow the keywords `requires` and `ensures`, respectively. The JML-specific `\old` expression

denotes the pre-state value of its argument. An interesting feature of JML is that it provides a built-in support for writing abstract specifications [18]. For example, the above pre and postconditions are written in terms of a specification-only variable `accSet` of which value is given as a mapping from a program variable `accounts`. This way of writing assertions have several advantages; for example, such assertions are less affected by implementation changes and do not expose implementation details such as a private field `accounts`. Another strength of using assertion languages is that OCL constraints are often directly mapped to assertions. This is particularly noticeable when translating OCL constraints consisting of iterator operations such as `forall` and `exists` because similar sorts of quantifiers are supported in JML.

## C. Using Aspect-Oriented Programming Languages

Aspect-oriented programming is a new programming paradigm to address in a modular way so-called *crosscutting concerns* such as logging that have to be implemented in multiple program modules. The key idea is to denote a set of execution points, called *join points*, and introduce additional behavior, called an *advice*, at the join points. AspectJ [19] is an aspect-oriented extension for Java and provides built-in language constructs for join points and advices. OCL constraints can be systematically translated to AspectJ code to check at runtime the conformance of a Java implementation [10] [11] [17]. For example, the following AspectJ code checks the pre and postconditions of the `addAccount` operation.

```

public privileged aspect CustomerChecker {
    pointcut addAccountExe(Customer c, Account a):
        execution(void Customer.addAccount(Account))
        && this(c) && args(a);

    void around(Customer c, Account a): addAccountExe(c, a) {
        assert !c.accounts.contains(a) : "Precondition";
        Set<Account> accsPre = new HashSet<Account>(c.accounts);
        proceed(c, a);
        accsPre.add(a);
        assert c.accounts.equals(accsPre) : "Postcondition";
    }
}

```

The `pointcut` declaration designates a set of execution points and optionally exposes certain values at these execution points. For example, the `pointcut addAccountExe` denotes execution of the `addAccount` method and exposes the receiver (`c`) and the argument (`a`). The `around` keyword introduces an advice that wraps around a join point and can potentially replace it; there are also `before` and `after` advices. The above advice first checks the precondition by referring to the values exposed by the `pointcut`, proceeds to continue with the normal flow of execution at the join point (as indicated by the `proceed` keyword), and finally checks the postcondition. If the `Customer` class is compiled with the above aspect, every execution of the `addAccount` method will be checked against the pre and postconditions. The aspect-oriented approach has several advantages over the previous approaches. For example, the constraint checking logic is completely separated from the implementation, and the implementation modules are oblivious of the constraint checking code, even its existence. Thus,

constraint checking code can be easily added or removed from the implementation. It will also enable runtime checks to be applied to different implementations of the same design and be selectively enabled or disabled, for example, for production code.

#### IV. COMPARISON

To find out the strengths and weaknesses of the approaches explained in the previous section, we compare them both quantitatively and qualitatively. For the comparison we use the following quantitative metrics.

- Source code size. We measure and compare this because it indicates the amount of work needed to implement constraint checking code. We also measure the number of source code lines needed to translate one line of a constraint.
- Bytecode size. This is one factor that determines the memory footprint of a program and thus may be important for certain systems like consumer electronics and embedded systems where low-memory-footprint programs are required.
- Dynamic memory usage. This is another factor that determines the memory footprint of a program.
- Execution time. This may be one of the most important criteria for selecting a constraint checking approach, especially for use in production code.

We also compare the approaches qualitatively using such criteria as easiness of translation, support for automation, and maintainability of checking code. In the following subsections we first describe the case study that we performed for the comparison and then analyze the comparison results.

##### A. Case Study

We performed a case study by using an open-source Java program that has a formal UML model including OCL constraints. The use of an open-source program eliminates subjectiveness during the experiment and makes the case study more realistic. The OCL standard specification defines several collection types such as Collection, Set, OrderedSet, Bag, and Sequence, and the behavior of each type is formally specified in OCL [20]. There are Java implementations of the OCL collection types [8] [21], and for our case study we used the one included in the Dresden OCL Toolkit [21]. The standard specifies 336 lines of OCL constraints, most of which are postconditions, and the implementation has 87 methods and 1781 lines of source code including comments.

We manually translated OCL constraints to runtime checks using each of the approaches. For the two Java-based approaches we directly modified the source code to insert assertion checking code. For the JML-based approach we also changed the source code to add JML annotations. For the AOP-based approach, however, instead of modifying the Java source code we introduced one AspectJ aspect for each Java class, responsible for checking all the constraints specified for that class. We next devised a suite of test data for each collection type and measured the runtime performance of each

TABLE I  
SOURCE CODE SIZE

	OCL	Src*	CC	CC/Src	CC/OCL
Stmt	336	1781	1025	0.58	3.05
Asrt	336	1781	401	0.22	1.19
JML	336	1781	330	0.19	0.98
AOP	336	1781	1257	0.71	3.74

\*Src: Java code; CC: constraint checking code; size in LOC

TABLE II  
BYTECODE SIZE

	All (kb)*	CC (kb)	CC/Src	CC/OCL
Stmt	42	20	0.91	0.06
Asrt	34	12	0.55	0.04
JML	307	285	12.96	0.85
AOP	71	49	2.22	0.15

\*The columns show the size of base code plus checking code, the size of checking code, the ratio of checking code over base code (22kb), and the ratio of checking code over OCL lines (336 lines), respectively.

approach. The test suite also showed that all approaches are equally effective in detecting constraint violations at runtime; it revealed several errors both in the implementation and in the constraints themselves [11].

##### B. Results

1) *Source Code Size*: The source code size is an important metric because it indicates the amount of work needed. Table I shows the measurement of source code size for each approach, given in the number of source code lines. It also shows the average number of source code lines needed to translate one line of OCL constraints (see the CC/OCL column). As expected, JML is superior in this metric because it supports similar language constructs as OCL including invariants, pre and postconditions, and quantifiers, and thus most OCL constraints can be directly translated to JML specifications. The AOP approach requires the most work because one has to not only translate OCL constraints to Java statements but also introduce AOP-specific declarations such as pointcuts, advices, and aspects.

2) *Bytecode Size*: Table II shows the bytecode size of constraint checking code. The Java `assert` statement-based approach produces the most compact bytecode. It produces about 24 times more compact bytecode than the JML approach and requires on average about 21 times less bytecode per line of OCL constraints. It is interesting to learn that the bytecode size is not always proportional to the source code size. This is perhaps because both AspectJ and JML compilers have to produce code for a runtime support framework—for example, for dynamic pointcut resolution and for specification inheritance. Furthermore, the particular JML compiler (`jml4c`) that we used for this case translates quantifiers to inner classes, which brings an additional overhead on bytecode size [22]; more than half of the translated JML assertions were quantified expressions.

3) *Dynamic Memory Usage*: Table III shows the dynamic memory requirement for each approach, obtained using the Eclipse profiling tools. It shows the number of live instances,

TABLE IV  
EXECUTION TIME

	No. of calls		CPU time	
	Number	Overhead	Sec	Overhead
Base	2101	0	0.04	0
Stmnt	5842	1.78	0.11	1.98
Asrt	3124	0.49	0.09	1.09
JML	45646	20.73	2.99	79.13
AOP	14869	6.08	0.70	17.75

TABLE V  
SUMMARY OF CONSTRAINT CHECKING OVERHEADS

	Source	Bytecode	Memory	CPU Time
Stmnt	0.58	0.91	0.01	1.98
Asrt	0.22	0.55	0.01	1.09
JML	0.19	12.96	0.84	79.13
AOP	0.71	2.22	0.02	17.75

active size in bytes, the total number of instances, and the total size in bytes; a live instance is an instance that is alive, i.e., not garbage collected. The table also shows the memory overhead for each approach. The JML-based approach requires eight times more heap storages than the base code, and for other approaches the memory overhead is negligible. We suspect that this is because the JML compiler translates quantified assertions to inner classes.

4) *Execution Time*: Table IV shows the number of method calls and the CPU time required to run the test suite by each approach, along with the overhead due to constraint checking. As shown, the Java-based approaches outperform both the JML and the AOP-based approaches; for example, the JML-based approach is about 27 to 38 times slower than the Java-based approaches and requires 80 times more CPU time than the base code. This may be explained in part by the huge number of additional method calls introduced by the constraint checking code. We also learned that application characteristics influence the execution times differently for different approaches; for example, both the AOP and the Java statement-based approaches require the longest execution time for the Collection class, the Java assert-based approach for the Set class, and the JML approach for the Sequence class.

5) *Summary of Overheads*: Table V and Figure 2 summarize the overheads of runtime constraint checking. The use of languages such as JML and AspectJ introduces significant runtime overheads on CPU time and dynamic memory storage. For example, programs with JML annotations require 84 times more CPU time and 13 times more heap storage than those without JML annotations. However, the increases of source and bytecode sizes are relatively negligible compared to runtime overheads. In summary, the Java-based approaches outperform the other approaches when considering only runtime overheads.

6) *Qualitative Comparison*: The case study also allowed us to compare the four approaches qualitatively using such criteria as translation easiness and maintainability of checking code, and Table VI summarizes the comparison result. Most OCL constructs and expressions were directly mapped and

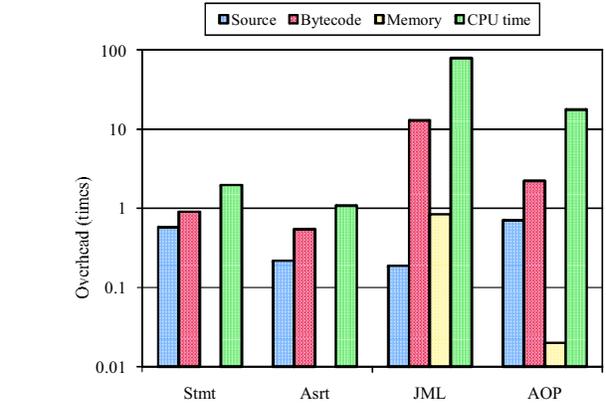


Fig. 2. Summary of overheads

translated to JML specifications. This was particularly noticeable for OCL expressions containing iterator operations such as `forall` and `exists`, as most OCL iterator operations can be mapped to JML quantifiers. In other approaches, such expressions were translated to sequences of Java statements composed of loop statements. Regarding support for automation, there are OCL translation rules defined for JML and AspectJ along with a support tool for the AspectJ translation [8] [9] [21]. We found that it is a lot easier to read and understand constraint checking code when constraints are expressed in assertions such as Java `assert` statements and JML annotations rather than Java code statements. JML specifications and AspectJ source code are modular in that they can reside in separate specification or source code files and the base code—the module being annotated in JML or advised in AspectJ—doesn't depend on them. This has several benefits including reusability, controllability, and maintainability. The same JML specification or AspectJ code can be used for different implementations or versions of the same design; it is reusable and plug-and-playable. It is also easy to selectively turn on and off constraint checking by simply recompiling program modules; in JML, it is also possible to enable assertions based on their kinds such as invariants and method pre and postconditions. JML specifications and AspectJ code are more maintainable because they can better accommodate changes both in OCL constraints and checking code itself; they are not tangled with nor scattered over the base code. Regarding maturity of technology, JML is a research language still being developed and lacks stable support tools, and as a relatively new technology AOP is not widely accepted or used yet.

## V. CONCLUSION

We explained four different approaches for translating OCL constraints to runtime checks: (1) using implementation languages such as Java, (2) using built-in assertion facilities such as the `assert` statement, (3) using assertion or design-by-contract languages such as JML, (4) using aspect-oriented programming language such as AspectJ. We then compared

TABLE III  
DYNAMIC MEMORY USAGE

	Live instances		Active size		Total instance		Total size	
	Number	Overhead	Byte	Overhead	Number	Overhead	Byte	Overhead
Base	248	0	5552	0	581	0	10880	0
Stmt	337	0.26	6976	0.20	589	0.01	11040	0.01
Asrt	316	0.22	6640	0.16	590	0.02	11024	0.01
JML	447	0.45	47256	0.88	705	0.18	67576	0.84
AOP	461	0.46	8936	0.38	597	0.03	11144	0.02

TABLE VI  
QUALITATIVE COMPARISON

	Stmt	Asrt	JML	AOP
Translation	×	△	○	×
Automation	×	×	○	○
Readability	×	△	○	×
Reusability	×	×	○	○
Controllability	×	△	○	○
Maintainability	×	×	○	△
Maturity	○	○	×	△

○: good; △: fair; ×: bad

these approaches critically through a case study. We learned that the first two approaches based on implementation languages are most efficient in terms of runtime performance such as CPU time and heap storage. However, our qualitative comparison favored the other two approaches. For example, OCL constraints, in most cases, can be directly translated to JML annotations, and the resulting JML specifications are easy to read and understand. There are translation rules from OCL to JML and AspectJ. JML specifications and AspectJ code are better modularized and thus reusable, plug-and-playable, controllable, and maintainable. In summary, the first two approaches may be a better choice for the use of constraint checking in production code if memory footprint or runtime speed is an important concern. On the other hand, the other two approaches may be more appealing for the development use (e.g., testing and debugging) of constraint checking where concerns such as accommodation for changes are more important.

#### ACKNOWLEDGMENT

The work of the authors was supported in part by NSF grants CNS-0707874 and DUE-0837567.

#### REFERENCES

- [1] A. W. Brown, "Model driven architecture: Principles and practice," *Software and System Modeling*, vol. 3, no. 4, pp. 314–327, Dec. 2004.
- [2] J. Warmer and A. Kleppe, *The Object Constraint Language: Getting Your Models Ready for MDA*, 2nd ed. Addison-Wesley, 2003.
- [3] J. Rumbaugh, I. Jacobson, and G. Booch, *The Unified Modeling Language Reference Manual*, 2nd ed. Addison-Wesley, 2004.
- [4] D. E. Perry and A. L. Wolf, "Foundations for the study of software architecture," *ACM SIGSOFT Software Engineering Notes*, vol. 17, no. 4, pp. 40–52, 1992.
- [5] M. Fowler, *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, Aug. 1999.
- [6] Y. Cheon and C. Avila, "Automating Java program testing using OCL and AspectJ," in *ITNG 2010: 7th International Conference on Information Technology: New Generations*, April 12–14, 2010, Las Vegas, NV. IEEE Computer Society, 2010, pp. 1020–1025.
- [7] H. Hussmann, B. Demuth, and F. Finger, "Modulr architecture for a toolset supporting OCL," in *UML 2000 — The Unified Modeling Language, Advancing the Standard*, York, UK, October 2000, ser. LNCS, A. Evans, S. Kent, and B. Selic, Eds. Springer-Verlag, 2000, vol. 1939, pp. 278–293.
- [8] C. Avila, G. Flores, and Y. Cheon, "A library-based approach to translating OCL constraints to JML assertions for runtime checking," in *International Conference on Software Engineering Research and Practice*, July 14–17, 2008, Las Vegas, Nevada, 2008, pp. 403–408.
- [9] A. Hamie, "Translating the Object Constraint Language into the Java Modeling Language," in *Proceedings of the ACM Symposium on Applied Computing*, Nicosia, Cyprus, March 14–17, 2004, 2004, pp. 1531–1535.
- [10] L. C. Briand, W. J. Dzidek, and Y. Labiche, "Instrumenting contracts with aspect-oriented programming to increase observability and support debugging," in *Proceedings of the 21st IEEE International Conference on Software Maintenance*, Budapest, Hungary, September 25–30, 2005, Sep. 2005, pp. 687–690.
- [11] Y. Cheon, C. Avila, S. Roach, and C. Munoz, "Checking design constraints at run-time using OCL and AspectJ," *International Journal of Software Engineering*, vol. 2, no. 3, pp. 5–28, Dec. 2009.
- [12] D. S. Frankel, *Model Driven Architecture: Applying MDA to Enterprise Computing*. Wiley, Jan. 2003.
- [13] L. Frohofer, G. Glos, J. Osrael, and K. M. Goeschka, "Overview and evaluation of constraint validation approaches in Java," in *ICSE '07: Proceedings of the 29th International Conference on Software Engineering*. IEEE Computer Society, 2007, pp. 313–322.
- [14] G. T. Leavens, A. L. Baker, and C. Ruby, "Preliminary design of JML: A behavioral interface specification language for Java," *ACM SIGSOFT Soft. Eng. Notes*, vol. 31, no. 3, pp. 1–38, Mar. 2006.
- [15] B. Meyer, "Applying "design by contract"," *Computer*, vol. 25, no. 10, pp. 40–51, Oct. 1992.
- [16] L. Burdy, Y. Cheon, D. Cok, M. Ernst, J. Kinary, G. T. Leavens, K. R. M. Leino, and E. Poll, "An overview of JML tools and applications," *International Journal on Software Tools for Technology Transfer*, vol. 7, no. 3, pp. 212–232, Jun. 2005.
- [17] W. J. Dzidek, L. C. Briand, and Y. Labiche, "Lessons learned from developing a dynamic OCL constraint enforcement tool for Java," in *ACM/IEEE 8th International Conference on Model Driven Engineering Languages and Systems*, Montego Bay, Jamaica, October 2–7, 2005, ser. LNCS. Springer-Verlag, 2006, vol. 3844, pp. 10–19.
- [18] Y. Cheon, G. T. Leavens, M. Sitaraman, and S. Edwards, "Model variables: Cleanly supporting abstraction in design by contract," *Software—Practice & Experience*, vol. 35, no. 6, pp. 583–599, May 2005.
- [19] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W. G. Griswold, "An overview of AspectJ," in *ECOOP 2001 — Object-Oriented Programming 15th European Conference*, Budapest Hungary, ser. LNCS, J. L. Knudsen, Ed. Berlin: Springer-Verlag, Jun. 2001, vol. 2072, pp. 327–353.
- [20] OMG, *UML 2.0 OCL Specification*. Object Management Group, Oct. 2003, available from <http://www.omg.org/docs/ptc/03-10-14.pdf> (retrieved on Feb. 23, 2010).
- [21] B. Demuth and C. Wilke, "Model and object verification by using Dresden OCL," in *Proceedings of the Russian-German Workshop Innovation Information Technologies: Theory and Practice*, July 25–31, Ufa, Russia, 2009, 2009, pp. 687–690.
- [22] A. Sarcar and Y. Cheon, "A new Eclipse-based JML compiler built using AST merging," Department of Computer Science, The University of Texas at El Paso, Tech. Rep. 10-08, Mar. 2010.

# Refinement Checking for Interface Automata with Z Notation

Zining Cao<sup>1,2,3</sup>

<sup>1</sup>National Key Laboratory of Science and Technology on Avionics System Integration  
Shanghai 200233, P. R. China

<sup>2</sup>Department of Computer Science and Technology  
Nanjing University of Aero. & Astro.  
Nanjing 210016, P. R. China

<sup>3</sup>Provincial Key Laboratory for Computer Information Processing Technology, Soochow University  
Suzhou 215006, P. R. China  
Email: caozn@nuaa.edu.cn

**Abstract**—In this paper, we first introduce a specification approach combining interface automata and Z language, which is named ZIA. A refinement relation on ZIAs is proposed. We then give an algorithm for checking refinement relation between ZIAs with finite domain.

**Index Terms**—interface automata; Z notation; refinement checking

## I. INTRODUCTION

Modern software systems are comprised of numerous components, and are made larger through the use of software frameworks. Such software systems exhibit various behavioural aspects such as communication between components, and state transformation inside components. Formal specification and verification techniques for such systems have to be able to describe and verify all these aspects. But there is seldom works about a single specification technique that is well suited for all these aspects. In this paper we introduce a specification model to describe both data and behavioural aspects of software systems. This model, named ZIA, is a combination of interface automata and Z. Furthermore, we propose a refinement relation on ZIAs. At last, to verify properties of ZIAs, a refinement checking algorithm is presented.

Interface automata is a light-weight automata-based languages for component specification, which was proposed in [1]. An interface automaton (IA), introduced by de Alfaro and Henzinger, is an automata-based model suitable for specifying component-based systems. IA is part of a class of models called interface models, which are intended to specify concisely how systems can be used and to adhere to certain well-formedness criteria that make them appropriate for modelling component-based systems.

Z [3], [13], [14] is a typed formal specification notation based on first order predicate logic and set theory. The formal basis for Z is first order predicate logic extended with type set theory. Using mathematics for specification is all very well for small examples, but for more realistically sized problems, things start to get out of hand. To deal with this, Z includes the schema notation to aid the structuring and modularization

of specifications. A boxed notation called schemas is used for structuring Z specifications. This has been found to be necessary to handle the information in a specification of any size. In particular, Z schemas and the schema calculus enable a structured way of presenting large state spaces and their transformation.

In this paper, we introduce a specification language which combines Z language and interface automata, named ZIA. Roughly speaking, a ZIA is in a style of interface automata but its states and operations are described by Z language. Then the refinement relation between ZIAs is defined. At last, we provide an algorithm for checking refinement relation on ZIAs with finite domain.

This paper is organized as follows: Section 2 gives a brief review of interface automata. Section 3 gives a brief review of Z language. In Section 4, we propose a specification language-ZIA. Furthermore, the refinement relation for ZIA is presented. In Section 5, we give a refinement checking algorithm for ZIAs with finite domain. The paper is concluded in Section 6.

## II. INTERFACE AUTOMATA

An interface automaton (IA) [1], introduced by de Alfaro and Henzinger, is an automata-based model suitable for specifying component-based systems. IA is part of a class of models called interface models, which are intended to specify concisely how systems can be used and to adhere to certain well-formedness criteria that make them appropriate for modelling component-based systems. The two main characteristics of interface models are that they assume a helpful environment and support top-down design.

**Definition 1** An interface automaton (IA)  $P = \langle V_P, V_P^i, A_P^l, A_P^o, A_P^h, T_P \rangle$  consists of the following elements:

- (1)  $V_P$  is a set of states,
- (2)  $V_P^i \subseteq V_P$  is a set of initial states. If  $V_P^i = \emptyset$  then  $P$  is called empty.
- (3)  $A_P^l, A_P^o$  and  $A_P^h$  are disjoint sets of input, output, and internal actions, respectively. We denote by  $A_P = A_P^l \cup A_P^o \cup A_P^h$  the set of all actions.

(4)  $T_P$  is the set of transitions between states such that  $T_P \subseteq V_P \times A_P \times V_P$ .

The interface automaton  $P$  is closed if it has only internal actions, that is,  $A_P^I = A_P^O = \emptyset$ ; otherwise we say that  $P$  is open.

IA  $Q$  refines IA  $P$  if  $Q$  provides the services of  $P$ ; it can have more inputs but no more output actions. As such, a refinement of an IA does not constrain the environment more than the original IA does.

### III. Z LANGUAGE

Z was introduced in the early 80's in Oxford by Abrial as a set-theoretic and predicate language for the specification of data structure, state spaces and state transformations. The first systematic description of Z is [13]. Since then the language has been used in many case studies and industrial projects (e.g. [3], [14]).

Z includes the schema notation to aid the structuring and modularization of specifications. A boxed notation called schemas is used for structuring Z specifications.

For example, the schema below introduces a symbol  $x$ , an element of  $S$ , satisfying the predicate  $p$ .

$$\boxed{\begin{array}{l} A \\ x : S \\ p \end{array}}$$

A generic form of schema may be used to define a family of global constants, parameterised by some set  $X$ .

$$\boxed{\begin{array}{l} A[X] \\ x : X \\ p \end{array}}$$

This definition introduces a generic constant  $x$  of type  $X$ , satisfying predicate  $p$ . The set  $X$  is a formal parameter; it can be regarded as a basic type whose scope is the body of the definition. Any value given to this parameter when the definition is used must be of set type.

Schemas are primarily used to specify state spaces and operations for the mathematical modelling of systems. For example, here is a schema called StateSpace:

$$\boxed{\begin{array}{l} \text{StateSpace} \\ x_1 : S_1; \dots; x_n : S_n \\ \text{Inv}(x_1, \dots, x_n) \end{array}}$$

This schema specifies a state space in which  $x_1, \dots, x_n$  are the state variables and  $S_1, \dots, S_n$  are expressions from which their types may be systematically derived. Z types are sets -  $x_1, \dots, x_n$  should not occur free in  $S_1, \dots, S_n$ , or if they do, they refer instead to other occurrences of these variables already in scope (e.g., globally defined variables).  $\text{Inv}(x_1, \dots, x_n)$  is the state invariant, relating the variables in some way for all possible allowed states of the system during its lifetime.

Z makes use of identifier decorations to encode intended interpretations. A state variable with no decoration represents the current (before) state and a state variable ending with a prime ( $'$ ) represents the next (after) state. A variable ending with a question mark ( $?$ ) represents an input and a variable ending with an exclamation mark ( $!$ ) represents an output.

A typical schema specifying a state change is the following operation schema:

$$\boxed{\begin{array}{l} \text{Operation} \\ x_1 : S_1; \dots; x_n : S_n \\ x'_1 : S_1; \dots; x'_n : S_n \\ i_1? : T_1; \dots; i_m? : T_m \\ o_1! : U_1; \dots; o_p! : U_p \\ \text{Pre}(i_1?, \dots, i_m?, x_1, \dots, x_n) \\ \text{Inv}(x_1, \dots, x_n) \\ \text{Inv}(x'_1, \dots, x'_n) \\ \text{Op}(i_1?, \dots, i_m?, x_1, \dots, x_n, x'_1, \dots, x'_n, o_1!, \dots, o_p!) \end{array}}$$

The inputs are  $i_1?, \dots, i_m?$ ; the outputs are  $o_1!, \dots, o_p!$ ; the precondition is:  $\text{Pre}(i_1?, \dots, i_m?, x_1, \dots, x_n)$ . The state change  $(x_1, \dots, x_n)$  to  $(x'_1, \dots, x'_n)$  is specified by:  $\text{Op}(i_1?, \dots, i_m?, x_1, \dots, x_n, x'_1, \dots, x'_n, o_1!, \dots, o_p!)$ .

The vertical form of schema

$$\boxed{\begin{array}{l} S \\ D_1; \dots; D_m \\ P_1; \dots; P_n \end{array}}$$

is equivalent to the horizontal form of schema

$$S \triangleq [D_1; \dots; D_m \mid P_1; \dots; P_n]$$

In the following, we sometimes use the horizontal form.

In Z [3], [13], [14], there are many schema operators. For example, we write  $S \wedge T$  to denote the conjunction of these two schemas: a new schema formed by merging the declaration parts of  $S$  and  $T$  and conjoining their predicate parts.  $S \Rightarrow T$  ( $S \Leftrightarrow T$ ) is similar to  $S \wedge T$  except connecting their predicate parts by  $\Rightarrow$  ( $\Leftrightarrow$ ). The hiding operation  $S \setminus (x_1, \dots, x_n)$  removes from the schema  $S$  the components  $x_1, \dots, x_n$  explicitly listed, which must exist. Formally,  $S \setminus (x_1, \dots, x_n)$  is equivalent to  $(\exists x_1 : t_1; \dots; x_n : t_n \bullet S)$ , where  $x_1, \dots, x_n$  have types  $t_1, \dots, t_n$  in  $S$ . The notation  $\exists x : a \bullet S$  states that there is some object  $x$  in  $a$  for which  $S$  is true. The notation  $\forall x : a \bullet S$  states that for each object  $x$  in  $a$ ,  $S$  is true.

For the sake of space, more details of Z can be referred to some books on Z [3], [13], [14].

### IV. INTERFACE AUTOMATA WITH Z NOTATION

Interface automata and Z seem in all ways to complement each other in their capabilities. Interface automata can characterise precisely the behavioural aspects of a system, whereas they are not suitable for modelling concisely (abstractly)

the system data structures. On the other hand, Z has great expressive power to describe abstract data structures but lack the notion of operation evaluation order. Currently, there are a lot of language integration proposals. Some examples are LOTOS [2], temporal logic and CSP [?], LOTOS and Z [4], and CSP-Z and CSP-OZ [7], [8], [9]. But temporal logic and CSP are theoretical models and most programmer are not familiar to such specification models. This paper is based on ZIA, a specification language which integrates interface automata and Z. ZIA is defined such that apart from enabling one to deal with the behavioural and the data structure aspects of a system independently.

In this section, we combine inference automata and Z language to give a specification approach for software components. We first give the definition of such model, named ZIA. Then we define the refinement of ZIA.

### A. Model

Interface automata provide a specification approach for interface behavior properties. But this approach can not describe data structures specification of states. On the other hand, Z can specify the state of a system, but is not suitable to behavioural properties. In this section, ZIA is presented, which is a conservative extension of both interface automata and Z in the sense that almost all syntactical and semantical aspects of interface automata and Z are preserved.

In the original interface automata, states and operations are abstract atomic symbols. But in ZIA, states and operations are described by Z schemas.

In the rest of this paper, we use the following terminology:

- (1) a state schema is a schema which does not contain any variable with decoration ';
- (2) an input operation schema is an operation schema which contains input variables;
- (3) an output operation schema is an operation schema which contains output variables;
- (4) an internal operation schema is an operation schema which contains variables with decoration '.

In the rest of paper, given an assignment  $\rho$  and a schema  $A$ , we write  $\rho \models A$  if  $\rho$  assigns every variable  $x$  in the declaration part of  $A$  to an element of its type set, which satisfies the predicate part of  $A$ ; we write  $\models A$  if  $\rho \models A$  for any assignment  $\rho$ .

**Definition 2** An interface automaton with Z notation (ZIA)  $P = \langle S_P, S_P^i, A_P^i, A_P^o, A_P^h, V_P^i, V_P^o, V_P^h, F_P^i, F_P^o, F_P^h, T_P \rangle$  consists of the following elements:

- (1)  $S_P$  is a set of states;
- (2)  $S_P^i \subseteq S_P$  is a set of initial states. If  $S_P^i = \emptyset$  then  $P$  is called empty;
- (3)  $A_P^i, A_P^o$  and  $A_P^h$  are disjoint sets of input, output, and internal actions, respectively. We denote by  $A_P = A_P^i \cup A_P^o \cup A_P^h$  the set of all actions;
- (4)  $V_P^i, V_P^o$  and  $V_P^h$  are disjoint sets of input, output, and internal variables, respectively. We denote by  $V_P = V_P^i \cup V_P^o \cup V_P^h$  the set of all variables;

(5)  $F_P^i$  is a map, which maps any state in  $S_P$  to a state schema in Z language;

(6)  $F_P^o$  is a map, which maps any input action in  $A_P^i$  to an input operation schema in Z language, and maps any output action in  $A_P^o$  to an output operation schema in Z language, and maps any internal action in  $A_P^h$  to an internal operation schema in Z language;

(7)  $T_P$  is the set of transitions between states such that  $T_P \subseteq S_P \times A_P \times S_P$ . If  $(s, a, t) \in T_P$  then  $\models ((F_P^i(s) \wedge F_P^o(a)) \setminus (x_1, \dots, x_m) \Leftrightarrow F_P^i(t) [y'_1/y_1, \dots, y'_n/y_n])$ , where  $\{x_1, \dots, x_m\}$  is the set of the variables in  $F_P^i(s)$ ,  $\{y_1, \dots, y_n\}$  is the set of the variables in  $F_P^i(t)$ , the set of variables in  $F_P^o(a)$  is the subset of  $\{x_1, \dots, x_m\} \cup \{y'_1, \dots, y'_n\}$ .

An action  $a \in A_P$  is enabled at a state  $s \in V_P$  if there is a step  $(s, a, s') \in T_P$  for some  $s' \in S_P$ . We indicate by  $A_P^i(s), A_P^o(s), A_P^h(s)$  the subsets of input, output and internal actions that are enabled at the state  $s$  and we let  $A_P(s) = A_P^i(s) \cup A_P^o(s) \cup A_P^h(s)$ .

Intuitively, for any state  $s, F_P^i(s)$  specifies the data structure properties of all the variables in the state  $s$ ; for any action  $a, F_P^o(a)$  specifies the data structure properties of all the variables before and after performing action  $a$ . An input action  $a$  inputs all the input variables in the Z schema  $F_P^i(a)$ , and an output action  $a$  outputs all the output variables in the Z schema  $F_P^o(a)$ .

In a ZIA, the case  $F_P^i(s) \models \perp$  for a state  $s$ , or the case  $F_P^o(a) \models \perp$  for an action  $a$  is not prohibited. This is similar to the case of Z notation: schema  $[x : N \mid \perp]$  can not be implemented, but this schema is permitted.

### B. An Example

In this section, we give an example to demonstrate ZIA.

An interface automaton with Z notation  $P = \langle S_P, S_P^i, A_P^i, A_P^o, A_P^h, V_P^i, V_P^o, V_P^h, F_P^i, F_P^o, F_P^h, T_P \rangle$  consists of the following elements:

- (1)  $S_P = \{0, 1, 2, 3\}$ ;
- (2)  $S_P^i = \{0\}$ ;
- (3)  $A_P^i = \{a, c\}, A_P^o = \{d\}$  and  $A_P^h = \{b\}$ ;
- (4)  $V_P^i = \{x_1, x_2\}, V_P^o = \{y\}$  and  $V_P^h = \{z\}$ ;
- (5)  $F_P^i(0) = S_0 \hat{=} [z : N \mid z = 0], F_P^i(1) = S_1 \hat{=} [x_1? : N; z : N \mid x_1? \in \{0, \dots, 9\}; z = 0], F_P^i(2) = S_2 \hat{=} [x_1? : N; z : N \mid x_1? \in \{0, \dots, 9\}; z = x_1?], F_P^i(3) = S_3 \hat{=} [x_1? : N; x_2? : N; y! : N; z : N \mid x_1? \in \{0, \dots, 9\}; x_2? \in \{0, \dots, 6\}; z = x_1?];$
- (6)  $F_P^o(a) = A_a \hat{=} [x_1? : N \mid x_1? \in \{0, \dots, 9\}], F_P^o(b) = A_b \hat{=} [x_1? : \{0, \dots, 9\}; z : N \mid z' = z + x_1?], F_P^o(c) = A_c \hat{=} [x_2? : N \mid x_2? \in \{0, \dots, 6\}], F_P^o(d) = A_d \hat{=} [x_2? : N; y! : N; z : N \mid y! = z * x_2?];$
- (7)  $T_P = \{(0, a?, 1), (1, b, 2), (2, c?, 3), (3, d!, 0)\}$ .

This ZIA is given in figure 1.

### C. Labelled Refinement

The refinement relation aims at formalizing the relation between abstract and concrete versions of the same component for example between an interface specification and its implementation.

We first give the following definition which describes the set of states after performing a sequence of internal actions from a given state.

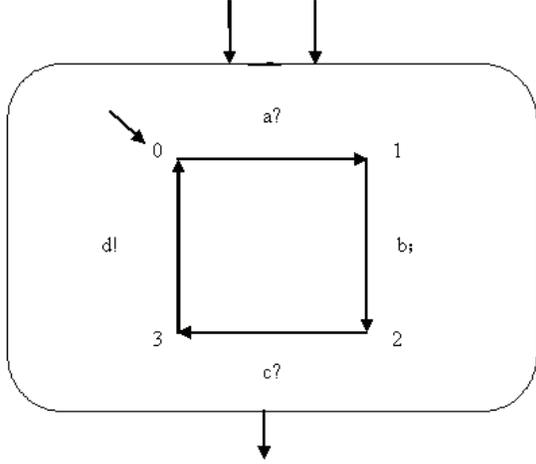


Figure 1: A ZIA

**Definition 3** Given a ZIA  $P$  and a state  $s \in S_P$  the set  $\varepsilon$ -closure $_P(s)$  is the smallest set  $U \subseteq S_P$  such that (1)  $s \in U$  and (2) if  $t \in U$  and  $(t, a, t^*) \in T_P^H$  then  $t^* \in U$ .

The following definition describes the set of states after performing several internal actions and an external action from a given state.

**Definition 4** Consider a ZIA  $P$  and a state  $s \in S_P$ . For an action  $a$ , we let

$ExtDest_P(s, a) = \{s^* \mid \exists (t, a, t^*) \in T_P. t \in \varepsilon$ -closure $_P(s)$  and  $s^* \in \varepsilon$ -closure $_P(t^*)\}$ .

In the following, we use  $V^I(A)$  ( $V^I(B)$ ) to denote the set of input variables in Z schema  $A$  ( $B$ ),  $V^O(A)$  ( $V^O(B)$ ) to denote the set of output variables in Z schema  $A$  ( $B$ ), and  $V^H(A)$  ( $V^H(B)$ ) to denote the set of internal variables in Z schema  $A$  ( $B$ ).

In order to define the refinement relation between Z schemas, we need the following notation.

**Definition 5** Consider two Z schemas  $A$  and  $B$  with  $V^I(A) = V^I(B)$ ,  $V^O(A) = V^O(B)$  and  $V^H(A) = V^H(B) = \emptyset$ . We use the notation  $A \geq B$  if one of the following cases holds:

(1) if  $V^I(A) \neq \emptyset$  and  $V^O(A) \neq \emptyset$  then given an assignment  $\rho$  on  $V^I(A)$ , for any assignment  $\sigma$  on  $V^O(A)$ ,  $\rho \cup \sigma \models B$  implies  $\rho \cup \sigma \models A$ , and given an assignment  $\sigma$  on  $V^O(A)$ , for any assignment  $\rho$  on  $V^I(A)$ ,  $\rho \cup \sigma \models A$  implies  $\rho \cup \sigma \models B$ ;

(2) if  $V^I(A) \neq \emptyset$  and  $V^O(A) = \emptyset$  then for any assignment  $\rho$  on  $V^I(A)$ ,  $\rho \models A$  implies  $\rho \models B$ ;

(3) if  $V^I(A) = \emptyset$  and  $V^O(A) \neq \emptyset$  then for any assignment  $\rho$  on  $V^O(A)$ ,  $\rho \models B$  implies  $\rho \models A$ ;

(4)  $V^I(A) = \emptyset$  and  $V^O(A) = \emptyset$ .

Intuitively,  $A \geq B$  means that schemas  $A$  and  $B$  have the same input variables and the same output variables, and schema  $B$  has bigger domains of input variables but smaller ranges of output variables than schema  $A$ . For example,  $A \hat{=} [x? : N; y! : R \mid y! = 2x?] \geq B \hat{=} [x? : R; y! : N \mid$

$y! = 2\lfloor x? \rfloor]$ , where  $N$  is the set of natural numbers,  $R$  is the set of reals, and  $\lfloor x? \rfloor$  is the largest natural number that is not larger than  $x?$ .

Now we give the refinement relation between Z schemas, which describe the refinement relation between data structures properties of states. Roughly speaking, for two Z schemas  $A$  and  $B$ , we say that  $B$  refines  $A$  if the input variables and the output variables in  $A$  are also in  $B$ , and schema  $B$  has bigger domains on these input variables but smaller ranges on these output variables than schema  $A$ .

**Definition 6** Consider two Z schemas  $A$  and  $B$ , we use the notation  $A \geq B$  if

(1)  $V^I(A) \subseteq V^I(B)$ ,  $V^O(A) \subseteq V^O(B)$ ;

(2)  $A \setminus (x_1, \dots, x_m) \geq B \setminus (y_1, \dots, y_n)$ , where  $V(A) = \{x_1, \dots, x_m\} \uplus V^I(A) \uplus V^O(A)$ ,  $V(B) = \{y_1, \dots, y_n\} \uplus V^I(A) \uplus V^O(A)$ .

For example,  $A \hat{=} [x? : N; y! : R \mid y! = 2x?] \geq B \hat{=} [x? : R; u? : R; y! : N; v! : R; z : N \mid y! = 2\lfloor x? \rfloor; v! = z * u?]$ .

In the following, we give a refinement relation between interface automata, which is defined in the style of a simulation on interface automata. Roughly speaking, this refinement relation is composed of a refinement relation between behavioural properties and a refinement relation between data structures properties.

**Definition 7** Consider two ZIAs  $P$  and  $Q$ . A binary relation  $\succeq_{lt} \subseteq S_P \times S_Q$  is a simulation from  $Q$  to  $P$  if for all states  $p \in S_P$ , there exists  $q \in S_Q$  such that  $p \succeq_{lt} q$  the following conditions hold:

(1)  $F_P^V(p) \geq F_Q^V(q)$ ;

(2) For any internal action  $a$ , if  $(p, a, p^*) \in T_P$ , there is a state  $q^* \in \varepsilon$ -closure $_Q(q)$ , such that  $F_P^V(p^*) \geq F_Q^V(q^*)$ , and  $p^* \succeq_{lt} q^*$ ;

(3) For any input action  $a$ , if  $(p, a, p^*) \in T_P$ , there is a state  $q^* \in ExtDest_Q(q, a)$ , such that  $F_P^A(a) \geq F_Q^A(a)$ ,  $F_P^V(p^*) \geq F_Q^V(q^*)$ , and  $p^* \succeq_{lt} q^*$ ;

(4) For any output action  $a$ , if  $(p, a, p^*) \in T_P$ , there is a state  $q^* \in ExtDest_Q(q, a)$ , such that  $F_P^A(a) \geq F_Q^A(a)$ ,  $F_P^V(p^*) \geq F_Q^V(q^*)$ , and  $p^* \succeq_{lt} q^*$ .

**Definition 8** The ZIA  $Q$  refines the ZIA  $P$  written  $P \succeq_{LT} Q$  if

There is a simulation  $\succeq_{LT}$  from  $Q$  to  $P$ , a state  $p \in S_P^i$  and a state  $q \in S_Q^i$  such that  $p \succeq_{lt} q$ .

## V. CHECKING REFINEMENT RELATION OF ZIAs

In this section we give a refinement checking algorithm for ZIAs. The refinement checking problem for ZIAs asks, given two ZIAs  $P$  and  $Q$ , whether  $P \succeq_{LT} Q$ .

### A. ZIAs With Finite Domain

In a ZIA, each state and each action are assigned to a schema in Z language, which can be regarded as a first logic formula. By the definition of refinement relation for ZIAs, to verify  $P \succeq_{LT} Q$ , one should verify logical implication of two Z schemas. Therefore, in general, the refinement checking problem for ZIAs is undecidable since first order logic is undecidable. But if we consider some decidable sublogics,

such as description logic and proposition logic, the refinement checking problem for ZIAs may become decidable.

In this section, we will introduce a class of ZIAs, for which refinement checking problem is decidable. Roughly speaking, in a ZIA, each state and each action are assigned to a schema in Z language. If every variable in any schema has finite possible values, such ZIAs are called ZIAs with finite domain. Our refinement checking algorithm will work on such ZIAs.

**Definition 9** A Z schema  $S \hat{=} [v_1 : T_1; \dots; v_m : T_m \mid P_1; \dots; P_n]$  is called a Z schema with finite domain, if each variable  $v_i$  has finite possible values, i.e., each type  $T_i$  has finite elements.

**Definition 10** A ZIA  $P = \langle S_P, S_P^i, A_P^I, A_P^O, A_P^H, V_P^I, V_P^O, V_P^H, F_P^V, F_P^A, T_P \rangle$  is called a ZIA with finite domain, if the following condition holds:

- (1) for each  $s \in S_P$ ,  $F_P^V(s)$  is a Z schema with finite domain;
- (2) for each  $a \in A_P$ ,  $F_P^A(a)$  is a Z schema with finite domain.

An example of ZIA with finite domain is given as follows:

$P = \langle S_P, S_P^i, A_P^I, A_P^O, A_P^H, V_P^I, V_P^O, V_P^H, F_P^V, F_P^A, T_P \rangle$  consists of the following elements:

- (1)  $S_P = \{0, 1, 2, 3\}$ ;
- (2)  $S_P^i = \{0\}$ ;
- (3)  $A_P^I = \{a, c\}$ ,  $A_P^O = \{d\}$  and  $A_P^H = \{b\}$ ;
- (4)  $V_P^I = \{x_1, x_2\}$ ,  $V_P^O = \{y\}$  and  $V_P^H = \{z\}$ ;
- (5)  $F_P^V(0) = S_0 \hat{=} [z : \{0, \dots, 9\} \mid z = 0]$ ,  $F_P^V(1) = S_1 \hat{=} [x_1? : \{0, \dots, 9\}; z : \{0, \dots, 9\} \mid x_1? \in \{0, \dots, 4\}; z = 0]$ ,  $F_P^V(2) = S_2 \hat{=} [x_1? : \{0, \dots, 9\}; z : \{0, \dots, 9\} \mid x_1? \in \{0, \dots, 4\}; z = x_1?]$ ,  $F_P^V(3) = S_3 \hat{=} [x_1? : \{0, \dots, 9\}; x_2? : \{0, \dots, 9\}; y! : \{0, \dots, 9\}; z : \{0, \dots, 9\} \mid x_1? \in \{0, \dots, 4\}; x_2? \in \{0, \dots, 4\}; z = x_1?]$ ;
- (6)  $F_P^A(a) = A_a \hat{=} [x_1? : \{0, \dots, 9\} \mid x_1? \in \{0, \dots, 4\}]$ ,  $F_P^A(b) = A_b \hat{=} [x_1? : \{0, \dots, 9\}; z : \{0, \dots, 9\} \mid z' = (z + x_1?) \bmod 10]$ ,  $F_P^A(c) = A_c \hat{=} [x_2? : \{0, \dots, 9\} \mid x_2? \in \{0, \dots, 6\}]$ ,  $F_P^A(d) = A_d \hat{=} [x_2? : \{0, \dots, 9\}; y! : N; z : N \mid y! = (z * x_2?) \bmod 10]$ ;
- (7)  $T_P = \{(0, a?, 1), (1, b, , 2), (2, c?, 3), (3, d!, 0)\}$ .

## B. Algorithm

In this section we presented an algorithm  $RC$  for checking refinement relation over ZIAs with finite domain.

Suppose  $P$  and  $Q$  are two ZIAs with finite domain, the algorithm is given as follows:

$$RC(P, Q) =$$

for each  $p \in S_P^i, q \in S_Q^i$   
 $A_{p,q} := RCS(p, q)$   
 $return(\bigvee_{p,q} A_{p,q})$

$$RCS(p, q) =$$

$B_{p,q} := RCZ(F_P^V(p), F_Q^V(q))$   
 if  $\bigwedge_{a \in IA(p,q) \cup EA(p,q)} (p \xrightarrow{a})$  then  $return(B_{p,q})$   
 else  $B := \bigwedge_{a \in IA(p,q) \cup EA(p,q)} Match_a(p, q)$   
 $return(B \wedge B_{p,q})$

$$Match_{a \in IA(p,q)}(p, q) =$$

if  $(p \xrightarrow{a})$  and  $(q \not\xrightarrow{a})$  then  $return(false)$   
 for each  $(p \xrightarrow{a} p_i)$  and  $(q \xrightarrow{a} q_j)$   
 $C_{i,j} := RCZ(F_P^V(p_i), F_Q^V(q_j))$

$$D_{i,j} := RCS(p_i, q_j)$$

$$return(\bigwedge_i (\bigvee_j (C_{i,j} \wedge D_{i,j})))$$

$$Match_{a \in EA(p,q)}(p, q) =$$

if  $(p \xrightarrow{a})$  and  $(q \not\xrightarrow{a})$  then  $return(false)$   
 $C_a = RCZ(F_P^A(a), F_Q^A(a))$   
 for each  $(p \xrightarrow{a} p_i)$  and  $(q \xrightarrow{a} q_j)$   
 $C_{i,j} := RCZ(F_P^V(p_i), F_Q^V(q_j))$   
 $D_{i,j} := RCS(p_i, q_j)$   
 $return(\bigwedge_i (\bigvee_j (C_a \wedge C_{i,j} \wedge D_{i,j})))$

$$RCZ(S, T) =$$

if  $(V^I(S) \not\subseteq V^I(T))$  or  $(V^O(S) \not\subseteq V^O(T))$  then  
 $return(false)$   
 else  
 $V_S := V(S) - V^I(S) - V^O(S)$   
 $V_T := V(T) - V^I(T) - V^O(T)$   
 $E := RCL(S \setminus V_S, T \setminus V_T)$   
 $return(E)$

$$RCL(M, N) =$$

if  $(V^I(M) \neq V^I(N))$  or  $(V^O(M) \neq V^O(N))$  or  
 $(V^H(M) \neq \emptyset)$  or  $(V^H(N) \neq \emptyset)$  then  
 $return(false)$   
 if  $(V^I(M) = \emptyset)$  and  $(V^O(M) = \emptyset)$  then  
 $return(true)$   
 if  $(V^I(M) = \emptyset)$  and  $(V^O(M) \neq \emptyset)$  then  
 $return(TV(N \Rightarrow M))$   
 if  $(V^I(M) \neq \emptyset)$  and  $(V^O(M) = \emptyset)$  then  
 $return(TV(M \Rightarrow N))$   
 if  $(V^I(M) \neq \emptyset)$  and  $(V^O(M) \neq \emptyset)$  then  
 $return(TV(\forall v_1^I : V_1^I; \dots; v_m^I : V_m^I \bullet$   
 $(N \Rightarrow M) \wedge \forall v_1^O : V_1^O; \dots; v_n^O : V_n^O \bullet$   
 $(M \Rightarrow N)))$   
 where  $V^I(M) = \{v_1^I, \dots, v_m^I\}$ ,  $V^O(M) =$   
 $\{v_1^O, \dots, v_n^O\}$ , the type of  $v_k^I$  is  $V_k^I$ ,  
 and the type of  $v_k^O$  is  $V_k^O$ .

$$TV(LS) =$$

rewrite schema  $LS$  to an equivalent first order  
 logical formula  $LF$   
 if  $(LF$  is always true for any assignment on  
 variables) then  $return(true)$   
 else  $return(false)$

Suppose that  $p$  and  $p'$  are states of  $P$ , and  $a$  is an action of  $P$ , we use the notation  $p \xrightarrow{a} p'$  to denote  $(p, a, p') \in T_P$ , the notation  $p \xrightarrow{a} p'$  to denote  $p' \in ExtDest_P(p, a)$ , and the notation  $p \xRightarrow{a} p'$  to denote  $p' \in \varepsilon - closure_P(p)$ .  $IA(p) = \{a \mid \exists p'. p \xrightarrow{a} p' \text{ and } a \text{ is an internal action.}\}$ .  $EA(p) = \{a \mid \exists p'. p \xrightarrow{a} p' \text{ and } a \text{ is an external action.}\}$ .  $IA(p) \cup IA(q)$  is abbreviated as  $IA(p, q)$ .  $EA(p) \cup EA(q)$  is abbreviated as  $EA(p, q)$ . We use  $p \xrightarrow{a}$  to represent there exists  $p'$  such that  $p \xrightarrow{a} p'$ , and  $q \not\xrightarrow{a}$  means there is no  $q'$  such that  $q \xrightarrow{a} q'$ . Since there are finite states in a ZIA,  $p \xrightarrow{a}$  and  $p \xRightarrow{a}$  is decidable.

In the above algorithm, the function  $TV(LS)$  returns *true* if  $LS$  is always true for any assignment, otherwise returns *false*. In general,  $TV(LS)$  can not be implemented since the tautology

problem of first order logic is not decidable. But if the logic is restricted to some decidable sublogics, for example, each variable of a logical formula has finite possible values, the tautology problem of such logic becomes decidable.  $TV(LS)$  is decidable because  $LS$  is a Z schema with finite domain, and there are only finite possible assignments on variables.

The function  $RCS(p, q)$  starts with the initial pair  $(p, q)$ , trying to check the similarity of  $p$  and  $q$  by matching transitions from them. While travelling the transition graph, at each pair of nodes the algorithm produces the outgoing transitions and next states according to the transition of ZIA. The transitions are then matched for simulation, and the algorithm goes on to the new state pairs if the matches are successful.

The function  $Match_a$ , performs a depth-first search on the product of the two labelled transition graphs. If one state fails to match another's transitions then they are not refinement and return *false*, otherwise return *true*.

### C. The Correctness of RC

The correctness of the algorithm for refinement relation is not difficult to justify. Each call of  $match_a(p, q)$  performs a depth-first search in the product graph of the two transition graphs. This ensures that  $match_a(p, q)$  can only be called for finitely many times since the states spaces of  $P$  and  $Q$  are finite. Therefore  $RC(P, Q)$  will always terminate. Furthermore we can give the correctness of the above algorithm:

**Lemma 1** The algorithm  $RCZ(S, T)$  given in the above terminates and is correct, i.e., it returns true iff  $S \triangleright T$ .

**Lemma 2** The algorithm  $RCS(p, q)$  given in the above terminates and is correct, i.e., it returns true iff  $p \succeq_H q$ .

By the above Lemmas, we have the following proposition:

**Proposition 1** The algorithm  $RC(P, Q)$  given in the above terminates and is correct, i.e., it returns true iff  $P \succeq_{LT} Q$ .

## VI. CONCLUSIONS

In this paper, we introduced a combination of interface automata and Z called ZIA, which can be applied to specify properties of behaviour and data structures of a system. We then defined the refinement relation for ZIA. At last, to verify systems, we provided a refinement checking algorithm for ZIAs with finite domain. It provides techniques to verify properties of data structures and control aspects in a common framework.

## ACKNOWLEDGMENT

This work was supported by the Aviation Science Fund of China under Grant No. 20085552023, the National Natural Science Foundation of China under Grants No. 60473036, No. 60873025, the Natural Science Foundation of Jiangsu Province of China under Grant No. BK2008389, and the Foundation of Provincial Key Laboratory for Computer Information Processing Technology of Soochow University under Grant No. KJS0920.

## REFERENCES

- [1] L. de Alfaro, T. A Henzinger. Interface Automata. In the Proceedings of the 9th Annual ACM Symposium on Foundations of Software Engineering (FSE), 2001.
- [2] T. Bolognesi and E. Brinksma. Introduction to the ISO specification language LOTOS. Computer Networks and ISDN Systems, 14(1): 25-59, 1987.
- [3] J. Bowen. Formal Specification and Documentation using Z: A Case Study Approach. 2003.
- [4] J. Derrick, E. Boiten, H. Bowman and M. Steen. Viewpoint consistency in Z and LOTOS: A case study. In FME'97: pp.644-664. Springer-Verlag, 1997.
- [5] M. Emmi, D. Giannakopoulou, and C. S. Pasareanu. Assuma-Guarantee Verification for Interface Automata. FM 2008: Formal Methods, Lecture Notes in Computer Science 5014, 116-131, 2008.
- [6] S. Esmacilsabzali, F. Mavaddat, N. A. Day. Interface Automata with Complex Actions. In Proceeding of the First IPM International Workshop on Foundations of Software Engineering (FSEN), pp. 79-97, 2006
- [7] C. Fischer. Combining CSP and Z. Technical report, TRCF-97-1, University of Oldenburg, 1996.
- [8] C. Fischer. CSP-OZ: A combination of Object-Z and CSP. In FMOODS'97, Chapman Hall, 1997.
- [9] C. Fischer. How to combine Z with a process algebra. In ZUM'98, LNCS 1493, pp 5-23, Springer-Verlag, 1998.
- [10] C. A. R. Hoare. Communicating Sequential Processes. Prentice-Hall, 1985.
- [11] B. Mahony and J. S. Dong. Overview of the Semantics of TCOZ. Integrated Formal Methods (IFM'99), pages 66-85, Springer-Verlag, 1999.
- [12] A. Mota and A. Sampaio. Model-Checking CSP-Z: Strategy, Tool Support and Industrial Application. Science of Computer Programming, v. 40, n. 3, p. 59-96, 2001.
- [13] J. M. Spivey. The Z Notation: A Reference Manual. Second Edition. Prentice Hall International (UK) Ltd.
- [14] J. Woodcock and J. Davies. Using Z - Specification, Refinement, and Proof. Prentice-Hall, 1996.

# Multi-objective Genetic Algorithms: Construction and Recombination of Passive Testing Properties

César Andrés, Mercedes G. Merayo, Manuel Núñez

Departamento de Sistemas Informáticos y Computación

Universidad Complutense de Madrid, Spain

e-mails: c.andres@fdi.ucm.es, mgmerayo@fdi.ucm.es, mn@sip.ucm.es

**Abstract**—Passive testing uses the historical interaction files of systems in order to decide their correctness. In our testing approach, we represent the most important properties to be checked by using so-called *invariants*. Normally, a tester introduces the set of invariants to perform this task, but in many cases the size of this set is too small. This paper presents a way to provide a bigger set of representative invariants. This novel approach is based on applying genetic algorithms in a small set of invariants in order to generate a representative bigger invariant suite.

**Index Terms**—Passive Testing, Genetic Algorithms

## I. INTRODUCTION

The complexity of current systems, the large number of people working on them, and the number of different modules that interact with each other, make it difficult to decide their correctness. *Testing techniques* [1], [2] allow us to provide a degree of confidence in the correctness of a system. These techniques can be combined with the use of formal methods [3], [4], [5], [6], [7] in order to semi-automatically perform some tasks involved in testing through the use of tools [8]. The application of formal testing techniques to check the correctness of a system requires to identify its *critical* aspects, that is, those characteristics that will make the difference between correct and incorrect behaviors. While the relevant aspects of some systems only concern *what* they do, in some other systems it is equally relevant *how* they do what they do. Thus, during the last years formal testing techniques also dealt with non-functional properties. In this paper we focus on systems that contain temporal restrictions, being already several proposals for timed testing (e.g. [9], [10], [11], [12]).

In testing, we can distinguish between two approaches: *Passive* and *Active*. The main difference between them is whether a tester can interact with the System Under Test (SUT), or not. If the tester can interact with the SUT, then we are in the active testing paradigm. On the contrary, if the tester simply monitors the behavior, then we are in the passive testing paradigm. Actually, it is very frequent that the tester is unable to interact with the SUT. In particular, such interaction can be difficult in the case of large systems working 24/7 since this interaction might produce a wrong behavior of the system. For example, let us consider the task of testing a bank account

manager. A possible test suite to check the correctness of this system might be:

- 1) *Create a new account.*
- 2) *Change the ID of a user of the system.*
- 3) *Insert \$1.000.000 in the account XXX.YYY.ZZZ.*
- 4) *Delete a user.*

Since it is very risky to perform these actions in systems that are already working, testers might use passive testing techniques in order to test these systems.

In our formal passive testing framework [13], [14], we consider the following scenario. First, we are provided with a formal *specification* of the system. In this specification all the correct behaviours of the system are represented. We also consider that we are provided with a set of formal requirements, that we will call *invariants*. They represent the most expected properties to be checked in the SUT.

Usually we consider different ways to obtain a set of invariants. In our original approach we assumed that this set is given by the tester. In this case, before using the invariants to check the correctness of the system, we have to check their correctness, with respect to the specification, since it makes no sense to check the correctness of a system with respect to wrong properties. After strong experimentation with this first approach, we detected the following two problems: The set of invariants might be erroneous and the size of the set of invariants was too small. However, the current availability of computational resources allows us to simultaneously check huge set of invariants without sensibly decreasing performance: Normally, testers do not generate “from scratch” big sets of invariants. In this line, in [15] we showed how to use the specification of the system to provide the tester with a set of invariants. We presented an algorithm that automatically derives a set of invariants. Unfortunately, the set of derived invariants was so big, sometimes infinite, and it was non-representative. By non-representative we mean that we were unable to establish any criterion to compare invariants, for example, to decide the set containing the *best n* invariants. In order to solve this drawback, in [16] we assumed the hypothesis of having knowledge about the system concerning the past-interactions of the users with the SUT. This information was collected in a database. We applied data mining techniques in order to generate a probabilistic model [17] to describe the users of the system. We adapted the algorithm of [15] to use

Research partially supported by the WEST project (TIN2006-15578-C02) and the UCM-BSCH programme to fund research groups (GR58/08 group number 910606).

this new information. The underlying idea was to guide the invariant generation in order to check the most frequent user sequence actions presented in the probabilistic model.

Unfortunately, we consider that this is neither the unique nor the optimum way to provide a tester with a set of invariants. Within the last approach we generated the properties based on the most common input sequences performed by the users of this SUT. Let us consider that we have produced several copies of the same SUT, they are placed in different locations, and we want to generate a representative invariant suite to test all of them. Usually, users that are interacting with a SUT do not interact with another one. In our previous approach, we would generate a user model for each system, but we were not able to generate a unique invariant suite, sharing the information presented in each user model. In this paper we present a novel methodology, based on *genetic algorithms*, to cross the different sets of invariants in order to generate a *powerful* set of representative invariants with respect to any user model. Let us remark that the use of genetic algorithms applied to formal testing tasks is not new (see, for example, [18], [19], [20], [21]). Therefore, we consider that it is a feasible way to work with them.

The rest of the paper is structured as follows. Section II presents our formal framework. In Section III we present the use of genetic algorithms in order to provide a set of representative invariants. Finally, in Section IV we give our conclusions and some lines for future work.

## II. FORMAL FRAMEWORK

In this section we introduce our framework to specify and (passively) test timed systems. We extend the well-known Finite State Machines model by adding time information concerning the amount of time that the system needs to perform transitions.

*Definition 1:* A *Timed Finite State Machine (TFSM)* is a tuple  $(\mathcal{S}, s_0, \mathcal{I}, \mathcal{O}, \mathcal{T})$ , where  $\mathcal{S}$  is a finite set of states,  $s_0 \in \mathcal{S}$  is the initial state,  $\mathcal{I}$  and  $\mathcal{O}$ , with  $\mathcal{I} \cap \mathcal{O} = \emptyset$ , are the finite sets of *input* and *output* actions, respectively, and  $\mathcal{T} \subseteq \mathcal{S} \times \mathcal{I} \times \mathcal{O} \times \mathbf{R}_+ \times \mathcal{S}$  is the set of transitions.

A TFSM  $M = (\mathcal{S}, s_0, \mathcal{I}, \mathcal{O}, \mathcal{T})$  is *deterministic* if for all state  $s$  and input  $i$  there exists at most one transition  $(s, i, o, t, s')$ .  $\square$

Let us consider the TFSM depicted in Figure 1 and the transition  $(s_1, a, x, 3, s_2)$ . Intuitively, if the machine is at state  $s_1$  and it receives the input  $a$ , then it will produce the output  $x$  after 3 time units. We usually write  $s \xrightarrow{i/o}_t s'$  as a shorthand of  $(s, i, o, t, s') \in \mathcal{T}$ . Finally, we assume that specifications are given by deterministic TFSMs.

Next we introduce the notion of *trace* of a system. A trace captures the behaviour of an implementation. Traces collect the outputs obtained after sending some inputs to the implementation and the amount of time between each input/output pair. The classical notion of trace, which does not include any time information, is called a *non-timed trace*.

*Definition 2:* Let  $M = (\mathcal{S}, s_0, \mathcal{I}, \mathcal{O}, \mathcal{T})$  be a TFSM. We say that  $\langle i_1/o_1/t_1, i_2/o_2/t_2, \dots, i_n/o_n/t_n \rangle$  is a *timed trace*,

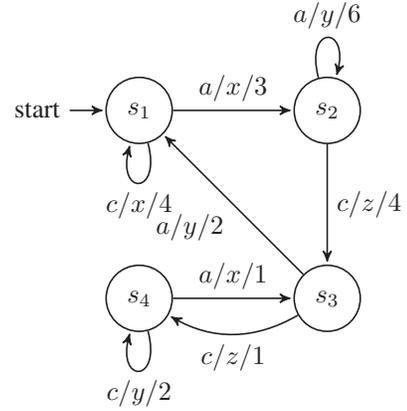


Figure 1. Example of TFSM.

or simply trace, of  $M$  if there is a sequence of transitions such that

$$s_1 \xrightarrow{i_1/o_1} t_1 s_2 \xrightarrow{i_2/o_2} t_2 s_3 \dots s_n \xrightarrow{i_n/o_n} t_n s_{n+1}$$

If  $\langle i_1/o_1/t_1, \dots, i_n/o_n/t_n \rangle$  is a timed trace of  $M$ , then the sequence  $\langle i_1/o_1, \dots, i_n/o_n \rangle$  is a *non-timed trace* of  $M$ .  $\square$

For example, if we consider the machine  $M$  depicted in Figure 1, we have that  $\langle c/x/4, c/x/4, a/x/3, c/z/4, c/z/1, a/x/1 \rangle$  is a timed trace of  $M$ . If we remove time information, we obtain *non-timed* traces. For instance,  $\langle c/x, c/x, a/x, c/z, c/z, a/x \rangle$  is the non-timed trace associated with the previous timed trace. SUTs are implemented as black boxes and our only assumption about them is that they produce sequences that can be converted into timed traces.

Our passive testing approach is similar to other methodologies since the basic idea consists in recording traces from the SUT to detect unexpected behaviours. The main novelty is that a set of *invariants* is used to represent the most relevant properties of the specification. Intuitively, an invariant expresses the fact that each time the SUT performs a given sequence of actions, then it must exhibit a behaviour reflected in the invariant. In order to express traces in a concise way, we will use the wild-card characters  $?$  and  $*$ . The wild-card  $?$  represents any value in the sets of inputs or outputs, while  $*$  represents a sequence of input/output pairs.

*Definition 3:* We say that  $\hat{p} = [p_1, p_2]$  is a *time interval* if  $p_1 \in \mathbf{R}_+$ ,  $p_2 \in \mathbf{R}_+ \cup \{\infty\}$ , and  $p_1 \leq p_2$ . We assume that for all  $t \in \mathbf{R}_+$  we have  $t < \infty$  and  $t + \infty = \infty$ . We consider that  $\mathcal{IR}$  denotes the set of time intervals.

Let  $M = (\mathcal{S}, s_0, \mathcal{I}, \mathcal{O}, \mathcal{T})$  be a TFSM. We say that the sequence  $\phi$  is an *invariant* for  $M$  if the following two conditions hold:

1)  $\phi$  is defined according to the following EBNF:

$$\begin{aligned} \phi &::= \text{Body} \mapsto \text{Consequent} \\ \text{Body} &::= a/z/\hat{p}, \text{Body} \mid \star/\hat{p}, \text{Body}' \mid i \\ \text{Body}' &::= i/z/\hat{p}, \text{Body} \mid i \\ \text{Consequent} &::= O/\hat{p} \triangleright \hat{t} \end{aligned}$$

In this expression we consider  $\hat{p}, \hat{t} \in \mathcal{IR}$ ,  $i \in \mathcal{I}$ ,  $a \in \mathcal{I} \cup \{?\}$ ,  $z \in \mathcal{O} \cup \{?\}$ , and  $O \subseteq \mathcal{O}$ .

2)  $\phi$  is *correct* with respect to  $M$  (formally defined in [13]).  $\square$

Even though, our machines present time information expressed as fix amounts of time, time conditions established in invariants are given by intervals. This fact is due to consider that it can be admissible that the execution of a task sometimes lasts more time than expected: If most of the times the task is performed on time, a small number of delays can be tolerated. Concerning the notion of correctness, the idea is that an invariant is correct with respect to a machine  $M$ , if  $M$  cannot perform a timed trace that would contradict what the invariant expresses.

Intuitively, the previous EBNF expresses that an invariant is either a sequence of symbols where each component but the last one is either an expression  $a/z/\hat{p}$ , with  $a$  being an input action or the wild-card character  $?$ ,  $z$  being an output action or  $?$ , and  $\hat{p}$  being an interval, or an expression  $\star/\hat{p}$ . There are two restrictions to this rule. First, an invariant cannot contain two consecutive expressions  $\star/\hat{p}_1$  and  $\star/\hat{p}_2$ . In the case that such situation was needed to represent a property, the tester could simulate it by means of the expression  $\star/(\hat{p}_1 + \hat{p}_2)$ . The second restriction is that an invariant cannot present a component of the form  $\star/\hat{p}$  followed by an expression beginning with the wild-card character  $?$ , that is, the input of the next component must be a *real* input action  $i \in \mathcal{I}$ . In fact,  $\star$  represents any sequence of inputs/outputs pairs such that the input is not equal to  $i$ .

The last component corresponding to the expression  $i \mapsto O/\hat{p} \triangleright \hat{t}$  is an input action followed by a set of output actions and two timed restrictions, denoted by means of two intervals  $\hat{p}$  and  $\hat{t}$ . The former is associated to the last expression of the sequence. The latter is related to the sum of times values associated to all input/output pairs performed before. For example, the meaning of an invariant as  $i/o/\hat{p}, \star/\hat{p}_*, i' \mapsto O/\hat{p}' \triangleright \hat{t}$  is that if we observe the transition  $i/o$  in a time belonging to the interval  $\hat{p}$ , then the first occurrence of the input symbol  $i'$  after a lapse of time belonging to the interval  $\hat{p}_*$  must be followed by an output belonging to the set  $O$ . The interval  $\hat{t}$  makes reference to the total time that the system must spend to perform the whole trace. Next we introduce some examples in order to present how invariants work.

*Example 1:* Let us consider the TFSM presented in Figure 1. One of the most simple invariants that we can define within our framework is  $a \mapsto \{x, y\}/[1, 6] \triangleright [1, 6]$ . The idea is that each occurrence of the input symbol  $a$  is followed by either the output symbol  $x$  or  $y$ , and this transition is performed between 1 and 6 time units.

We can specify a more complex property by taking into account that we are interested in observing the output  $x$  after the input  $a$  when the pair  $c/x$  was previously observed. In addition, we include time intervals corresponding to the amount of time the system takes for each of the transitions and to the total time it spends in the whole trace. We could express this property by using the following invariant:  $c/x/[3, 5], \star/[0, 5], a \mapsto \{x\}/[2, 4] \triangleright [4, 13]$ . An observed trace will be correct with respect to this invariant if each time that we find a (sub)sequence starting with the  $c/x$  pair, performed in an amount of time belonging to the interval  $[3, 5]$ , if there is an occurrence of the input  $a$  before 5 time units pass, then it must be paired with the output symbol  $x$  and the lapse of time between  $a$  and  $c$  must belong to the interval  $[2, 4]$ . In addition, the whole sequence must take a time belonging to the interval  $[4, 13]$ .  $\square$

Next we elaborate on the notion of *user models*. They indicate the probability that a user chooses each input in each situation. We will use a particular case of Probabilistic Finite State Machine (PFMSM) to represent user models (for technical details, see [17]). Let us note that the interaction of a user with an implementation ends whenever the user decides to stop it. Consequently, models denoting users must represent this event as well. Given a PFMSM representing a user model, we will require that the sum of the probabilities of all inputs associated to a state is lower than (or equal to) 1. The remainder up to 1 represents the probability of stopping the interaction at this state. We will combine user models and non-timed traces of the specification to assess the probability of performing certain sequences of inputs.

### III. GENETIC ALGORITHM

In this section we present our genetic algorithm and how we can use it to generate a new set of invariants. A genetic algorithm [22] is a search technique used in computing to find exact or approximate solutions to optimization and search problems. The *evolution* usually starts from a *population* of randomly generated individuals and happens in *generations*. Thus, in each generation of the algorithm, the fitness of every individual in the population is evaluated. Then, multiple individuals are stochastically selected from the current population, and modified to form a new population. Next, the new population is used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

According to this scheme, the structure of the rest of this section is as follows. In Section III-A we present the initialization of the set of invariants. After that, in Section III-B we present how we compare sets of invariants. Next, in Section III-C the reproduction process is described. Finally, in Section III-D the termination of the algorithm is presented.

#### A. Initialization

In this section we focus on how to get an initial population of invariants. Normally, this initial set contains several hun-

dreds or thousands of possible solutions. Traditionally, this population is randomly generated, but in our approach we make use of the algorithms to extract a set of representative invariants by taking into account the user model that we have, in order to generate a representative seed of the genetic algorithm.

*Definition 4:* Let  $S$  be the specification of a system,  $I_1, \dots, I_n$  be a set of implementations of this system, and  $DB_1, \dots, DB_n$  be a set of  $n$  databases recording the interactions of users with  $I_1, \dots, I_n$ , respectively. We define the initial population with degree  $\delta$  as:

$$\Phi = \bigcup_{i=1}^n \text{generate}(U_i, S, \delta)$$

where  $U_i$  corresponds to the user model generated from database  $DB_i$  and `generate` is the algorithm to extract invariants with the help of a user model presented in [16].  $\square$

## B. Selection

In this section we present how we make a selection of invariants from the existing population. During each successive generation, a proportion of the existing population is selected to breed a new generation. Individual solutions are selected through a *fitness-based* process, where fitter solutions are typically more likely to be selected. In our approach we have two different ways to make the selection of the candidates from the initial state:

- 1) The first way is to randomly generate a set. This helps to keep the diversity of the population large, preventing premature convergence and poor solutions.
- 2) The second way focuses on a tournament selection. In [23] we presented a heuristic approach to compare different sets of invariants. This approach is based on mutation testing techniques and the invariants are classified with respect to the probability to detect an erroneous behavior in different environments.

*Definition 5:* Let  $S$  be the specification of a system,  $I_1, \dots, I_n$  be a set of implementations of this system,  $DB_1, \dots, DB_n$  be a set of  $n$  databases recording the interactions of users with  $I_1, \dots, I_n$ , respectively, and  $\mathcal{U} = \{U_1, \dots, U_n\}$  be the set of user models extracted from these databases. Given an invariant suite  $\Phi$ , we define the fitness function as:

$$\text{fitness}_{\mathcal{U}||S}(\Phi) = \sum_{U \in \mathcal{U}, \phi \in \Phi} \text{prob}_{U||S}(\phi)$$

where  $\text{prob}_{U||S}(\phi)$  returns the degree of representativity to perform all the actions presented in  $\phi$  with respect to the parallel composition of the user model with the specification. This degree is a positive real value.  $\square$

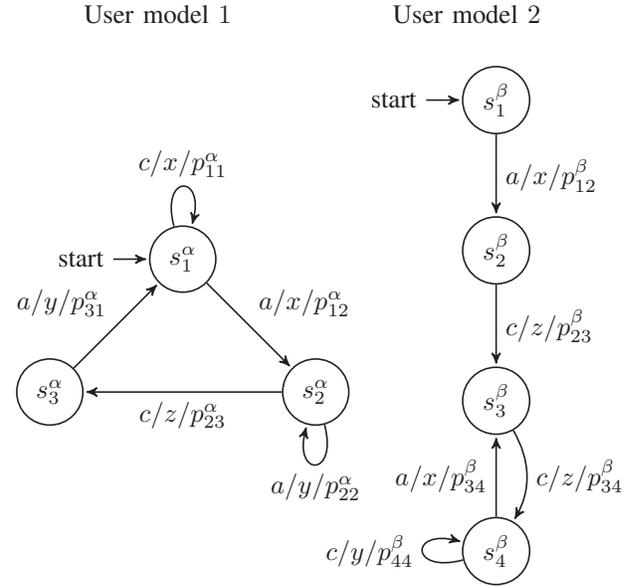


Figure 2. Example of two user models.

## C. Reproduction

The next step is to generate a second generation population of solutions from those selected through the genetic operator of crossover. In some heuristics, they divide the population into a bigger new population in order to increase their number. This approach cannot be used in our paradigm since our set of invariants has always to be correct with respect to the specification; if we divide an invariant we may obtain a set of invariants that might be incorrect.

*Example 2:* Let us consider the specification presented in Figure 1. We have that the invariant  $\phi = a/x/[2.8, 3.1], c \mapsto \{z\}/[3.9, 4, 1] \triangleright [6.8, 7.4]$  is a correct invariant for this specification. It means that if we observe the input  $a$  followed by the output  $x$  within a time belonging to  $[2.8, 3.1]$ , then if we observe the input action  $c$  we have to observe the output  $z$ , within a time belonging to  $[3.9, 4, 1]$ , and the global sum of observed time values belongs to  $[6.8, 7.4]$ .

Let  $\phi_1 = a \mapsto \{x\}/[2.8, 3.1] \triangleright [2.8, 3.1]$  be a reduction of  $\phi$ . We have that  $\phi_1$  is not correct with respect to the specification. Let us note that this invariant contradicts what is represented in the specification since it requires that each time that we observe the input  $a$  we have to observe the output  $x$ , within a time belonging to the interval  $[2.8, 3.1]$ . Taking into account the specification, we are able to produce the following transitions  $s_2 \xrightarrow{a/y/6} s_2, s_3 \xrightarrow{a/y/2} s_1, \text{ or } s_4 \xrightarrow{a/x/y} s_1$  that show the incorrectness of the invariant with respect to the specification.  $\square$

The idea of reproduction is used to represent user behaviours that were not presented into the user models that we have. Let us consider Figure 2 where two users models, for the specification introduced in Figure 1, are given. Let us consider the user model 1. From the first state, that is  $s_1^\alpha$ , it represents that users have three different possibilities. The first

one is that they can perform the input  $c$ , with probability  $p_{11}^\alpha$  returning the system the output  $z$ ; the second one corresponds to perform the input  $a$  with probability  $p_{12}^\alpha$  returning the system the output  $x$ ; the probability to stop in state  $s_1^\alpha$  is equal to  $1 - p_{11}^\alpha - p_{12}^\alpha$ . Taking into account the user model 1, the probability associated to perform the input actions  $\langle c, c, a \rangle$  is  $p = p_{11}^\alpha \cdot p_{11}^\alpha \cdot p_{12}^\alpha \cdot (1 - (p_{23}^\alpha + p_{22}^\alpha))$ . An invariant that checks this property is  $\phi = c/x/[3.9, 4.1], c/x/[3.9, 4.1], a \mapsto \{x\}[2.9, 3.1] \triangleright [10, 12]$ . So, the idea is that this invariant has associated  $p$  as degree of representation. Let us note that the previous invariant has 0 as degree of representation when analyzed with respect to the second user model.

Next, we present our crossover operation. The idea is to split the body of the invariant and combine its prefixes with other invariants by using the  $\star$  operation. This wildcard represents any, possibly empty, sequence of inputs and outputs. Intuitively, by taking into account Figure 2, we want to represent that we might obtain a new user that starts as the users represented in the first model do, and finishes as the behavior presented in the second model.

*Definition 6:* Let  $\phi_1$  and  $\phi_2$  be two invariants and  $S$  be the specification of the system. We define the cross operation of  $\phi_1$  and  $\phi_2$  as:

$$\text{cross}(\phi_1, \phi_2, S) = \text{cross}'(\phi_1, \phi_2, S) \cup \text{cross}'(\phi_2, \phi_1, S)$$

$$\text{cross}'(\phi_1, \phi_2, S) = \bigcup_{\phi \in \text{prefix}(\phi_1)} c_S(\phi, \star/[0, \infty], \text{mult}(\phi_2))$$

where  $\text{prefix}$  will return the set of all prefixes of an invariant,  $\text{mult}$  will replace the last time constraint of a invariant, that is, the one presented after the symbol  $\triangleright$  by  $[0, \infty]$ , and the function  $c_S$  returns  $\phi' = (\phi, \star/[0, \infty], \text{mult}(\phi_2))$  if this invariant is correct with respect to  $S$ ; otherwise it will return the empty set.  $\square$

*Example 3:* Let us consider the following two input sequences  $\langle c, c, c \rangle$  and  $\langle a, c, c \rangle$  performed from the user models presented in Figure 2. With the previous definition, we are able to represent a new invariant suite for checking the following behaviors:  $\langle c, a, c, c \rangle$ ,  $\langle c, c, a, c, c \rangle$ ,  $\langle c, c, c, a, c, c \rangle$ ,  $\langle a, c, c, c \rangle$ ,  $\langle a, c, c, c, c \rangle$ , and  $\langle a, c, c, c, c, c \rangle$  respectively.

In a possible user model generated from the combination of these models, we will have that the probability of checking the sequence,  $\langle c, c, c \rangle$  is  $p_{11}^\alpha \cdot p_{11}^\alpha \cdot p_{11}^\alpha \cdot (1 - (p_{11}^\alpha + p_{12}^\alpha))$ , and the probability of checking the sequence  $\langle a, c, c, c, c, c \rangle$  is  $p_{12}^\beta \cdot p_{23}^\beta \cdot p_{34}^\beta \cdot p_{11}^\alpha \cdot p_{11}^\alpha \cdot p_{11}^\alpha \cdot (1 - (p_{11}^\alpha + p_{12}^\alpha))$ .  $\square$

Due to the following property of real numbers, we suggest that our heuristic crossover function does not use the complete set of prefixes of an invariant to generate a new one. Let us consider  $0 \leq a \leq b \leq 1$  be two real values. We have that  $a \cdot b \leq a$  and  $a \cdot b \leq b$ . This property shows that adding new values to check the correctness of the system, that is, adding new probability values to the chain of input sequences to generate the invariant, will reduce the degree of representativity of the new invariant with respect to its parents. So, just to solve this problem, we suggest not to use the complete set of prefixes of an invariant, but only the prefixes composed within  $\gamma$  values.

*Definition 7:* Let  $\gamma$  be a degree of representativity of the invariant. We redefine the function  $\text{cross}'$  as:

$$\text{cross}'(\phi_1, \phi_2, S) = \bigcup_{\phi \in \text{prefix}_\gamma(\phi_1)} c_S(\phi, \star/[0, \infty], \text{mult}(\phi_2))$$

where  $\text{prefix}_\gamma(\phi_1)$  represents the set of prefix with representativity degree  $\gamma$ .  $\square$

*Example 4:* Let us consider again the previous input sequences  $\langle c, c, c \rangle$  and  $\langle a, c, c \rangle$ . If want to represent the behaviors of the invariant suite after the crossover with a degree equal to 2, then we are able to check the following behaviors:  $\langle c, a, c, c \rangle$ ,  $\langle c, c, a, c, c \rangle$ ,  $\langle c, c, \dots, a, c, c \rangle$ ,  $\langle a, c, c, c \rangle$ ,  $\langle a, c, c, c, c \rangle$ ,  $\langle a, c, \dots, c, c, c, c \rangle$  and  $\langle c, c, \dots, c, c, c, c \rangle$  respectively. We remark that the probability to perform the token  $\dots$  is equal to 1 because it represents any (possible empty) sequence.  $\square$

For each new solution to be produced, a pair of parent solutions is selected for breeding from the previously selected pool. By producing a *child* solution using the above method of crossover, a new solution is created which typically shares many of the characteristics of its *parents*. New parents are selected for each new child and the process continues until a new population of solutions of appropriate size is generated. These processes ultimately result in the next generation population of chromosomes that is different from the initial generation. Generally the average fitness will have increased by this procedure for the population, since only the best invariants from the first generation are selected for breeding, along with a small proportion of less fit solutions.

#### D. Termination

This process of generating a new set of invariants is repeated until a termination condition is reached. Next we present some common terminating conditions for our approach. First, the constructed invariant suite satisfies a minimum criteria provides by a tester. This criteria can be expressed, for example, by a fixing the number of generations reached. Sometimes, in other environments, we may consider that the access to the user models is restricted, and we can access this information a certain number of times. The last terminating condition corresponds to let the tester to check the invariant extraction process and letting her to stop at any moment.

In order to conclude the paper, we informally present the basic steps of our genetic algorithm:

- Choose the initial population of invariants (with  $\delta$  degree of representativity).
- Evaluate the fitness of each invariant in that population.
- Repeat this generation until *termination* (different tester criteria):
  - Select the best-fit individuals for reproduction (by using the definition of *fitness* function).
  - Breed new individuals through crossover to give birth to offsprings (using a degree  $\gamma$  provided by the tester).
  - Evaluate the individual fitness of new invariants.
  - Replace least-fit population with new invariants.

#### IV. CONCLUSIONS AND FUTURE WORK

In this paper we have presented a novel approach to generate a set of invariants to perform passive testing of timed systems. These invariants have been generated by using a genetic algorithm. The initial population of this approach is obtained by applying the algorithm to extract invariants with user models presented in [15], [16]. We showed how we can simulate new users that are not contained into these models and check these properties.

Genetic algorithms are an artificial intelligence technique that uses metaphors of mechanisms present in nature for organisms to develop, adapt, reproduce, and try to adapt to the system in which they act and live. Its main operators are mutation, reproduction (genetic crossing) and selection of the fittest individuals. Genetic algorithms are a good method to use with black-box testing since if we do not have any information regarding the internal structure of the SUT, then the testing problem can be expressed as a search in the space of solutions, guided by a heuristic. In our case, the space of solutions is the set of all possible implementations that may have been generated, located in different places, from the given specification. Genetic algorithms can adapt themselves to find this optimum, as long as the fitness function is correctly defined.

As future work, we would like to compare this approach with others existing in this field, in order to compare the quality of the sets of invariants that we obtain. In this line, we will consider the following techniques. The first one is Ant colony optimization, which uses many ants to traverse the solution space and find locally productive areas. Another approach is adding an entropy with respect to an invariant in order to use a cross-entropy method to generate candidate solutions via a parameterized probability distribution. The last approach to investigate is the cultural algorithm, which consists of the population component almost identical to that of the genetic algorithm and, in addition, a knowledge component called the belief space.

#### REFERENCES

- [1] G. Myers, *The Art of Software Testing*. John Wiley and Sons, 2nd ed., 2004.
- [2] P. Ammann and J. Offutt, *Introduction to Software Testing*. Cambridge University Press, 2008.
- [3] R. Hierons, J. Bowen, and M. Harman, eds., *Formal Methods and Testing, LNCS 4949*. Springer, 2008.
- [4] J. Jacky, M. Veanes, C. Campbell, and W. Schulte, *Model-Based Software Testing and Analysis with C#*. Cambridge University Press, 2008.
- [5] I. Rodríguez, M. Merayo, and M. Núñez, "HOTL: Hypotheses and observations testing logic," *Journal of Logic and Algebraic Programming*, vol. 74, no. 2, pp. 57–93, 2008.
- [6] R. Hierons, K. Bogdanov, J. Bowen, R. Cleaveland, J. Derrick, J. Dick, M. Gheorghe, M. Harman, K. Kapoor, P. Krause, G. Luetzgen, A. Simons, S. Vilkomir, M. Woodward, and H. Zedan, "Using formal methods to support testing," *ACM Computing Surveys*, vol. 41, no. 2, 2009.
- [7] I. Rodríguez, "A general testability theory," in *20th Int. Conf. on Concurrency Theory, CONCUR'09, LNCS 5710*, pp. 572–586, Springer, 2009.
- [8] M. Utting and B. Legeard, *Practical Model-Based Testing: A Tools Approach*. Morgan-Kaufmann, 2007.
- [9] A. En-Nouaary, R. Dssouli, and F. Khendek, "Timed Wp-method: Testing real time systems," *IEEE Transactions on Software Engineering*, vol. 28, no. 11, pp. 1024–1039, 2002.
- [10] M. Merayo, M. Núñez, and I. Rodríguez, "Formal testing from timed finite state machines," *Computer Networks*, vol. 52, no. 2, pp. 432–460, 2008.
- [11] M. Merayo, M. Núñez, and I. Rodríguez, "Extending EFSMs to specify and test timed systems with action durations and timeouts," *IEEE Transactions on Computers*, vol. 57, no. 6, pp. 835–848, 2008.
- [12] Y. Wang, M. Uyar, S. Bath, and M. Fecko, "Fault masking by multiple timing faults in timed EFSM models," *Computer Networks*, vol. 53, no. 5, pp. 596–612, 2009.
- [13] C. Andrés, M. Merayo, and M. Núñez, "Passive testing of timed systems," in *6th Int. Symposium on Automated Technology for Verification and Analysis, ATVA'08, LNCS 5311*, pp. 418–427, Springer, 2008.
- [14] C. Andrés, M. Merayo, and M. Núñez, "Formal correctness of a passive testing approach for timed systems," in *5th Workshop on Advances in Model Based Testing, A-MOST'09*, pp. 67–76, IEEE Computer Society Press, 2009.
- [15] C. Andrés, M. Merayo, and M. Núñez, "Using a mining frequency patterns model to automate passive testing of real-time systems," in *21st Int. Conf. on Software Engineering & Knowledge Engineering, SEKE'09*, pp. 426–431, Knowledge Systems Institute, 2009.
- [16] C. Andrés, M. Merayo, and M. Núñez, "Supporting the extraction of timed properties for passive testing by using probabilistic user models," in *9th Int. Conf. on Quality Software, QSIC'09*, pp. 145–154, IEEE Computer Society Press, 2009.
- [17] C. Andrés, L. Llana, and I. Rodríguez, "Formally transforming user-model testing problems into implementer-model testing problems and viceversa," *Journal of Logic and Algebraic Programming*, vol. 78, no. 6, pp. 425–453, 2009.
- [18] D. Fatiregun, M. Harman, and R. M. Hierons, "Evolving transformation sequences using genetic algorithms," in *4th IEEE Int. Workshop on Source Code Analysis and Manipulation, SCAM'04*, pp. 65–74, IEEE Computer Society Press, 2004.
- [19] Q. Guo, R. M. Hierons, M. Harman, and K. Derderian, "Computing unique input/output sequences using genetic algorithms," in *3rd Int. Workshop on Formal Approaches to Software Testing, FATES'03, LNCS 2931*, pp. 169–184, Springer, 2004.
- [20] D. Berndt and A. Watkins, "High volume software testing using genetic algorithms," in *38th Annual Hawaii Int. Conf. on System Sciences, HICSS'05*, p. 318b, IEEE Computer Society Press, 2005.
- [21] K. Derderian, M. Merayo, R. Hierons, and M. Núñez, "Aiding test case generation in temporally constrained state based systems using genetic algorithms," in *10th Int. Conf. on Artificial Neural Networks, IWANN'09, LNCS 5517*, pp. 327–334, Springer, 2009.
- [22] D. Goldberg, *Genetic Algorithms in Search, Optimisation and Machine Learning*. Addison-Wesley, 1989.
- [23] C. Andrés, M. Merayo, and C. Molinero, "Advantages of mutation in passive testing: An empirical study," in *4th Workshop on Mutation Analysis, Mutation'09*, pp. 230–239, IEEE Computer Society Press, 2009.

# Formal verification of UML 2.0 Sequence diagram

Sachoun Park<sup>1</sup>, Taeman Han<sup>1</sup>, and Gihwon Kwon<sup>2</sup>

<sup>1</sup>Electronic and Telecommunications Research Institute,  
138, Gajeongno, Yuseong-Gu, Daejeon, Korea  
{sachem, tmhan}@etri.re.kr

<sup>2</sup>Department of Computer Science, Kyonggi University,  
San 94-6, Yiui-Dong, Youngtong-Gu, Suwon-Si, Kyonggi-Do, Korea  
khkwon@kgu.ac.kr

## Abstract

*In UML 2.0, Sequence diagram is extended to have some reusable factors of interaction, e.g. combined fragment. But these additional features are hard to formally verify sequence diagrams. To overcome this obstacle, in this paper, we propose the formal semantics of sequence diagram with Finite State Process which is input language of LTS-BMC bounded model checker. The similarity between semantics of sequence diagram and FSP, interleaving and trace semantics, lead to formal verification of sequence diagram using a legacy well-developed bounded model checker.*

**Keywords** : Formal Verification, Bounded Model checking, Labeled Transition System, Sequence diagram, Finite State Process

## 1 Introduction

The formal verification of UML[1] diagrams plays an important role in detecting errors in early design phase. In UML, a common issue with sequence diagrams is how to show conditions and iterations. Indeed, the activity diagram is more appropriate to model control logic that involves conditions, loop etc, but in practice, most developers prefer to stick with the sequence diagram to show how objects interact together with the control logic involved. To increase reusability of interactions, Sequence diagram in UML 2.0 provide additional concepts such as combined fragment. But because of the lack of formal semantics of these features, it is hard to apply formal verification to sequence diagram. In this

---

This work was supported by the GRRRC program of Gyeonggi province. [200911963, Software Technology for Effective Digital Contents Service]

paper, to huddle this obstacle, we propose formal semantics of sequence diagram.

One of the most widely studied verification method is model checking first proposed in the early 1980's independently by two groups[2,3] The central idea is to model a system in such a way that it is possible to generate a reachability graph whose vertices represent the states the system can reach and the edges the possible transitions from a state to another. To perform model checking, it has to model a concrete system using some specification language. Several possibilities have been proposed depending on the domain. A simple but general formalism is called transition systems[4]. The transitions of the system can be associated with some action that is used as a transition label. Such a system model is referred to as a labeled transition system(LTS) which have well-defined mathematical properties.

In this paper, for defining formal verification of UML 2.0 sequence diagram, we use a previously developed bounded model checker LTS-BMC[5] whose input language is FSP[6]. We use this textual representation as our intermediate semantic model for verifying sequence diagram. The structure of this paper is that chapter 2 gives background knowledge about BMC(Bounded Model Checking), FSP and their encoding rules, and the formal syntax and semantics of sequence diagram are in chapter 3, and finally conclude in chapter 5.

## 2 Background

### 2.1 BMC

Bounded Model Checking was first proposed by Biere et al. in 1999[7]. It does not solve the complexity problem of model checking, since it still relies on an exponential procedure and hence is limited in its capacity. But experiments have shown that it can solve many cases that

cannot be solved by BDD-based techniques.

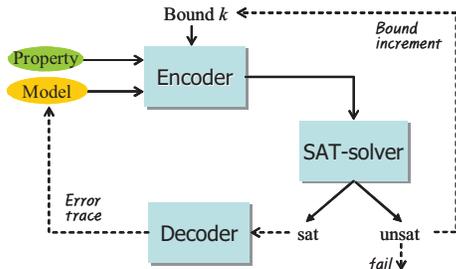


Figure 1. Bounded Model Checking Overview

As shown in figure 1, the basic idea in BMC is to search for a counter-example in executions whose length is bounded by some integer  $k$ . If no bug is found then one increases  $k$  until either a bug is found, the problem becomes fail to verification or some pre-known upper bound is reached. The BMC problem can be efficiently reduced to a propositional satisfiability problem, and can therefore be solved by SAT-solver rather than BDDs. Modern SAT solvers can handle propositional satisfiability problems[8,9,10].

During the last few years there has been a tremendous increase in reasoning power of SAT solvers. They can now handle instances with hundreds of thousands of variables and millions of clauses[11].

## 2.2 FSP

FSP(Finite State Processes) is a process calculus – one of a family of notations pioneered by Milner(1989), CCS(Calculus of Communicating System), and Hoare(1985), CSP(Communicating Sequential Processes), for concisely describing and reasoning about concurrent programs[6]. In figure 2, we give the quick reference of FSP model to help understand.

The semantics of basic *FSP* are defined in terms of Labeled Transition Systems(LTSs). Let *States* be the universal set of states including  $\pi$  a designed *error* state,  $L$  be the universal set of labels, and  $Act = L \cup \{\tau\}$ , where  $\tau$  is used to denote an internal action that cannot be observed by the environment of an *LTS*. A finite *LTS*  $P$  is a quadruple  $\langle S, A, \Delta, q \rangle$  where:

- $S \subseteq States$  is a finite set of states
- $A = \alpha P \cup \{\tau\}$ , where  $\alpha P \subseteq L$  denotes the *alphabet* of  $P$
- $\Delta \subseteq S - \{\pi\} \times A \times S$ , denotes a transition relation that maps from a state and action onto another state.
- $q \in S$  indicates the initial state of  $P$ .

Symbol	Name	Description
$\rightarrow$	Action Prefix	If $x$ is a action and $P$ a process then $(x \rightarrow P)$ describes a process that initially engages in the action $x$ and then behaves exactly as described by $P$
$ $	Choice	If $x$ and $y$ are actions then $(x \rightarrow P   y \rightarrow Q)$ describes a process which initially engages in either of the actions $x$ or $y$ .
<b>when</b>	Guarded Action	The choice ( <b>When</b> $B$ $x \rightarrow P   y \rightarrow Q$ ) means that when the guard $B$ is true then the actions $x$ and $y$ are both eligible to be chosen, otherwise if $B$ is false then the action $x$ cannot be chosen.
$+$	Alphabet Extension	The alphabet of a process is the set of actions in which it can engage. $P+S$ extends the alphabet of the process $P$ with the actions in the set $S$ .
$\parallel$	Parallel Composition	If $P$ and $Q$ are processes then $(P \parallel Q)$ represents the concurrent execution of $P$ and $Q$ .
<b>forall</b>	Replicator	<b>forall</b> $[i:1..N]$ $P(i)$ is the parallel composition $(P(1) \parallel \dots \parallel P(N))$ .
$:$	Process Labeling	$a:P$ prefixes each label in the alphabet of $P$ with $a$ .
$::$	Process Sharing	$\{a_1, \dots, a_n\} :: P$ replaces every label $n$ in the alphabet of $P$ with the labels $a_1.n, \dots, a_n.n$ . Further, every transition $(n \rightarrow Q)$ in the definition of $P$ is replaced with the transitions $\{a_1.n, \dots, a_n.n\} \rightarrow Q$ .
$\ll$	Priority High	$\ C=(P \parallel Q) \ll \{a_1, \dots, a_n\}$ specifies a composition in which the actions $a_1, \dots, a_n$ have higher priority than any other action in the alphabet of $P \parallel Q$ including the silent action $\tau$ .
$\gg$	Priority Low	$\ C=(P \parallel Q) \gg \{a_1, \dots, a_n\}$ specifies a composition in which the actions $a_1, \dots, a_n$ have lower priority than any other action in the alphabet of $P \parallel Q$ including the silent action $\tau$ .
<b>if then else</b>	Conditional	The process <b>if</b> $B$ <b>then</b> $P$ <b>else</b> $Q$ behaves as the process $P$ if the condition $B$ is true; otherwise it behaves as $Q$ . If the <b>else</b> $Q$ is omitted and $B$ is false, then the process behaves as <b>STOP</b> .
$/$	Re-labeling	Re-labeling is applied to a process to change the names of action labels. $\{newlabel\_1/oldlabel\_1, \dots, newlabel\_n/oldlabel\_n\}$
$\backslash$	Hiding	When applied to a process $P$ , the hiding operator $\{a_1, \dots, a_n\}$ removes the action names $a_1, \dots, a_n$ from the alphabet of $P$ and makes these concealed actions "silent". Silent actions in different processes are not shared. These silent actions are labeled $\tau$ .
$@$	Interface	When applied to a process $P$ , the interface operator $@\{a_1, \dots, a_n\}$ hides all actions in the alphabet of $P$ not labeled in the set $a_1, \dots, a_n$ .

Figure 2. FSP quick reference

The only *LTS* that is allowed to have the error state  $\pi$  as its initial state is  $\langle \{\pi\}, Act, \{\}, \pi \rangle$ , named  $\Pi$ . An *LTS*  $P = \langle S, A, \Delta, q \rangle$  transits with action  $a \in A$  into an *LTS*  $P'$ , denoted as  $P \xrightarrow{a} P'$ , if:

- $P' = \langle S, A, \Delta, q' \rangle$ , where  $q' \neq \pi$  and  $(q, a, q') \in \Delta$ , or
- $P' = \Pi$ , and  $(q, a, \pi) \in \Delta$ .

We use  $P \xrightarrow{a}$  to mean that  $\exists P'$  such that  $P \xrightarrow{a} P'$ , and we define a set of designed end states  $ES \subseteq States$  such that an *LTS*  $P = \langle S, A, \Delta, q \rangle$  is terminating is there is a state  $e \in S$  and  $e \in ES$  and  $\exists(e, a, q) \in \Delta$  for all  $a \in A$ .

In the following,  $E$  range over  $FSP$  process expressions,  $Q$  ranges over process identifiers, and  $A, B$  range over sets of observable actions. And  $lts:Exp \rightarrow \wp$ , where  $Exp$  is the set of  $FSP$  process expressions, and  $\wp$  the set of  $LTS$ s. The function  $lts$  is defined inductively on the structure of  $FSP$  process expressions:

- $Q = E$  means that  $lts(Q) = \text{def } lts(E)$
- $lts(\text{END}) = \langle \{e\}, \{\tau\}, \{\}, e \rangle$  where  $e \in ES$
- $lts(\text{STOP}) = \langle \{s\}, \{\tau\}, \{\}, s \rangle$ .
- $lts(\text{ERROR}) = \Pi$ .
- If  $lts(E) = \langle S, A, \Delta, q \rangle$  and  $E$  is not  $\text{ERROR}$  then  $lts(a \rightarrow E) = \langle S \cup \{p\}, A \cup \{a\}, \Delta \cup \{(p, a, q)\}, p \rangle$  where  $p \notin S$ .  $lts(a \rightarrow E) = \langle \{p, \pi\}, \{a\}, \{(p, a, \pi)\}, p \rangle$  where  $p \neq \pi$
- Let  $1 \leq i \leq n$ , and  $lts(E_i) = \langle S_i, A_i, \Delta_i, q_i \rangle$  then  $lts(a_1 \rightarrow E_1 \dots a_n \rightarrow E_n) = \langle S \cup \{p\}, A \cup \{a_1 \dots a_n\}, \Delta \cup \{(p, a_1, q_1) \dots (p, a_n, q_n)\}, p \rangle$  where  $p \notin S_i, S = \bigcup_i S_i, A = \bigcup_i A_i, \Delta = \bigcup_i \Delta_i$ . If  $E_i$  is  $\text{ERROR}$  then  $A_i = \{\}$ .
- If  $lts(E) = \langle S, A, \Delta, q \rangle$  then  $lts(E+B) = \langle S, A \cup B, \Delta, q \rangle$ .
- The recursive expression  $lts(\text{rec}(X=E))$ , where  $X$  is a variable in  $E$ , is the smallest  $LTS$  that satisfies the rule:  $\text{rec}(X=(a \rightarrow X)) \sim lts(E[X \leftarrow \text{rec}(x=E)])$ , where “ $\sim$ ” is strong semantic equivalence.
- If  $lts(P) = \langle S_p, A_p, \Delta_p, q_p \rangle$  is terminating with end state  $e_p \in ES$  and  $lts(E) = \langle S_e, A_e, \Delta_e, q_e \rangle$  then,  $lts(P; E) = \langle S_p \cup S_e, A_p \cup A_e, \Delta_p \cup \Delta_e, q_p \rangle$ , where  $q_e = e_p$ .
- If  $P = \Pi$  or  $Q = \Pi$ , then  $P \parallel Q = \Pi$ . For  $P = \langle S_1, A_1, \Delta_1, q_1 \rangle$  and  $Q = \langle S_2, A_2, \Delta_2, q_2 \rangle$ , such that  $P \neq \Pi$  and  $Q \neq \Pi$ ,  $P \parallel Q = \langle S_1 \cup S_2, A_1 \cup A_2, \Delta, (q_1, q_2) \rangle$ , where  $\Delta$  is the smallest relation satisfying the rules:

$$\frac{P \xrightarrow{a} P'}{P \parallel Q \xrightarrow{a} P' \parallel Q} a \notin \alpha Q \quad \frac{Q \xrightarrow{a} Q'}{P \parallel Q \xrightarrow{a} P \parallel Q'} a \notin \alpha P$$

$$\frac{P \xrightarrow{a} P', Q \xrightarrow{a} Q'}{P \parallel Q \xrightarrow{a} P' \parallel Q'} a \neq \tau$$

### 2.3 LTS Encoding

To encode  $LTS$ s into  $CNF$ (clause Normal Form), we reconstruct the flatten structure of the  $LTS$ s with semantics from original  $FSP$ . If we finish making a naïve flatten  $LTS$ s and their composition then we translate into

$CNF$ , format of  $SAT$ -solver. Figure 3 shows simple example about flattening  $LTS$ .

```

const N = 3
range T = 0..N
COUNT = COUNT[0],
COUNT[u:T] = (when(i<N) inc -> COUNT[i+1]
               |when(i>0) dec -> COUNT[i-1]).

COUNT = COUNT0,
COUNT0 = (inc -> COUNT1),
COUNT1 = (inc -> COUNT2 | dec -> COUNT0),
COUNT2 = (inc -> COUNT3 | dec -> COUNT1),
COUNT3 = (dec -> COUNT2).

```

Figure 3. Example of  $LTS$

$LTS P = \langle S, A, \Delta, q \rangle$  consists of an initial state, a set of transition, a set of alphabet and transition relation. Propositional formula of a given model  $\llbracket P \rrbracket_k$  is encoding as follow:

$$\llbracket P \rrbracket_k := \llbracket q \rrbracket_0 \wedge \bigwedge_{i=0}^{k-1} \llbracket \Delta \rrbracket_k$$

In the first bound, initial state  $q$  is encoded by  $\llbracket q \rrbracket_0$  and in each bound  $k$ , transition relation  $\bigwedge_{i=0}^{k-1} \llbracket \Delta \rrbracket_k$  is encoded by below four constraints:

C1. (Exactly One State) Exactly one state in the model is selected at the same time.

$$\bigwedge_{b=0}^{k-1} \bigvee_{i=0}^n s_{i,b}$$

$$\bigwedge_{b=0}^k \left( \bigwedge_{i=1}^{n-1} \bigwedge_{j=i+1}^n (\neg s_{i,b} \vee \neg s_{j,b}) \right)$$

where  $|S| = n$ , and  $s_{i,b}$  is  $i$ th state of  $S$  at the bound  $b$ .

C2. (AT Most One Action) At most one action is selected at the same time.

$$\bigwedge_{b=0}^k \left( \bigwedge_{i=1}^{n-1} \bigwedge_{j=i+1}^n (\neg a_{i,b} \vee \neg a_{j,b}) \right)$$

where  $|A| = n$ , and  $a_{i,b}$  is  $i$ th action of  $A$  at the bound  $b$ .

C3. (Enable Actions) Actions, that are possible to occur in a state, are encoded with corresponding state.

When  $|S| = n$ , for every state  $s_{i,b} \in S$  with bound  $b$ , we can define the set of enable actions in the state  $s_i$  as follow:

$$\text{enableActions}(s_i) = \{a \in A \mid \exists t. (s_i \xrightarrow{a} t) \in \Delta\}$$

Thus C3 is encoded by:

$$\bigwedge_{b=0}^k \bigwedge_{i=0}^n \left( \neg s_{i,b} \vee \bigvee_{j=1}^m a_{j,b} \right)$$

where  $m$  is  $|\text{enableActions}(s_i)|$  and  $a_{j,b}$  is one of action in the set of enable actions with the bound  $b$ .

C4. (Triggered) If the current state meets one of enable actions, then the transition is triggered and control is moved to the next state.

When  $|\Delta| = n$  and every transition  $\delta = (s, a, s') \in \Delta$ , in each bound  $k$ , C4 is encoded as follow:

$$\bigwedge_{b=0}^k \bigwedge_{i=0}^n (\neg s_{i,b} \vee \neg a_{i,b} \vee s'_{i,b})$$

LTSA (Labelled Transition System Analyser), developed by Jeff Magee and Jeff Kramer [12], supports a process algebra notation (FSP) for concise description of component behavior. LTSA is a verification tool for concurrent systems. It mechanically checks that the specification of a concurrent system satisfies the properties required of its behavior. In addition, LTSA supports specification animation to facilitate interactive exploration of system behavior. Figure 4 shows previous well-developed Bounded Model checker, LTS-BMC.

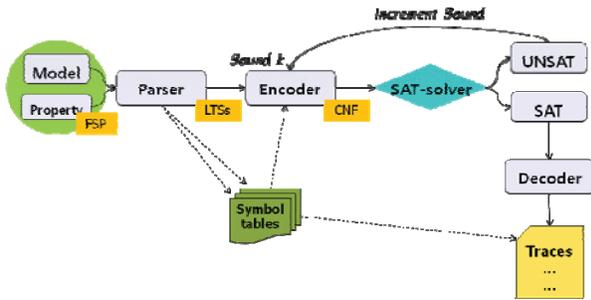


Figure 4. Overview LTS-BMC

### 3 Interpretation of SD over FSP

A sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. Interactions are used to get a better grip of an interaction situation for an individual designer or for a group that needs to achieve a common understanding of the situation. The most visible aspects of an Interaction are the messages between the lifelines. The sequence of the messages is considered important for the understanding of the situation. The data that the messages convey and the lifelines store may also be very important, but the Interactions do not focus on the manipulation of data even though data can be used to decorate the diagrams. A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

In UML A common issue with sequence diagrams is how to show conditions and iterations. Indeed, the

activity diagram is more appropriate to model control logic that involves conditions, loop etc, but in practice, most developers prefer to stick with the sequence diagram to show how objects interact together with the control logic involved. To do this, Sequence diagram in UML 2.0 provide the concept of combined fragment.

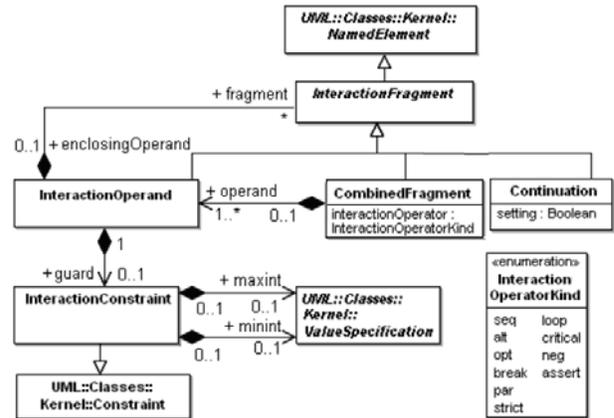


Figure 5. CombinedFragment

As shown in figure 5, a combined fragment defines an expression of interaction fragments. A combined fragment is defined by an interaction operator and corresponding interaction operands. Through the use of Combined Fragments the user will be able to describe a number of traces in a compact and concise manner. Combined Fragment is a specialization of InteractionFragment.

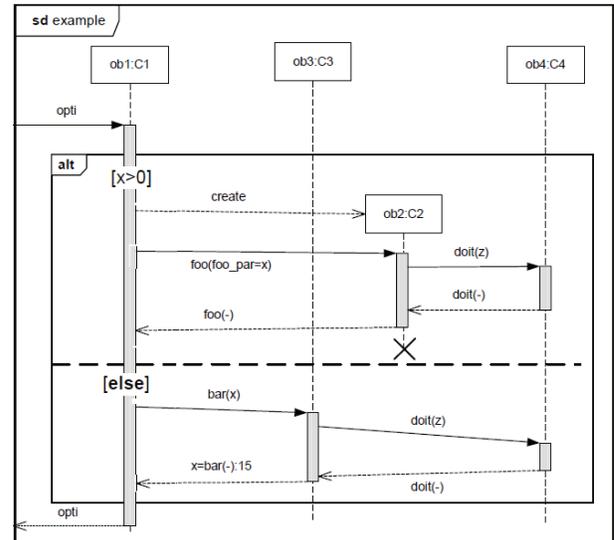


Figure 6. Example of Sequence diagram

Table 1 describes some selected interaction operators. In this paper, we only deal with limited operators such as

*alt*, *break*, *loop*, *opt*, and *par*. Figure 6 shows an example of a sequence diagram with *alt* operator.

<Table 1> Brief overview of interaction operators

Operator	Keyword	Description
alt	[guard1] [guard2] [else]	At most one of the operands will be chosen. The chosen operand must have an explicit or implicit guard expression that evaluates to true at this point in the interaction. An implicit true guard is implied if the operand has no guard.
assert		Assertion
break		A breaking scenario
loop	Minit, Maxint, [guard]	The Guard may include a lower and an upper number of iterations of the loop as well as a Boolean expression.
neg		Traces that are defined to be invalid.
opt	[guard]	A choice of behavior where either the (sole) operand happens or nothing happens.
par		A parallel merge between the behaviors of the operands.

### 3.1 Formal Syntax of Sequence diagram

A message which is sent or received between life lines is major element of sequence diagram, and the message occurrence formally defined as follow.

**Definition 1. (Message occurrence)**  $O = \{o_1, \dots, o_n\}$  is the finite set of message occurrence, where each  $o_i \in O$  is consist of a triple  $(s_i, m_i, r_i)$ , and  $s_i, r_i \in L \cup \{g\}$ .  $L$  is the finite set of life line and  $g \in Gate$  denotes gate.  $Gate$  is the set of all gates used in given sequence diagram.  $sender, receiver : O \rightarrow L \cup Gate$  are functions whose input is message occurrence and output is a life line or a gate, e.g.  $sender(o_i) = s_i$  and  $receiver(o_i) = r_i$ . A message script  $m_i \in Msg \subseteq SIG \cup CALL \cup \{create, destruct\}$  is the set of messages, and each element is one of elements of the set of signal  $SIG$  and the set of operation calls  $CALL$ , *create* signal, or *destruct* signal.  $message : O \rightarrow Msg$  is the function that returns a message corresponding to the input message occurrence, i.e.  $message(o_i) = m_i$ .

**Definition 2. (Combined fragment)**  $F = \{f_1, \dots, f_n\}$  denotes the finite set of combined fragments, which are consist of operator, operand, and guards, where  $f_i \in \wp(Operand) \times Operator$ , and  $Operand \subseteq Sd$  is the subset of sequence diagram and  $\wp(Operand)$  denotes the power set of operands.  $guard : Operand \rightarrow Guard \cup \{\varepsilon\}$  is the function that returns the set of guards which belong to the input operand.  $Guard$  denotes the set of expressions

which can be used in UML.  $Operator = \{opt, alt, ref, par, seq, loop\}$  is the set of interaction operators.

**Definition 3. (Sequence diagram)** Sequence diagram  $Sd = \{sd_1, \dots, sd_n\}$  denotes the finite set of interaction, where  $sd_i = (F, O_i, \triangleright_i)$ .  $O_i$  is the set of message occurrence, which is included in  $sd_i$ .  $\triangleright_i \subseteq (F \cup O_i) \times (F \cup O_i)$  is total order relation on message occurrences and combined fragments.

The interaction relation  $\angle$  is irreflexive transitive and satisfies following conditions, where  $sd_0$  is the unique root interaction which cannot be an operand.

$$\begin{aligned} & \forall i \cdot \neg(sd_i \angle sd_i) \\ & \forall i, j, k \cdot (sd_i \angle sd_j) \wedge (sd_j \angle sd_k) \Rightarrow (sd_i \angle sd_k) \\ & \forall i > 0 \cdot sd_i \angle sd_0 \end{aligned}$$

For example, according to the above formal syntax, the sequence diagram of figure 6 is represented as follow:  $Sd = \{sd_0, sd_1, sd_2\}$ ,  $sd_1 \angle sd_0$ ,  $sd_2 \angle sd_0$ ,  $F = \{(\{sd_1, sd_2\}, alt)\}$ ,  $guard(sd_1) = x > 0$ ,  $guard(sd_2) = x \leq 0$ . When  $sd_0 = (F, O_0, \triangleright_0)$ ,  $O_0 = \{o_1, o_2\}$ ,  $o_1 = (gate(sd_0), opt_i, ob1:C1)$ ,  $o_2 = (ob1:C1, opt_i, gate(sd_0))$  and  $o_1 \triangleright_0 (\{sd_1, sd_2\}, alt) \triangleright_0 o_2$ . When  $sd_1 = (F, O_1, \triangleright_1)$ ,  $O_1 = \{o_3, o_4, o_5, o_6, o_7\}$ ,  $o_3 = (ob1:C1, create, ob2:C2)$ , ...,  $o_7 = (ob2:C2, foo(), ob1:C1)$  and  $o_3 \triangleright_1 \dots \triangleright_1 o_7$ .

### 3.2 Formal Semantics of Sequence diagram

In UML 2.0 specification document, the term trace means to “sequence of event occurrences,” which corresponds well with common use in the area of trace-semantics, which is a preferred way to describe the semantics of Interactions. *FSP* semantics is also based on this Interleaving Semantics. Therefore, with translating from formal syntax defined by previous definitions into *FSP* grammar carefully, we can get the method for formal verification of sequence diagram. These translations are explained as following three translation rules.

**Rule 1. (Message occurrence)** A message occurrence  $o_i = (s_i, m_i, r_i)$  can be translated to action of *FSP*  $a \in \alpha P$ .  $\alpha P$  is the set of actions in *FSP*. Given a sequence diagram  $sd_i = (F, O_i, \triangleright_i)$ , the set of actions in *FSP*,  $\alpha P_i = O_i$ .

Considering the translation from sequence diagram into *FSP* over *LTS*  $P = \langle S, A, \Delta, q \rangle$ , the set of action  $A = \alpha P \cup \{\tau\}$ , the set of states  $S$  is  $L \cup Gate$ , i.e. life lines and gates, and the initial state  $q$  is correspond to the first occurrence of message in the sequence diagram. When

$sd_0 \in Sd$  is the root interaction and  $sd_0 = (F, O_0, \triangleright_0)$ ,  $q = sender(o_0^1)$ , where  $\forall i \triangleright_1 \cdot o_0^1 \in O_0 \wedge o_0^i \in O_0 \Rightarrow o_0^1 \triangleright_0 o_0^i$ .

**Rule 2. (Combined fragment)** A combined fragment  $f_i \in \wp(Operand) \times Operator$  is translated into process composition of FSP. Each *Operand* is correspond to each process of FSP, and translation rules of Operator  $\{opt, alt, ref, par, seq, loop\}$  are proposed in table 2.

<Table 2> Rules of translation *Operator* into FSP

Operator	Keyword	FSP
alt	[guard1] $sd_i$ [guard2] $sd_j$ [else] $sd_k$	ALT = ( <b>when</b> [guard1] $a_i \rightarrow Process_{sd_i}$   <b>when</b> [guard2] $a_j \rightarrow Process_{sd_j}$   $a_k \rightarrow Process_{sd_k}$ ).
break		BREAK = ( a $\rightarrow$ STOP ).
loop	minint, maxint, [guard] $sd$	<b>const</b> min = minint <b>const</b> max = maxint MININT(I=0) = <b>if</b> (I < min) <b>then</b> a[I] $\rightarrow$ Process $_{sd}$ ; <b>END</b> ) <b>else</b> <b>END</b> . MAXINT(I=min) = <b>if</b> (I < max) <b>then</b> <b>when</b> [guard] a[I] $\rightarrow$ Process $_{sd}$ ; <b>END</b> ) <b>else</b> <b>END</b> . LOOP = MININT; MAXINT.
opt	[guard] $sd$	OPT=( <b>when</b> [guard] a $\rightarrow$ Process $_{sd}$ ).
par	$sd_1, \dots, sd_k$	PAR=(Process $_{sd_1}$   ...   Process $_{sd_k}$ ).

**Rule 3. (Interaction)** For interaction  $sd_i = (F, O_i, \triangleright_i)$ , the translation of  $F$  is compliance to **Rule 1**, and the translation of  $O_i$  follows **Rule 2**.  $\triangleright_i = \langle oc_1, \dots, oc_n \rangle$  is  $\forall 1 \leq i \leq n \cdot oc_i \in (O_i \cup F)$  and total order, therefore sequence diagram  $sd_i$  can be translated a process  $Process_{sd_i} = (oc_1 \rightarrow \dots \rightarrow oc_n \rightarrow \text{END})$ .

## 4 Conclusions

The formal verification of UML diagrams plays an important role in detecting errors in early design phase. In UML, a common issue with sequence diagrams is how to show conditions and iterations. Nearly all developers prefer to stick with the sequence diagram to show how objects interact together with the control logic involved. To increase reusability of interactions, Sequence diagram in UML 2.0 provide additional concepts such as combined fragment. But because of the lack of formal semantics of these features, it is hard to apply formal verification to sequence diagram. In this paper, to huddle this obstacle, we propose formal semantics of sequence diagram. The similarity between semantics of sequence

diagram and FSP, interleaving and trace semantics, lead to formal verification of sequence diagram using a legacy well-developed bounded model checker.

## References

- [1] Object Management Group, Unified Modeling Language (UML), version 2.1.2, OMG Documentation, <http://www.omg.org/technology/documents/formal/uml.htm>, 2009.
- [2] E. M. Clarke and E. Emerson, "Design and synthesis of synchronization skeletons using branching time temporal logics," In Logic of Programs: Workshop, volumn 131 of LNCS, pages 52-71. Springer-Verlag, 1981.
- [3] J. P. Quielle and J. Sifakis, "Specification and verification of concurents systems in CESAR", In Proceedings of the 5<sup>th</sup> International Symposium of Programming, pages 337-350, 1981.
- [4] A. Arnold, Finite Transition Systems, Prentice Hall, 1994.
- [5] S. Park and G. Kwon, "Using Boolean Cardinality Constraint for LTS Bounded Model Checking," In the Proceedings of The 20th International Conference on Software Engineering & Knowledge Engineering, pp. 537-542, 2008.
- [6] J. Magee and J. Kramer, Concurrency – State Models and Java Programs, Chichester, John Wiley & Sons, 1999.
- [7] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu, "Symbolic model checking without BDDs", In Proceeding of Workshop on Tools and Algorithms for the Construction and Analysis of Systems, LNCS, Springer-Verlag, 1999.
- [8] A. Biere, A. Cimatti, E. Clarke, Ofer Strichman, and Y. Zhu, "Bounded Model Checking", Vol. 58 of Advances in Computers, 2003. Academic Press (pre-print).
- [9] T. Jussila, "BMC via dynamic atomicity analysis," In Proceedings of the International Conference on Application of Concurrency to System Design, IEEE Computer Society, June 2004.
- [10] T. Jussila, K. Heljanko, and I. Niemela, "BMC via on-the-fly determinization," In Proceedings of the 1<sup>st</sup> International Workshop on Bounded Model Checking, 2003.
- [11] M.W. Moskewicz, C. Madigan, Y. Zhao, L. Zhang and S. Malik, "Chaff: Engineering an Efficient SAT Solver," In Proceedings of Design Automation Conference, 2001.
- [12] <http://www.doc.ic.ac.uk/ltsa/>

# A Hierarchical Timed Coloured Petri Nets for BPMN-based Process Analysis

Ching Huey Wang, Pei Shu Huang  
 Dept. of Computer Science, National Chiao Tung  
 University,  
 Hsinchu, Taiwan  
 e-mail: {chinghui,pshuang}@cs.nctu.edu.tw

Feng Jian Wang  
 Dept. of Computer Science, National Chiao Tung  
 University,  
 Hsinchu, Taiwan  
 e-mail: fjwang@cs.nctu.edu.tw

**Abstract**—Although many business process models have been proposed, most of them do not contain all the arguments of control, message and data flows, defined in BPMN. They do not concern time conditions as in BPMN. On the other hand, these features allow a process to be defined with richer semantics but increase the difficulty of correcting an error or inaccurate process at workflow design. To simplify the analysis, we defined Hierarchical Timed Coloured Petri Nets ( $H_C^T$  PNETs), which is extended from Coloured Petri Nets (CPNETs) and adopted the analysis techniques in CPNETs and Timed PNETs (TPNETs). Finally, a comparison among  $H_C^T$  PNETs, PNETs, CPNETs and TPNETs is given.

**Keywords**—business process; control flow; data flow; BPMN; CPNETs; Timed PNETs

## I. INTRODUCTION

A BPMN-based workflow [1] is described with four entities: 1) role: describing the performers of task instantiated, 2) control flow: defining what, when and how tasks a workflow performs, 3) data flow: specifying what information entities are produced/manipulated/ passed in corresponding activities and 4) message flow: representing the interaction between processes through messages. An analysis based on the correlations among these four entities can help check or maintain consistency between execution order and data transition [2][3][4][5][6][7], as well as prevents exceptions due to contradiction between data flow, control and message interaction.

Here, we provide an easier way to extract knowledge from the four entities of a workflow. Based on our previous work [8], a model,  $H_C^T$  PNETs, extended from CPNETs [9][10] and TPNETs for analysis is proposed. The techniques associated with PNET, CPNET or Timed PNET are comprehensively applied to analyze reachability and detect deadlock.

The remainder of this paper is organized as follows. TCPNETs is introduced in Section 2 and its extension,  $H_C^T$  PNETs, is proposed in Section 3. In Section 4, a comparison between our approach and PNET, CPNET, and Timed PNET is given.

## II. TIMED COLOURED PETRI NETS – TCPNETS

In order to extend CPNETs with time, our approach is to associate a *time stamp*, denoted as  $@_r$ ,  $r \in \mathbb{R}$ , with token,

and attach a pair  $(\alpha, \beta)$  of *restricted firing interval* and *time consumption*, with transition  $t$ .  $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n)$  is a sequence of time intervals,  $t$  can be fired in  $\alpha_i$ ,  $1 \leq i \leq n$ .  $\alpha_i$  can be represented as  $[start_i, end_i]$ , a pair of real number referred to the start and end of firing interval  $\alpha_i$ , i.e.,  $t$  can be fired between  $start_i$  and  $end_i$  only. Time intervals  $\alpha_{i-1}$  and  $\alpha_i$  can be continuous, but not overlapped. If  $start_i = 0$ ,  $t$  can be fired before  $end_i$ ; otherwise,  $t$  can be fired only after  $start_i$ . For an iterative sequence, such as every Monday, we can use a notation, e.g.,  $\forall(Mon)$  to simplify the expression. In addition,  $\beta$  denotes how long the transition is expected to execute.

When a token is associated with a time stamp, the token is *timed*. If the time stamp is  $@_r$ , the token is available to consume after  $r$ , i.e.,  $r$  is the earliest time the token can be used. When a transition  $t$  is associated with  $(\alpha, \beta)$ ,  $t$  is timed. A timed transition can be fired in the time interval(s) in  $\alpha$  by taking  $\beta$  time. An untimed transition can be fired when it is enabled. The firing mechanism of untimed transition is the same as that defined in CPNETs.

In a TCPNET, defined in Definition 1, a *global clock* is introduced. The tokens in TCPNET are timed. Let an activity, declared with  $(\alpha, \beta)$  where  $\alpha = ([start, end])$ , be presented with a transition  $t$  in a TCPNET and  $t$  be fired at  $\tau$ ,  $start \leq \tau \leq end$ . An execution of  $t$  takes  $\beta$  time units. The value of the time stamp(s) associated with the token(s), which will be removed from  $t$ 's input place(s) when  $t$  is fired, needs to be less than or equal to  $\tau$ . When  $t$  is fired,  $t$  creates a time stamp  $\tau + \beta$  for its output token(s).

### Definition 1 (Timed Coloured Petri Nets)

A TCPNET is a 4-tuple  $TNet = (CNet, \tilde{G}, r', r_0)$  where

1. Time conditions associated with  $CNet = (P, T, F, \Sigma, C, V, G, A, m_0)$ , a CPNET, is specified by time declaration function  $\tilde{G}$ . In order to make the paper self-contained, the definitions of CPNETs are

given in [10].

2.  $\tilde{G}: T \rightarrow \text{timeCond}$  is a function that maps each transition into one time condition  $\text{timeCond}$ , such that  $\forall t \in T, \tilde{G}(t) = \text{timeCond} = (\alpha, \beta)$  where  $\alpha$  is  $(\alpha_1, \alpha_2, \dots, \alpha_n)$ , a sequence of time intervals, and  $\alpha_i = [start_i, end_i]$ ,  $[start_i, end_i] \in R \times R: start_i \leq end_i, 1 \leq i \leq n$ .
3. The global clock associated with  $TNet$  is started from time  $r_0$ .
4. The initial marking  $M_0$  is generated by  $m_0$  at time  $r', r_0 \leq r'$ .

Here is a sample TCPNet *net* (shown in Figure 2.1), designed with four timed tokens and two timed transitions, for explaining how a TCPNet works. *net*'s global clock is started from 0 and initial marking  $M_0$  is generated at time 100. Current time is 100. The  $U$ -typed token of  $M_0$ , denoted as  $1'(p_0, (U, x))@100$ , is assigned with value  $x$  and located in place  $p_0$ . The remaining three  $W$ -typed tokens of  $M_0$ , denoted as  $1'(p_1, (W, 0))@100$ ,  $1'(p_1, (W, 1))@100$  and  $1'(p_3, (W, 1))@100$ , are respectively assigned with value 0, 1, 1 and located in place  $p_1$ ,  $p_1$ ,  $p_3$ .  $M_0$  can be denoted as  $1'(p_0, (U, x))@100 + 1'(p_1, (W, 0))@100 + 1'(p_1, (W, 1))@100 + 1'(p_3, (W, 1))@100$ . Marking Transition  $t_0$  and  $t_1$  are associated with time condition  $(([180, 200], [210, 220]), 20)$  and  $([0, 250], 30)$ , respectively.  $t_0$  can be fired between time 180 and 200, or 210 and 230, and  $t_1$  can be fired before 250. An execution of  $t_0/t_1$  takes 20/30 time units.

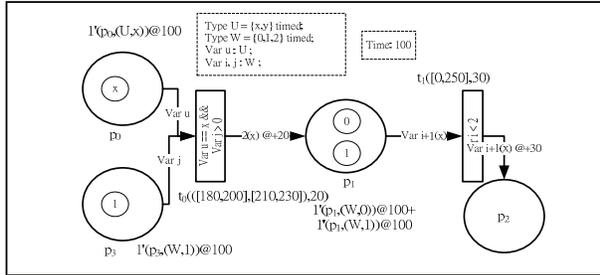


Figure 2.1 An example of TCPNet *net*.

**Definition 2 (Transition Binding with Time Conditions)**

A *binding*  $b_t$  of transition  $t$  in  $TNet$  is defined with

$B(t)$  of  $TNet$ 's  $cNet$  (defined in [10]). A binding  $b = \langle v_1 = val_1, v_2 = val_2, \dots, v_m = val_m \rangle$  in  $B(t)$  is also in  $B(t)$ , of  $TNet$ , if and only if the following condition is satisfied:

$\forall v \in \text{Var}(G(t)): b(v) = 1'(p, (c, val))$  and the time stamp  $@r$  of token  $(p, (c, val))$  is less than  $end_i$  of  $\alpha_n$ , an interval at the end of  $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n)$ .

**Definition 3 (Step Enabled)**

A step  $Y$  of  $TNet$ , a set of bindings, is enabled in a timed marking  $M_t$  at time  $\tau$  if and only if the following properties are satisfied:

$$(1) \quad \forall p \in P: \sum_{(t,b) \in Y} A(p,t) \langle b \rangle \subseteq m_t(p),$$

(2)  $\forall (t,b) \in Y$ ,  $\forall v \in \text{Var}(G(t)): b(v) = 1'((p, (c, val)), @r)$  and the time stamp  $@r$  of token  $(p, (c, val))$  is less than or equal to  $\tau$ , and

(3) there is an interval  $\alpha = [start, end]$  in  $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n)$  of  $t$ ,  $start \leq \tau \leq end$ .

**Definition 4 (Firing a Step)**

Let a step  $Y$  of  $TNet$  be enabled in a timed marking  $M_t$ .  $m_t'$  is a marking function that generates marking  $M_t'$  after firing  $Y$  at time  $\tau'$ .  $m_t'$  can be defined as:

$$\forall p \in P: m_t'(p) = \left( m_t(p) - \sum_{(t,b) \in Y} A(p,t) \langle b \rangle \right) + \sum_{(t,b) \in Y} A(t,p') \langle b \rangle @\tau' + \beta$$

Multi-set  $\sum_{(t,b) \in Y} A(p,t) \langle b \rangle$  represents the tokens removed from  $p$ , while  $\sum_{(t,b) \in Y} A(t,p') \langle b \rangle$  denotes the

tokens added to  $p'$ . After firing  $t$ , the time stamp of the tokens added is  $\tau' + \beta$  while an execution of  $t$  takes  $\beta$  time.  $M_t'$  is directly reachable from  $M_t$  by the occurrence of the step  $Y$ , denoted as  $M_t \xrightarrow{Y, \tau' > M_t'} M_t'$ .

Let two sequential steps  $Y_1$  and  $Y_2$  of *net* be  $\{(t_0, b_0)\}$  and  $\{(t_1, b_1), (t_2, b_2)\}$  where  $b_0 = \langle u = 'x', j = '1' \rangle$ ,  $b_1 = \langle i = '0' \rangle$  and  $b_2 = \langle j = '1' \rangle$ . The time conditions of transition  $t_0$  and  $t_1$  are  $(([180, 200], [210, 220]), 20)$  and  $([0, 250], 30)$ .

For the case of  $Y_1$ ,  $Y_1$  can be fired between time 180 and 200 or 210 and 220. If  $Y_1$  is fired at  $r_1$ ,  $r_1=180$ , one U-type token with value  $x$  and one I-type token with value 1 are removed from  $p_0$  and  $p_3$ , respectively, and two U-type tokens with value  $x$  are added into  $p_1$ . A time stamp  $@200$ ,  $@200 = @r_1 + 20$ , is created for the two added tokens. The timed marking of the result net, shown in Figure 4.2, is

$$1'(p_0, (1,0))@100 + 1'(p_1, (1,1))@100 + 2'(p_1, (U,x))@200$$

After firing  $Y_1$ ,  $Y_2$  is enabled. If  $Y_2$  is fired at  $r_2 = 220$ , the tokens of binding elements  $(t_i, b_i)$  and  $(t_i, b_i)$  are removed from  $p_1$  at the same time. A time stamp  $@250$ ,  $@250 = @r_2 + 30$ , is created for the four tokens generated by  $t_i$ .

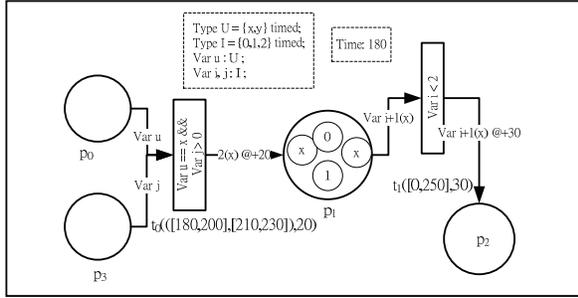


Figure 2.2 The result net of firing step  $Y_1$ .

### III. TCPNETS WITH HIERARCHY – $H_c^T$ CPNETS

Continuing the CPNet extension in Section 2, hierarchy is another property added to CPNets.  $H_c^T$  CPNet, a TCPNet with hierarchy and defined in Definition 5, supports levels of refining and abstracting transition and token. A transition  $t$  denoting a collapsed sub-process, whose expansion is another net, is called *compound transition*. A token  $tk$  denoting an artifact, whose state transition is represented with a PNet, is called *compound token*.

#### Definition 5 ( $H_c^T$ CPNets)

A  $H_c^T$  CPNets is a 5-tuple  $HNet = (TNet, CT, CTk, \tilde{CT}, \tilde{CTk})$ , where

1.  $T = T_a \cup T_c$ , a set of transitions in  $TNet$  is divided into two disjoint sets,  $T_a$  and  $T_c$ . The transitions in  $T_a$  are atomic and the transitions in  $T_c$  are compound.
2.  $CT$  is a set of  $H_c^T$  CPNets, each of which represents the expansion of a compound transition in  $T_c$ .

3.  $CTk$  is a set of PNets, each of which represents the state transition diagram of a data type in  $\Sigma$ .
4.  $\tilde{CT} : T_c \rightarrow CT$ , a 1-1 and onto function, is defined from  $T_c$  to  $CT$  by the conditions given in Definition 6.
5.  $\tilde{CTk} : \Sigma \rightarrow CTk$ , a 1-1 and onto function, is defined from  $\Sigma$  to  $CTk$  by the conditions given in Definition 7.

#### Definition 6 (Expansion of Compound Transition)

Given two  $H_c^T$  CPNets,  $HNet$  and  $HNet'$ ,  $HNet \neq HNet'$ ,  $HNet'$  is the expansion of a compound transition  $t$  in  $HNet$ , if and only if the following conditions are held.

Let  $CNet'$  of  $HNet'$  be  $(P', T', F', \Sigma', V', C', G', A', m'_0)$ .

1. The input and output places of  $t$  are transferred into  $P'$ ,  $(In(t) \cup Out(t)) \subseteq P'$ , i.e.,  $HNet'$  is started from the places in  $In(t)$  and terminated at the places in  $Out(t)$ ,
2.  $|T'| > 1$ , the number of transitions in  $T'$  is more than 1,
3.  $\bigcup_{p \in In(t) \cup Out(t)} c(p) \subseteq \Sigma'$ , the types (color sets) associated with the places in  $In(t) \cup Out(t)$  are included in  $\Sigma'$  and
4.  $\forall p \in (In(t) \cup Out(t))$ ,  $c'(p) = c(p)$ , i.e., the types associated with  $p$  in  $HNet$  are the same as that in  $HNet'$ .

#### Definition 7 (Expansion of Color Set)

Given a  $H_c^T$  PNet  $HNet$  and a PNet  $PNet = (P, T, F, m_0)$ , a color set (data type)  $c$  involved in  $HNet$  is designed with  $PNet$ , if and only if the following conditions are held:

1.  $|P| \geq 1$ , i.e.,  $PNet$  is composed of a place at least,
2.  $\forall t \in T$ ,  $|t^+| = |t^-| = 1$ , i.e.,  $t$  has exact one input and output places,
3.  $m_0 : P \rightarrow \{0,1\}$  and  $\sum_{p \in P} m_0(p) = 1$ , i.e., all reachable markings from  $M_0$  include one token only.

The number of colors (states) of color set (data type)  $c$  is equal to the number of places in  $PNet$ . These colors are presented with markings in  $[M_0, \succ$ , at set of reachable markings from  $M_0$ .

Let  $net$  be the TCPNet of  $H_c^T$  CPNet  $hnet$ , an example in Figure 3.1(a) which is applied to explain how a  $H_c^T$  CPNet

works. Transition  $t_1$  in  $hnet$  is compound and can be expanded to  $net'$  (shown in Figure 3.1 (c)). The input and output places  $p_1$  and  $p_2$  in  $hnet$  are transferred into  $net'$ . The cases of firing  $t_1$  before time 200 are transferred to  $t_{1-1}$  and the  $U$ -typed token is taken from  $p_1$  and put into  $p_2$  without changing value. The rest of cases of firing  $t_1$  are transferred to  $t_{1-2}$  and the value of the token is changed from  $x$  to  $y$ . In addition, the state transitions of data type  $U$  are represented with a PNet  $pnet$ , shown in Figure 3.1(b). The colors,  $x$  and  $y$ , of  $U$  are represented respectively with marking  $(1,0)$  and  $(0,1)$  in  $pnet$ , while the place array of  $pnet$  is  $(p_x, p_y)$ .

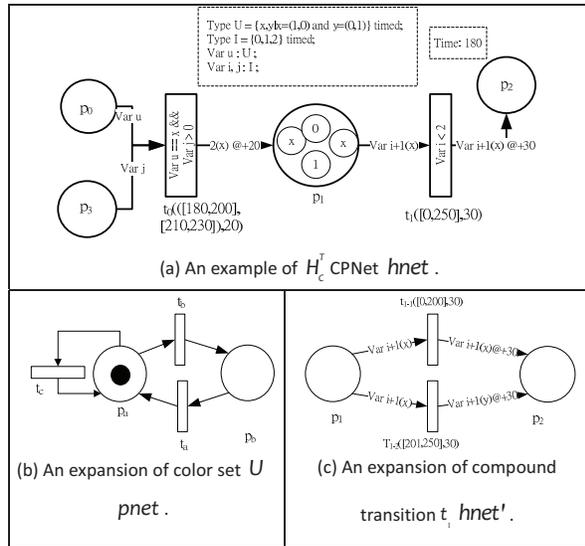


Figure 3.1 An example of  $H_c^T$  CPNet.

Here, transition  $t_0 / t_1$  in  $hnet$  correlates to transition  $t_c / t_a$  in  $pnet$ . Firing  $t_0$  or  $t_1$  triggers a state transition of  $U$ -typed token,  $pnet$ . When  $t_0$  is fired,  $t_c$  is fired concurrently. The state of the  $U$ -typed token, taken from input place  $p_0$ , is transformed from  $(1,0)$  to  $(1,0)$ . When  $t_1$  is fired,  $t_a$  is fired concurrently. The state of the  $U$ -typed token, taken from input place  $p_1$ , is transformed from  $(1,0)$  to  $(0,1)$ .

For simplicity, and without losing generality, we assume that each  $H_c^T$  CPNet has two levels in its hierarchy only. When a  $H_c^T$  CPNet is designed with more than two

levels, the compound transitions located in higher levels, 2 or more than 2, can be recursively replaced by its finer nets.

In addition, any  $H_c^T$  CPNet, restricted to start and end with places, is *weakly connected*, i.e., there is a path between any pair of distinct nodes in the net at least.

#### IV. CONCLUSION

This paper proposes an advanced PNets,  $H_c^T$  PNets, to represent a process modeled with BPMN, which includes roles, data flow, control flow, and message flow. In  $H_c^T$  PNets, hierarchical token is introduced to maintaining correlations between artifact state transition and activities. Hierarchical transition introduces a mechanism to show results of activity refinement. Time condition is allowed to be denoted and varieties of data semantic are easier to express with hierarchical token.

Table 4.1 Advantages of  $H_c^T$  PNets

$H_c^T$ PNets	PNets\ CPNets\ Timed PNets	
Hierarchical Token (Net within Net)	High difficulty of maintaining correlations between an artifact state transition and its operations	All
Hierarchical Transition	Un-introduce element refinement mechanism	All
Time Semantic	Time Condition Omission	PNets\ CPNets
Data Semantic	Weak Data Presentation	PNets\ Timed PNets

#### REFERENCES

- [1] Object Management Group. *Business Process Modeling Notation (BPMN) Version 1.2*. OMG Final Adopted Specification. Object Management Group, 2009.
- [2] R. Dijkman, M. Dumas, and C. Ouyang. "Semantics and Analysis of Business Process Models in BPMN.", *Information and Software Technology*, Vol.50, Issue 12, pp. 1281-1294, 2008.
- [3] Ivo Raedts, Marija Petkovic, Yaroslav S. Usenko, Jan Martijn E. M. van der Werf, Jan Friso Groote, Lou J. Somers. "Transformation of BPMN Models for Behaviour Analysis.", *MSVVEIS 2007*: 126-137.
- [4] S. Sadiq, M.E. Orłowska, W. Sadiq, and C. Foulger, "Data flow and validation in workflow modeling.", *Proceedings of the 15th Australasian database conference*, pp. 207-214, Dunedin, New Zealand, January 2004.
- [5] S.X. Sun, and J.L. Zhao, "A data flow approach to workflow design.", *Proceedings of the 14th Workshop on Information Technology and Systems*, pp. 80-85, 2004.
- [6] S.X. Sun, J.L. Zhao, and O.R. Sheng, "Data flow modeling and verification in business process management.", *Proceedings of the AIS Americas Conference on Information Systems*, pp. 4064-4073, New York, August 5-8, 2004.
- [7] S.X. Sun, J.L. Zhao, J.F. Nunamaker, and O.R.L. Sheng, "Formulating the data flow perspective for business process management.", *Information Systems Research*, Vol. 17, No. 4, pp. 374-391, December 2006.
- [8] C.H. Wang and F.J. Wang, "Detecting artifact anomalies in business process specifications with a formal model.", *Journal System and Software*, 2009.
- [9] W. Shen et al., "Hierarchical Timed Colored Petri Nets Based Product Development Process Modeling.", *CSCWD2004, LNCS 3168*, pp. 378 - 387, 2005.
- [10] K. Jensen, "Coloured Petri nets: Basic concepts, Analysis methods, and Practical use", volume 1-3, Springer-Verlag, 1992.

# Temporal Filter

## A Temporal Extension to Wireshark Display Filter \*

Shaochun Wang

State Key Laboratory of Computer Science  
Institute of Software, Chinese Academy of Sciences  
Beijing, China  
Graduate University of Chinese Academy of Sciences  
E-mail: scwang@ios.ac.cn

### Abstract

*Wireshark is a widely used network analysis tool. Its display filter is one of the most useful mechanisms in wireshark. In a display filter, we specify the packet property based on only individual packet. In this paper, we provide a temporal logic extension to the wireshark's display filter which gives us the power to specify property based on the whole network flow. This temporal filter has enough power to express properties we care about in reality.*

### 1. Introduction

The behavior of network exists in packets transferred in the network, and exists in every octet transferred in the network. When something unnormal happens, people need to use some tool to determine where the problem is. A very frequently used tool is wireshark [4]. We use wireshark to capture the packets transferred in the network and then analyse these packets manually. After making an initial assumption about where the problem is, we can use wireshark's display filter to filter out the packets we do not want to see. Using the display filter mechanism of wireshark, we can restrict our analysis to a small subset of the whole captured packets.

Normally, there are still many packets even after applying display filter. It is not enough to just consider individual packets. We need to analyse the packet's behavior. A packet's behavior is the functionality of a packet, which is the character that the packet takes in the whole network sequence. As a matter of fact, almost all normal packets in a network belong to some types of protocols. Protocol defines the sequence of network packets.

\*Supported by the National Natural Science Foundation of China under Grant Nos. 60721061, 60833001, and the CAS Innovation Program.

In order to describe the behavior of a packet, we need the following abilities:

1. The ability to describe an event sequence.
2. The ability to specify the packet relation

The first ability means that it has the power to describe the concept of "an event A occurs before an event B" or "an event A occurs after an event B" and so on. Temporal logic [3] provides us such ability. Temporal logic is widely used in model checking [1] as property specification language. In model checking, the model of temporal logic is infinite future string, but the network flow we capture is just a finite string. Orna Lichtenstein and Amir Pnueli [2] discussed the extension of *LTL* over finite strings. Temporal operators give us succinct, intuitive description of an event sequence.

The second ability we need is to specify the packet relation. To solve this problem, we extend the traditional temporal logic formula with one special packet variable *cur*, which always points to the packet we want to check currently. By referencing the variable *cur*, we can specify the mathematical relation between packets. We use such extended version of temporal logic to specify the property of the packet we want to analyse.

The remaining parts of this paper are outlined as following. Section 2 defines the formal model of a network packet flow and the syntax and semantics of this temporal logic. Section 3 exemplifies how to use this temporal logic to detect network attacks. Section 4 presents an algorithm to find all packets at which the formula to be checked holds. We conclude our current work at section 5.

### 2. Model and Specification

As discussed above, we can view network flow as a path. In order to specify properties of a path, we need to specify

properties concerning with individual packets firstly. Then we can combine them with temporal logic operator to express flow properties.

## 2.1. Variables

Formally, a *network flow* is a finite length string  $\pi = p_1, p_2, \dots, p_n$ , in which  $p_i, 1 \leq i \leq n$  is a packet. A packet can be viewed as a combination of properties we concern. For example, we may concern the source IP address, destination IP address of a packet. We denote such variables by a set  $V$ . The set  $V$  contains all the symbols we want to check against an individual packet. Using wireshark's display filter convention, we can represent  $V$  by

$$V = \{ip, ip.src, ip.dst, tcp.srcport, tcp.dstport, \dots\}.$$

Every variable in  $V$  can also be considered as a map, which maps a packet to the corresponding domain. For example,  $ip.src$  maps an IP packet to its source IP address. Formally, we have

$$v \in V, v : P \mapsto Values$$

$P$  is packet set, which contains all of the network packets;  $Values$  is the value domain, which contains all of the possible values, e.g. IP address 192.168.8.20, TCP flags 0x02 and so on.

We denote the value of a variable  $v$  at a packet  $p$  by  $p[v]$ , e.g. for a packet  $p$  whose IP source address is 10.84.35.1, the  $p[ip.src] = 10.84.35.1$ .

## 2.2. Expressions

Expression is constants, functions and variables combined as usual. Constant can be viewed as a special function whose arity is zero. Functions are  $+$ ,  $-$ ,  $==$  and so on. The meaning of functions are as usual. But in some situation, function meaning can be overloaded when appropriate. For example, in expression

$$ip.src == 192.168.8.0/24$$

the function  $==$  means "in", which is true if and only if the IP source address is in the subnet 192.168.8.0/24.

Formally, expression can be defined as following

- Variable  $v$  is an expression
- $f$  is n-arity function, and  $e_1, e_2, \dots, e_n$  is expressions with appropriate type, then  $f(e_1, e_2, \dots, e_n)$  is an expression. Here  $f$  is not logical operator.

At an individual packet, an expression will have a value. That is the semantics of expressions. We define it as following:

- Variable  $v$ 's value at packet  $p$  is  $p[v]$
- $p[f(e_1, e_2, \dots, e_n)] = f(p[e_1], p[e_2], \dots, p[e_n])$

The above syntax and semantics definition of expressions are almost the same as traditional definition of expression of a state. But it does not give us enough power to express properties we concern. In order to have the ability 2 in section 1, we need a mechanism to express property with respect to a particular packet. For example, when we look at a network flow  $\pi = p_1, p_2, \dots, p_n$ , we may want to check  $p_i$  one by one. When checking the packet  $p_i$ , we may want to check whether  $p_i$  satisfies a property such as "There is a packet in the past whose IP destination address equals to the source address of this current packet  $p_i$ ".

This requires us to extend the expression syntax to include a specific referencing packet: the packet we concerns currently. We extend the definition of expressions with one more rule

- $v$  is a variable,  $cur[v]$  is an expression

Correspondingly, the semantics of such extended expression with the packet reference variable  $cur$  points to packet  $q$ , is defined as

- $p[cur \mapsto q][cur[v]] = q[v]$

Here and after, without explicitly specified, expression is the extended expression.

- A *proposition* is an expression whose value is boolean.

## 2.3. Temporal Filter

A temporal filter is just a temporal logic formula whose proposition defined on the above. Traditionally, temporal logic formula is defined with finite past and infinite future. When we do network analysis, we always cope with finite network packet sequence. So here we define temporal filter not only with finite past, but also with finite future.

The syntax of temporal filter is

- proposition is temporal filter
- $p$  is temporal filter, then  $\neg p$
- $p, q$  are temporal filters, then  $p \wedge q$
- $p$  is temporal filter, then  $\ominus p$
- $p, q$  are temporal filters, then  $p S q$
- $p$  is temporal filter, then  $\bigcirc p$
- $p, q$  are temporal filters, then  $p U q$

We interpret temporal filter on a specified packet  $p_i$  of a network flow  $\pi$  with  $cur$  pointing to a packet  $p_j$ . The semantics of temporal filter is

- $\pi(i, p_j) \models p$  iff  $p_i[cur \mapsto p_j][p] = true$ , if  $p$  is a proposition
- $\pi(i, p_j) \models \neg\varphi$  iff  $\pi(i, p_j) \not\models \varphi$
- $\pi(i, p_j) \models \psi_1 \wedge \psi_2$  iff  $\pi(i, p_j) \models \psi_1$  and  $\pi(i, p_j) \models \psi_2$
- $\pi(i, p_j) \models \ominus\psi$  iff  $i > 1$  and  $\pi(i - 1, p_j) \models \psi$
- $\pi(i, p_j) \models \psi_1 S \psi_2$  iff there exists a  $k, 1 \leq k \leq i$  such that  $\pi(k, p_j) \models \psi_2$  and for every  $m, k < m \leq i, \pi(m, p_j) \models \psi_1$
- $\pi(i, p_j) \models \circ\psi$  iff  $i + 1 \leq n$  and  $\pi(i + 1, p_j) \models \psi$
- $\pi(i, p_j) \models \psi_1 U \psi_2$  iff there exists a  $k, i \leq k \leq n$  such that  $\pi(k, p_j) \models \psi_2$  and for every  $m, i \leq m < k, \pi(m, p_j) \models \psi_1$

Additional temporal operators can be defined by:

- $\bar{\diamond}\varphi = true S \varphi$
- $\bar{\square}\varphi = \neg\bar{\diamond}\neg\varphi$
- $\diamond\varphi = true U \varphi$
- $\square\varphi = \neg\diamond\neg\varphi$

### 3. Using Temporal Filter to Capture Evil Packets

In this section, we will exemplify how to use the temporal filter we describes above to capture the packets we concerns in a network flow. Here, the meaning of variables are the same as the one used in the display filter of wireshark.

#### 3.1. Unsolicited ARP Reply

Address Resolution Protocol (ARP) is the protocol for finding a host's link layer address. Normally, when a host wants to communicate with other host in the same subnet, it sends out an ARP request to the broadcast address. The host, whose IP address is the same as the destination address in the ARP request packet, will send back an ARP reply with its MAC address contained after receiving such ARP request. So usually, ARP reply is the result of ARP request. In other words, we can say "Every ARP reply is preceded by an ARP request". We define an unsolicited ARP reply as the ARP reply packet which is not preceded by an ARP request. Using temporal filter, we can formulate it as:

$$arp.code == 0x0002 \wedge \neg\bar{\diamond}arp.opcode == 0x0001 \quad (1)$$

In formula 1,  $arp.opcode == 0x0002$  means the packet is an ARP reply packet because ARP reply operation code

is 0x0002;  $arp.opcode == 0x0001$  means the packet is an ARP request packet because ARP request operation code is 0x0001. All of these propositions can be determined by only the individual packet.

Using formula 1 to capture unsolicited ARP reply, we will get almost nothing useful in a typical environment. This is because we failed to specify the ARP request which requests the current ARP reply. Instead, we specify all the ARP requests before. So what we really want is "Every ARP reply is preceded by an ARP request, which requests this ARP reply".

In the above declaration, the difficulty is to express "which requests this ARP reply". We need a method to describe this relation between an ARP request and the ARP reply. In ARP, an ARP request requests for an IP address to resolve; ARP reply replies with MAC address corresponding to this IP address. The IP address exists in both ARP request and reply packet, so we can express this relation as:

$$arp.dst.proto_ipv4 == cur[arp.dst.proto_ipv4]$$

Then the unsolicited ARP replay can be specified as

$$\begin{aligned} &arp.opcode == 0x0002 \wedge \\ &\neg\bar{\diamond}(arp.opcode == 0x0001 \\ &\wedge arp.dst.proto_ipv4 == cur[arp.dst.proto_ipv4]) \end{aligned} \quad (2)$$

All packets which satisfy formula 2 will be suspected as an unsolicited ARP reply packet. The hosts who are sending out such packets may be launching ARP spoofing [5] at that moment.

#### 3.2. Nmap ACK Scanning

Nmap (Network Map) is a network security analysis tool, which is a two-blades sword. Many attackers use nmap to do host discovery in the early stage of attacking. Nmap has a few facilities to do host discovery, e.g. PING scanning, SYN scanning, ACK scanning and so on. Among them, ACK scan is very effective and difficult to be detected.

Usually, an ACK packet is used to acknowledge the acceptance of a previous received packet in TCP. In ACK scanning, nmap sends out a bogus ACK packet to the remote host. If that host is on-line, it will send RST packet back to nmap; otherwise, nothing is sent back. By observing whether there is RST packet sent back, nmap knows whether the host is on-line or off-line. Using traditional network analysis tool, such as wireshark, such ACK host scanning is not easily detected because there are hundreds of ACK packets in typical scenario. You can not determine which one is the ACK scanning only by looking at the ACK packet itself.

In order to tell ACK scanning packets from normal ACK packets, we observe that "In normal situation, all ACK packets are preceded by packets which come from the destination IP and port of that ACK packet" and ACK scanning packet does not hold such property. Then we can use the following formula to pick ACK scanning packets

$$\begin{aligned}
tcp.flags == ACK \wedge \\
\neg \odot \diamond(ip.dst == cur[ip.src] \\
\wedge tcp.srcport == cur[tcp.dstport] \\
\wedge tcp.dstport == cur[tcp.srcport])
\end{aligned} \quad (3)$$

After finding the packets which satisfy formula 3, we can know the hosts who are doing host scanning.

## 4. Algorithm

In this section, we present an algorithm which can find all packets in a network flow which satisfy a temporal filter. Given a network flow  $\pi = p_1, p_2, \dots, p_n$ , we want to find all  $p_i$  such that  $\pi(i, p_i) \models \varphi$ .

The algorithm in Fig. 1 checks packets in  $\pi$  one by one. It calls function *Check* to determine whether formula  $\varphi$  holds at the packet  $p_i$  with *cur* pointing to  $p_i$ . If holds, then  $i$  is added to set  $A$ . At the end of computation, set  $A$  contains all the packets where formula  $\varphi$  is *true*. The function *Check* in Fig. 1 checks whether filter  $\varphi$  holds according to the structure of  $\varphi$ . The correctness of this algorithm depends on the following two equations

$$\psi_1 S \psi_2 = \psi_2 \vee (\psi_1 \wedge \odot(\psi_1 S \psi_2)) \quad (4)$$

$$\psi_1 U \psi_2 = \psi_2 \vee (\psi_1 \wedge \odot(\psi_1 U \psi_2)) \quad (5)$$

We omits the detail of the proof of the correctness of this algorithm.

## 5. Conclusion

The temporal filter presented in this paper provides a mathematically accurate mechanism to describe the sequential property of a network flow. It is also intuitive to use and flexible enough to describe real network attack pattern. It extends the Wireshark's network analysis power. Our temporal filter is an application of temporal logic to low level network analysis, not at the high level protocol way. It deals with the real packets captured in network. Our examples also show that it is suitable to do network security analysis.

## References

[1] E.M. Clarke, O. Grumberg, and D.A. Peled. *Model checking*. Springer, 1999.

```

procedure TLFILTER
  A ← ∅
  for i = 1 to n do
    if CHECK(i, pi, φ) = true then
      A = A ∪ {pi}
    end if
  end for
end procedure
▷ Now A contains all packets
▷ which satisfy filter φ

function CHECK(i, pj, φ)
  Case φ = p
    return pi[cur ↦ pj][p]
  Case φ = ¬ψ
    return ¬CHECK(i, pj, ψ)
  Case φ = ψ1 ∧ ψ2
    return CHECK(i, pj, ψ1) ∧ CHECK(i, pj, ψ2)
  Case φ = ⊙ψ
    if i > 1 then
      return CHECK(i - 1, pj, ψ)
    else
      return false
    end if
  Case φ = ψ1 S ψ2
    if CHECK(i, pj, ψ2) = true then
      return true
    else if CHECK(i, pj, ψ1) = true and i > 1 then
      return CHECK(i - 1, pj, φ)
    else
      return false
    end if
  Case φ = ⊙ψ
    if i + 1 ≤ n then
      return CHECK(i + 1, pj, ψ)
    else
      return false
    end if
  Case φ = ψ1 U ψ2
    if CHECK(i, pj, ψ2) = true then
      return true
    else if CHECK(i, pj, ψ1) = true and
      i + 1 ≤ n then
      return CHECK(i + 1, pj, ψ1 U ψ2)
    else
      return false
    end if
end function

```

**Figure 1. Finding all packets which satisfy the temporal filter**

[2] Orna Lichtenstein, Amir Pnueli, and Lenore D. Zuck. The glory of the past. In Rohit Parikh, editor, *Logic of Programs*, volume 193 of *Lecture Notes in Computer Science*, pages 196–218. Springer, 1985.

[3] Z. Manna and A. Pnueli. *The temporal logic of reactive and concurrent systems: Specification*. Springer, 1992.

[4] A. Orebaugh, G. Ramirez, and J. Burke. *Wireshark & Ethereal network protocol analyzer toolkit*. Syngress Publishing, 2007.

[5] S. Whalen. An introduction to arp spoofing. *Node99 [Online Document]*, April, 2001.

# System Modeling from Extended Task Descriptions

Jose Luis de la Vara and Juan Sánchez

*Centro de Investigación en Métodos de Producción de Software*  
*Universidad Politécnica de Valencia, Camino Vera s/n, 46022, Valencia, Spain*  
*{jdelavara, jsanchez}@pros.upv.es*

## Abstract

*Success of a software system highly depends on the requirements engineering process. When an information system is going to be developed for an organization, business processes should be modeled and used for requirements elicitation. Nonetheless, any requirements engineering approach must be linked to subsequent stages in order to be useful in a software development process. This paper addresses this need by linking a business process-based requirements engineering approach to system modeling. Once requirements have been specified from business process models by means of extended task descriptions, a set of rules determine how to derive an analysis class diagram and analysis state transition diagrams. As a result, extended task descriptions are integrated into object-oriented software modeling and development. The link is called ETD2OO, is the result of an industrial project and has been applied in field trials.*

## 1. Introduction

Success of a software system not only depends on the resolution of technical problems, but also highly depends on the requirements engineering (RE) process. Requirements can be regarded as the main indicators of the quality of a software system, and a system can fail if the RE process is not properly addressed [3].

When developing an information system (IS) for an organization, system analysts should model its business processes. Any IS should support the execution of business processes involving people, applications, and/or information sources on the basis of process models [8], and business process modeling can be regarded as essential for requirements elicitation [1].

Nonetheless, proper software requirements specification (SRS) from business processes does not imply that a RE approach is adequate for IS development. A RE approach is not useful per se, but it must be appropriate for the software development process into which it is integrated [16]. Therefore, any RE approach must be linked to subsequent development stages.

Furthermore, integration of a RE approach into a development process can be problematic. For example, problems may arise in IS development from object-oriented (OO) modeling. If requirements and OO diagrams are not properly managed, then inconsistency and incompleteness may appear between and in them [9][10]. Mechanisms and guidance for OO modeling from requirements are needed to avoid these problems.

As a solution, this paper presents system modeling from business process-based requirements. The link is the result of a project with a software development company. A business process-based RE approach [4][6][7] has been developed for the company, but it must also be integrated into OO-Method [15], a methodology for automatic software generation that the company uses and that is based on OO modeling.

In the RE approach, software requirements of an IS are specified by means of extended task descriptions (ETD). Once ETDs have been specified, an analysis class diagram (ACD) and analysis state transition diagrams (ASTD) are derived from them as part of system modeling with OO-Method. The link is called ETD2OO, is determined by a set of rules, and has been applied and evaluated in several field trials.

The paper is organized as follows. Section 2 describes a running example. Section 3 presents ETD specification. Sections 4 and 5 describe ETD2OO and its evaluation, respectively. Section 6 reviews related work. Finally, Section 7 presents our conclusions.

## 2. Running example

The software development company belongs to a holding company. Other organizations of this company have been used to apply ETD2OO in field trials. As a running example, a rent-a-car company is used. Nonetheless, its actual running is not completely explained.

The company is located in a tourist area, and its fleet of cars varies between the summer and the winter seasons. Cars are usually bought at the beginning of a season and sold at the end. The main activity of the company is car rental, but it involves other activities such car maintenance and extras rental.

### 3. ETD specification

The RE approach consists of three stages (organizational modeling, purpose analysis and requirements specification), and ETDs are used for SRS. They aim to specify adequate software support for business tasks, and are elicited from business process models. It should be noted that ETDs are different from use cases, which aim to specify interactions with a system.

Extended Task Description: CAR RENTAL			
Business process: Car rental		Role: Office employee	
Subtasks: Choose a car, Check whether a customer is new or not, Record customer data, Search for customer data, Fill contract, Choose extras, Take deposit, Print contract details			
Triggers: -			
Preconditions: -			
Postconditions: -			
Frequency: 10 times per day during winter season; 30 times per day during summer season			
Critical: Days on which a holiday period begins in summer season			
Input		Output	
Domain Entity	State	Domain Entity	State
Car	Ready	Rental contract	Open
Customer (1)	-	Car	Rented
Extra	Ready	Customer (2)	-
		Extra	Rented
<b>Business Rules</b>			
<ul style="list-style-type: none"> <li>The insurance of a car must be valid during the rental period</li> <li>A car cannot be rented if it has more than 300000 km</li> </ul>			
User Intention	System Responsibility	Information Flows	
<i>Normal interaction</i>			
1. Show cars	1. Show cars	$\leftarrow \{ \text{Car} / \text{make} + \text{model} / \}_n$ $\rightarrow \text{Car}$	
2. Select a car	3. Show customers	$\leftarrow \{ \text{Customer} / \text{name} + \text{surname} + \text{ID number} / \}_n$ $\rightarrow \text{Customer (1)}$	
4. Select a customer	5. Introduce rental contract information	$\rightarrow \text{Rental contract} / \text{contract number} + \text{current date} + \text{current time} + \text{office} + \text{return date} + \text{return office} /$ $\leftarrow \text{Rental contract} / \text{contract number} + \text{current date} + \text{current time} + \text{office} + \text{return date} + \text{return office} + \text{rental cost} + \text{extras cost} + \text{VAT} + \text{deposit} + \text{total cost} / + \text{Car} / \text{make} + \text{model} + \text{plate number} / + (\text{Customer (1)} / \text{name} + \text{surname} + \text{ID number} / + \text{Customer (2)} / \text{name} + \text{surname} + \text{ID number} / + \{ \text{Extra} / \text{name} / \}_n$	
5. Introduce rental contract information	6. Show rental contract details		
6. Show rental contract details	7. Print rental contract details		
7. Print rental contract details			
<i>Alternatives</i>			
(New customer)			
4.a.1. Introduce customer data [5]		$\rightarrow \text{Customer (2)} / \text{number} + \text{name} + \text{surname} + \text{ID number} + \text{address} + \text{city} + \text{telephone number} + \text{credit card type} + \text{credit card number} + \text{credit card expiration date} /$	
<i>Extensions</i>			
	(Extras request)		
5.a.2. Select extras	5.a.1. Show extras	$\leftarrow \{ \text{Extra} / \text{name} / \}_n$ $\rightarrow \{ \text{Extra} / \}_n$	
	(Deposit payment)		
5.b.1. Introduce deposit amount		$\rightarrow \text{Rental contract} / \text{deposit} /$	
<b>Quality attributes</b>			
<ul style="list-style-type: none"> <li>When an office employee selects a car, no other office employee will be able to select the same car (Functionality/Suitability)</li> </ul>			

Figure 1. ETD

A significance criterion for ETDs has been defined so that their granularity is homogeneous and, thus, their specification is consistent and proper [6]. As explained below, homogeneous and adequate granularity is also necessary for correct derivation of OO diagrams.



Figure 2. Domain data model

Figure 1 shows an example of ETD for the running example. The domain entities of input and output and of the information flows (IF) are part of a domain data model (Figure 2), which is created in the organizational modeling stage of the RE approach. More details about ETDs and how to specify them can be found in [6][7].

The ETD version that is shown in this paper is different from previous ones. Quality attributes (on the basis of [12]), frequency and critical sections are new, and IFs are specified for each user and system action.

Finally, ETDs are ordered. Figure 3 shows the sequence of ETDs that are related to car lifecycle.

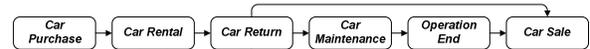


Figure 3. Sequence of ETDs

### 4. ETD2OO: derivation of OO diagrams

Once the ETDs of an IS have been specified, analysis OO diagrams are derived for system modeling from ETDs. These diagrams are an ACD and ASTDs.

#### 4.1. Rules for derivation of ACDs

An ACD of an IS is derived from the IFs and input and output of its ETDs, their sequence and a domain data model. There are ten rules for the derivation, and they allow system analysts to model classes and their attributes, methods and associations.

Figure 5 shows an ACD for the running example. It has been derived from the ETD shown in Figure 1, the domain data model shown in Figure 2, the sequence of ETDs shown in Figure 3, and the input, output and IFs shown in Figure 4. These figures correspond to just a part of the running example, and thus the ACD is incomplete. In addition, parameters and data types have not been modeled to keep Figure 5 as small as possible.

The rules are the following ones.

**Rule C1 (classes)** A class is modeled in an ACD for each domain entity of an ETD.

For the running example, the classes are “Insurance”, “Rate”, “Car”, “Customer”, “Rental Contract” and “Extra”.

**Rule C2 (attributes)** An attribute is modeled in a class for each attribute that belongs to the domain entity from which the class was modeled and that is in an input flow of an ETD. A data type is specified for each attribute.

Some attributes of “Car” are “plate number”, “make”, “model”, “engine”, “color”, and “seats”.

**Rule C3 (creation method)** A creation method is modeled for each class. Its parameters are the attributes

of the domain entity from which the class was modeled in the first ETD where the domain entity is in an input flow. A data type is specified for each parameter.

The creation method of “Car” is “create car (plate number, make, model, engine, color, seats, date)”.

**Rule C4 (deletion method)** A deletion method is modeled in a class if: 1) there exists an ETD in which the domain entity from which the class was modeled is part of an input flow; 2) the domain entity does not have attributes; 3) the domain entity is not in the IFs of any subsequent ETD, and; 4) the domain entity will be no longer needed in the IS.

For the running example, this rule is not applied.

**Rule C5 (modification method)** A modification method is modeled in a class for each ETD in which the domain entity from which the class was modeled has attributes in an input flow and the creation method of the class was not modeled from the ETD. Its parameters are the attributes of the domain entity in the input flow. A data type is specified for each parameter.

A modification method of “Car” is “return car (km, fuel)”.

**Rule C6 (calculation method)** A calculation method is modeled in a class for each attribute that: 1) belongs to the domain entity from which the class was modeled; 2) is in an output flow, and; 3) does not correspond to an attribute of the class. A return data type is specified for each calculation method.

A calculation method of “Rental Contract” is “calculate rental cost ()”.

**Rule C7 (state change method)** A state change method is modeled in a class for each ETD in which the domain entity from which the class was modeled has different states in the input and output of the ETD and no other method has been modeled in the class from the ETD.

A state change method of “Extra” is “rent extra ()”.

Extended Task Description: CAR PURCHASE			
Input		Output	
Domain Entity	State	Domain Entity	State
Insurance	-	Car	Ready
Rate	-		
Information flows			
(Normal)			
← { Insurance / company + expiration date / } <sub>n</sub>			
→ Insurance			
← { Rate / name + price / } <sub>n</sub>			
→ Rate			
→ Car / plate number + make + model + engine + color + seats + purchase date /			
Extended Task Description: CAR RETURN			
Input		Output	
Domain Entity	State	Domain Entity	State
Rental Contract	Open	Rental Contract	Closed
Car	Rented	Car	Ready   Disabled
Extra	Rented	Extra	Ready
Information flows			
(Normal)			
← { Car / plate number / } <sub>n</sub>			
→ Car / km + fuel /			
→ Rental Contract / return date /			
← Rental Contract / amount to pay /			
(Extension)			
→ Car / disability date /			

Figure 4. Input, output and IFs of ETDs

**Rule C8 (associations)** An association between two classes is modeled if: 1) the domain entities from which the classes were modeled are part of the input or output of a same ETD; 2) there exists a relationship between the entities in the domain data model, and; 3) an association between the classes is created in the ETD.

“Car” and “Rental Contract” are associated from the ETD “Car rental”.

**Rule C9 (minimum multiplicity)** The minimum multiplicity of a class in an association is 0 if the association is not modeled from the ETD from which the creation method of the class was modeled; otherwise, the minimum multiplicity is the minimum number of occurrences of the domain entity in the input flows of the ETD from which the association was modeled.

The minimum multiplicities in “Car - Rental Contract” are 0 for “Car” and 1 for “Rental Contract”.

**Rule C10 (maximum multiplicity)** The maximum multiplicity of a class in an association is the maximum number of occurrences of the domain entity from which the class was modeled in the input flows of the ETD from which the association was modeled; maximum multiplicity may be increased.

The maximum multiplicities in “Car - Rental Contract” are indeterminate (“\*”) for “Car” (increased) and 1 for “Rental Contract”.

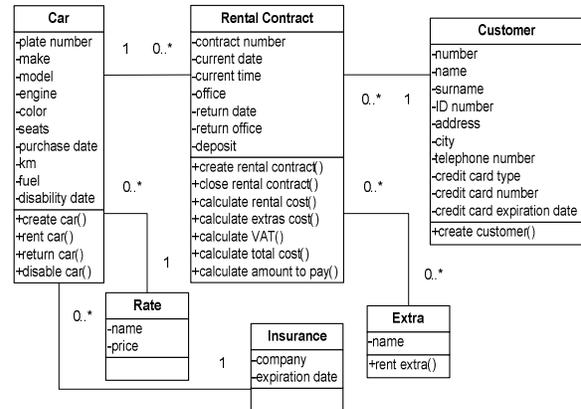


Figure 5. ACD

## 4.2. Rules for derivation of ASTDs

An ASTD is derived for each class of the ACD of an IS from input and output of its ETDs and from the ACD. There are six rules for the derivation, and they allow system analysts to model states and transitions.

Figure 6 shows the ASTD of the class “Car” for the running example. It has been derived from the ETD shown in Figure 1, the input and output of the ETDs shown in Figure 4, and the ACD shown in Figure 5.

The rules are the following ones.

**Rule S1 (initial state)** An initial state is modeled in each ASTD.

**Rule S2 (final state)** A final state is modeled in an ASTD if the class has a deletion method.

**Rule S3 (intermediate states)** An intermediate state is modeled in an ASTD for each state that the domain entity from which the class was modeled can reach in the ETDs.

The intermediate states of “Car” are “Ready”, “Rented” and “Disabled”.

**Rule S4 (first transition)** A transition is modeled in an ASTD from the initial state. The event of the transition is the creation method of the class. The target state is the state of the domain entity from which the class was modeled in the output of the ETD from which the creation method was modeled.

The target state of the first transition of “Car” is “Ready”, and its event is “create car”.

**Rule S5 (last transition)** If an ASTD has a final state, then a transition is modeled to it. The event of the transition is the deletion method of the class. The source state is the state of the domain entity from which the class was modeled in the input of the ETD from which the deletion method was modeled.

For the running example, this rule is not applied.

**Rule S6 (intermediate transitions)** For each method of a class that is not its creation or deletion methods, a transition is modeled in its ASTD. The event of the transition is the method. If the domain entity from which the class was modeled is part of the input of the ETD from which the method was modeled, then the source state is the state of the domain entity in the input; otherwise, the source state is the state of the domain entity in the output of the ETD. The target state is the state of the domain entity in the output.

An intermediate transition of “Car” corresponds to the event “rent car”, whose source state is “Ready” and target state is “Rented”.

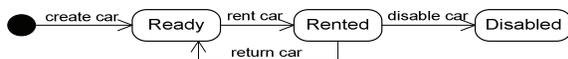


Figure 6. ASTD

## 5. ETD2OO evaluation

As mentioned above, ETD2OO has been applied in several field trials. Their purpose has been to acquire knowledge and to show that it can be used in practice. Stakeholders of the organizations and company analysts participated in the field trials, and several properties have been analyzed and evaluated.

Completeness of OO diagrams has been assessed by comparing diagrams that were derived with ETD2OO

and diagrams that the software development company possessed. Company diagrams corresponded to those that analysts had modeled when the company had developed ISs for the organizations of the field trials.

Company diagrams were larger (had more information), but the reason is the way of modeling. Company analysts model internal characteristics of an IS with OO-Method (e.g. design classes), whereas ETD2OO diagrams are analysis diagrams that are derived from a SRS. As other authors (e.g. [3]), we adopt an external view of an IS when specifying requirements, thus internal characteristics cannot be derived.

Nonetheless, ETD2OO diagrams contained all the external characteristics that analysts had modeled in their diagrams. Therefore, they were complete from a requirements perspective, i.e. they contained all the information that was necessary to meet requirements.

When asking company analysts about ETD2OO usefulness, we obtained different opinions. Although all of them stated that ETD2OO (and the RE approach) allowed them to better understand the requirements and address system modeling, some senior analysts (very skilled in using OO-Method) said that they would probably not use it. They thought that ETD2OO could not improve their job significantly. Nonetheless, junior analysts stated that it could really help them.

Complexity of ETD2OO has also been analyzed. The rules for derivation of OO diagrams can be considered not to be very complex. However, the objective of ETD2OO is not to define a complex process for system modeling from ETDs, but to define a straightforward and precise link that can help system analysts to model ISs that meet business process-based requirements. Otherwise, ETD2OO could be difficult to use, thus the company would not adopt it.

Other issues that are related to ETD2OO and that have been previously discussed are application of the RE approach (including stakeholder’s feedback) [4][6] and completion of ACDs and their similarity with domain data models [5].

## 6. Related work

The closest approaches to ETD2OO are those that address modeling of analysis OO diagrams from software requirements. They usually model class diagrams from use cases or jointly with them, and use mechanisms such as IFs [5][9], consistency guidelines [10], sequence diagrams [11] activity graphs [13] and goals of use cases [14]. Some approaches (e.g. [2]) model classes from diagrams of the i\* framework.

The mechanisms that are used in ETD2OO are input and output of ETDs and IFs. IFs of ETDs are based on

the ones that are presented in [9], and initially they were specified in the same way (a unique IF for each ETD [7] to derive class diagrams [5]). However, several problems arose (e.g complexity and size of IFs and incompleteness in ETDs, ACDs and ASTDs). They have been solved in ETD2OO by specifying an IF for each user and system action.

Homogeneous and adequate requirements granularity is also necessary for correct ACD derivation. On the basis of our experience, problems in associations can arise (identification and multiplicity) if granularity is not properly addressed. However, just [5] and [9] deal with requirements granularity and homogeneity.

Finally, none of the reviewed approaches address derivation of state transition diagrams. Both static and dynamic properties are important for IS modeling [15].

Table 1 summarizes related work and compares it with ETD2OO. Eight criteria are used. The symbols of the cells are: “+”, if an approach properly supports a criterion; “+/-”, if the support that is provided should be improved, and; “-”, if improper or no support is provided. As shown in Table 1, ETD2OO is the only approach that properly supports all the criteria.

**Table 1. Comparison of related work**

	ETD2OO	[2]	[5]	[9]	[10]	[11]	[13]	[14]
BP	+	+/-	+	-	-	-	-	-
HG	+	-	+	+	-	-	-	-
CIM	+	+	+	+	+	+	+/-	+/-
AtM	+	+	+/-	+/-	+	+	+	+
MeM	+	+	+/-	+/-	+	+	+	+
AsM	+	+/-	+	+	+/-	+/-	+/-	+/-
MuM	+	-	+/-	+/-	+/-	+/-	+/-	+/-
SM	+	-	-	-	-	-	-	-

Criteria: modeling and analysis of business processes for elicitation of software requirements (BP); homogeneous granularity of software requirements (HG); class modeling (CIM); attribute modeling (AtM); method modeling (MeM); association modeling (AsM); multiplicity modeling (MuM); state modeling (SM).

## 7. Conclusions and future work

This paper has presented ETD2OO. Its purpose is to integrate business process-based SRS in the form of ETDs into OO IS modeling and development, and two sets of rules have been presented for derivation of an ACD and of ASTDs of an IS. As a result, system modeling from ETDs is facilitated, ETDs are linked to subsequent development stages, and ETDs are now really useful for OO-Method, a methodology for software development that is used in industry.

The paper has also presented a new ETD version and ETD2OO evaluation. The new version addresses quality-related requirements, and IFs are now specified for each user and system action. ETD2OO allows derivation of complete OO diagrams from a requirements perspective and can facilitate analysts' work in a straightforward and precise way.

As future work, tool support for ETD2OO is being developed and further evaluation is planned. We also want to address derivation of abstract user interfaces from ETDs for requirements validation, and derivation of the presentation model (interface) of OO-Method.

## 8. Acknowledgments

This work has been developed with the support of the Spanish Government under the project SESAMO TIN2007-62894 and the program FPU AP2006-02324, and has been co-financed by FEDER.

## 9. References

- [1] Alexander, I., I. Bider, and G. Regev, “Workshop on Requirements Engineering for Business Process Support (REBPS'03)”, CAISE'03 Workshops Proceedings, 355-358
- [2] Castro, J.F., et al., “Integrating Organizational Requirements and Object Oriented Modeling”, RE'01, 146-153
- [3] Davis, A.M., *Software requirements*, Prentice-Hall, 1993
- [4] de la Vara, J.L., J. Sánchez, and O. Pastor, “Business Process Modelling and Purpose Analysis for Requirements Analysis of Information Systems”, CAiSE 2008, 213-227
- [5] de la Vara, J.L., et al., “A Requirements Engineering Approach for Data Modelling of Process-Aware Information Systems”, BIS 2009, 133-144
- [6] de la Vara, J.L., and J. Sánchez, “BPMN-Based Specification of Task Descriptions: Approach and Lessons Learnt”, REFSQ 2009, 124-138
- [7] de la Vara, J.L., and J. Sánchez, “Specification of Data Requirements from Task Descriptions”, SEKE 2009, 55-60
- [8] Dumas, M., W. van der Aalst, and A. ter Hofstede, *Process-Aware Information Systems*, Wiley, 2005
- [9] Fortuna, M., C. Werner, and M. Borges, “Info Cases: Integrating Use Cases and Domain Models”, RE'08, 81-84
- [10] Glinz, M., “A Lightweight Approach to Consistency of Scenarios and Class Models”, ICRE'00, 49-58
- [11] Insfran, E., O. Pastor, and R. Wieringa, “Requirements Engineering-Based Conceptual Modelling”, *Requirements Engineering* 7(2), 61-72, 2002
- [12] ISO, International Standard ISO/IEC 9126-1: Software engineering – Product quality – Part 1: Quality Model, 2001
- [13] Kösters, G., H.W. Six, and M. Winter, “Coupling Use Case and Class Models for Validation and Verification of Requirements Specifications”, *Requirements Engineering* 6(1), 3-17, 2001
- [14] Liang, Y., “From use case to classes: a way of building object model with UML”, *Information and Software Technology* 45(2), 83-93, 2003
- [15] Pastor, O., and J.C. Molina, *Model-Driven Architecture in Practice*, Springer, 2007
- [16] Verner, J., et al. “Requirements engineering and software project success: an industrial survey in Australia and the U.S”, *Australasian Journal of Information Systems* 13(1), 225-238, 2005

# Specification patterns can be formal and still easy

Fernando Asteasuain  
FCEyN-University of Buenos Aires  
Email: fasteasuain@dc.uba.ar

Víctor Braberman  
FCEyN-University of Buenos Aires  
Email: vbraber@dc.uba.ar

**Abstract**—Property specification is still one of the most challenging tasks for transference of software verification technology like model checking. The use of patterns has been proposed in order to hide the complicated handling of formal languages from the developer. However, this goal is not entirely satisfied. When validating the pattern the developer may have to deal with the pattern expressed in some particular formalism. For this reason, we identify three desirable quality attributes for the underlying specification language: *succinctness*, *ease of validation* and *modifiability*. We show that typical formalisms such as temporal logics or automata fail at some extent to support these features. In this work we propose FVS, a graphical scenario-based language, as a possible alternative to specify behavioral properties. We illustrate FVS' features by describing one of the most commonly used pattern, the Response Pattern, and several variants of it. Other known patterns such as the Precedence pattern and the Constrained Chain pattern are also discussed. We also thoroughly compare FVS against other used approaches.

## I. INTRODUCTION

Property specification is still one of the most challenging tasks for transference of software verification technology like model checking. Users of these techniques must still face the challenge of expressing properties in the language or formalism used in the specification tool. Using natural language to express requirements may appear as an alternative to formal approaches, but in general leads to ambiguous and imprecise specifications, threatening the advantages of an automated verification process. Two of the most common formal approaches are temporal logics such as Linear Temporal Logic (LTL) and operational notations such as finite-state machines. In these approaches final users are required to have a considerable expertise on the formalism to accurately express the requirement they want to express ([10], [12], [9], [21], [6], [18], [2]) and this clearly constitutes not a minor obstacle.

The usage of specification patterns has been proposed as an interesting alternative to overcome this problem [9], [8]. The main purpose of a pattern is to capture recurring solutions to a particular type of problem. In [9], patterns are described considering two aspects. On the one hand, the *intent* of the pattern is described, that is, the structure of the defined behavior. This is usually expressed using a disciplined natural language (DNL) notation [21]. For example, the intent for the *Response* pattern is denoted as “One or more occurrences of **action** result in one or more occurrences of **response**”. On the other hand, each pattern has a scope, which is the extend of the program execution over which the pattern must hold. For example, we can say that the pattern must hold between

the occurrence of two given events. Although patterns offer a friendlier way to express typical requirements, in order to determine if the pattern is accurately expressing the desired property, the developer usually must deal with not only the DNL description of the pattern, but its translation into an underlying formalism such as LTL [12], [21]. This situation also arises when deciding the right pattern to be employed, or even more important, when validating the usage of the pattern. Therefore, using patterns is not enough to entirely hide the subtleties of the underlying formalism from the user. This suggests that the specification language should be easy to use, and sufficiently expressive to enable skilled and non-skilled users to use it appropriately [16], [12]. More specifically, we define three quality attributes desirable for the formalism: *succinctness*, *ease of validation* and *modifiability*. The first one requires that the specification of a pattern expressed in the formalism should be as succinct as the DNL description of the pattern. Ease of validation estimates how simple it is to determine if the specification of a pattern actually expresses the desired property. This attribute can be further subdivided into two sub-attributes: *comparability* and *complementariness*. The resulting specification of the patterns should be easy to compare and distinguish. For example, when comparing two patterns it is natural to try to understand which one is more restrictive. Complementariness refers to the ability of reasoning about how the property is violated, which provides meaningful information to the developer. The formalism should provide an easy way to reason about complementary behavior. Finally, *modifiability* is the ability to manipulate objects expressed in the formalism, so that the pattern can be adapted to subtle modifications in the application context.

We believe that the provided mappings of the patterns to LTL detailed in [9], [8] fail at some extent to support these features. The resulting LTL formulae may result in complicated artifacts, which are difficult to compare without deductive manipulation. Reasoning about complementary behavior is also troublesome since it requires sophisticated formulae manipulation. If the underlying formalism is an automata-based one instead, the analysis is similar. In order to accurately compare two automata language inclusion needs to be tested. Similarly, operations to complement the language of an automaton are not trivial and may suffer from exponential state-explosion problems. Modifiability also follows an analogous reasoning. Intricate operations may be needed to modify a LTL formula or an automaton to adapt to different situations.

Given this context we propose a graphical language based

on scenarios which aims to improve and ease the property specification process. The language, called FVS (Featherweight Visual Scenarios) is a simple fragment of VTS (Visual Timed Scenarios) [5], a visual language to define complex event-based requirements. We show that FVS, although simple, is powerful enough to properly describe specification patterns. For one side, its visual aspect will help the user concentrate in the properties themselves instead of dealing with the subtleties of their formalization. The scenarios are built using just a few and simple elements, therefore obtaining compact and minimal specifications. Semantic and logical relationships between the elements involved can be extracted directly from the scenarios, improving reasoning about patterns. The language supports the automatic construction of anti-scenarios, which help the user in the property specification by looking at behavior that leads to the violation of the property. What's more, patterns specification is very flexible and can be easily adapted to different application contexts. Concretely, we analyze FVS as a specification language by describing some of the patterns introduced by [9] with an special focus on one of the most used pattern, namely the Response Pattern, and several variants of it. The Constrained Chain Pattern and the Precedence pattern are also modeled in FVS. We compare FVS against the original patterns specification in [9] and also against the Propel language [21], an approach that define one of the most known extensions to patterns specification. Propel uses two notations: an extended finite-state automaton representation and a DNL representation, based on a restricted subset of natural language. The comparison is based on the previously defined concepts: succinctness, ease of validation and modifiability.

The rest of the paper is structured as follows. After describing some initial background, FVS is explained in section II. Section III shows the specification of the Response Pattern (including several variants of it), the Constrained Chain pattern and the Precedence pattern in FVS. Analysis and comparison with other formalisms are presented in section IV. The paper concludes mentioning future work and conclusions.

### A. Background

Using patterns for property specification was first introduced by [9]. According to the authors, patterns fall into two main categories based on their semantics: *Occurrence*, where patterns require events to occur or not to occur and *Order*, where patterns constrain the order of events. The Absence pattern and the Universality pattern are examples of the first category, whereas Precedence and Response are examples of the second category. The complete list of the patterns is available online [8] where also mappings to different formalisms are supplied. In this work we will focus only on their mapping to LTL, being one of the most used formalisms for specification. The LTL formulae discussed in this work refer to the LTL mapping specified in [8]. The other formalism discussed here is the Propel Language [21]. Propel presents a very exhaustive extension to some of these patterns, which covers additional issues regarding patterns behavior. FVS is thoroughly compared to

both approaches.

## II. FEATHER WEIGHT VISUAL SCENARIOS

In this section we will informally describe the standing features of FVS. The reader is referred to [5] for a formal characterization of the language. FVS is a graphical language based on scenarios. Scenarios consist of points, which are labeled with the possible events occurring at that point, and arrows connecting them. An arrow between two points indicates precedence of the source with respect to the destination: for instance, in figure 1-(a) *A*-event precedes *B*-event. We use an abbreviation for a frequent sub-pattern: a certain point represents the next occurrence of an event after another. The abbreviation is a second (open) arrow near the destination point. For example, in figure 1-b the scenario captures the very next *B*-event following an *A*-event, and not any other *B*-event. Conversely, to represent the previous occurrence of a (source) event, there is a symmetrical notation: an open arrow near the source extreme. In figure 1-c the scenario captures just the immediately previous *A*-event from *B*-event. Events labeling the arrow are interpreted as forbidden events between both points. In figure 1-d *A*-event precedes *B*-event such that *C*-event does not occur between them. The abbreviations presented before can also be expressed using labeled arrows. Scenario in figure 1-e shows an equivalent version of the scenario in figure 1-b. Finally, two distinguished points are introduced to denote the beginning and the end of the trace: a big full circle for *begin*, and two concentric circles for *end* (shown in figure 1-f).

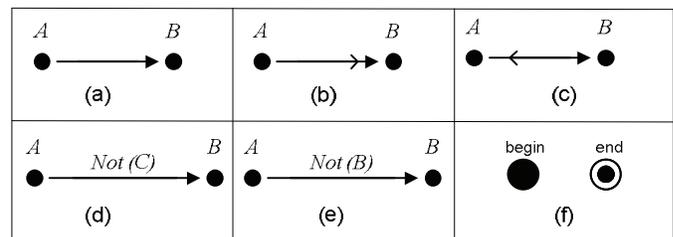


Fig. 1. Basic Elements in FVS

### A. FVS Rules

We now introduce the concept of Rule Scenario (RS) or simply a rule<sup>1</sup>, a core concept in the language. Roughly speaking, a rule is divided into two parts: a scenario playing the role of an antecedent and at least one scenario playing the role of a consequent. The intuition is that whenever a trace “matches” a given antecedent scenario, then it must match at least one of the consequents. In other words, rules take the form of an implication: an antecedent scenario and one or more consequent scenarios. The antecedent is a common substructure of all consequents, enabling complex relationship between points in antecedent and consequents: our rules are not limited,

<sup>1</sup>Rule Scenarios represent the FVS's version of Conditional Scenarios available in VTS [5]

like most triggered scenario notions, to feature antecedent as a pre-chart where events should precede consequent events. Thus, rules can state expected behavior happening in the past or in the middle of a bunch of events. Graphically, the antecedent is shown in black, and consequents in grey. Since a rule can feature more than one consequent, elements which do not belong to the antecedent scenario are numbered to identify the consequent they belong to. An example is shown in figure 2. The interpretation of the rule is that, whenever an *Access request* event is followed by an *Access granted* event (without a logoff in between), one of two other event sequences must be observed too. One of them (consequent1) requires that, after the access has been requested, a valid password is entered. The other one (consequent 2) allows for the case where a valid password has been entered before access to the resource is requested. Observe the power of our trigger notation, where the antecedent need not precede the consequents in time.

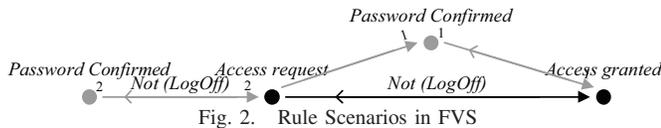


Fig. 2. Rule Scenarios in FVS

### B. Anti-Scenarios

An interesting feature in FVS is that anti-scenarios can be automatically generated from rule scenarios. This is valuable information for the developer since it represents a sketch of how things could go wrong and violate the rule. The complete procedure is detailed in [5], but informally the algorithm computes all possible situations where the antecedent is found, but none of the consequents is matchable. One anti-scenario for the rule in figure 2 is shown in figure 10. In this case, a valid password was not entered since the beginning of the trace and still access is granted.

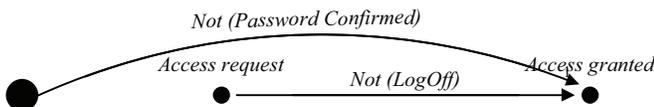


Fig. 3. An anti-scenario in FVS

## III. PATTERN SPECIFICATION IN FVS

In this section we illustrate how FVS may describe some of the specification patterns introduced by [9]: the Response Pattern, the Constrained Chain pattern and the Precedence pattern. One of the most widely used patterns is the *Response* pattern, in which the occurrence of one event (referred as the *Action* event), leads to an occurrence of another event, the *Response* event. This illustrates a cause-and-effect relationship between the events involved. Examples of the Response pattern may be requirements such as “Every client request is acknowledged by the server” or “every time the doors are opened the lights are turned on”. The rule depicted in figure 4 reflects



Fig. 4. The basic Response Pattern

this behavior. As pointed out in [21], a closer examination of this pattern’s behavior reveals there are significant questions about the pattern that need to be answered and may lead to variants of it. For example, can the *Action* event occur more than once before the *Response* event occurs? Taking this fact into consideration, [21] proposes an extension to the patterns’ specification, defining six possible options to obtain a more accurate and complete definition of the pattern: **Pre-arity**, which determines whether the *Action* event may occur one time or many times before the *Response* event does, **Post-arity**, which determines whether the *Response* event may occur one time or many times after the *Action* event does, **Immediacy**, which determines whether or not other intervening events may occur between the *Action* event and the *Response* event, **Precedence**, which determines whether or not the *Response* event is allowed to occur before the first occurrence of the *Action* event, **Nullity**, which determines whether or not the *Action* event must ever occur and finally, **Repeatability**, which determines whether or not occurrences of the *Action* event after an occurrence of a *Response* event are required to be followed by a *Response* event. In what follows we model all six cases in FVS.

1) *Pre-arity*: the rule in figure 4 models the situation where the *Action* event can occur several times before the *Response* event (by default in FVS, any arbitrary *Action* event may become a valid antecedent to trigger the rule). To limit the behavior such that *Action* event can not be repeated before *Response* event, a restriction is added to the rule. The condition prohibits the occurrence of another *Action* event until the occurrence of the *Response* event. This is shown in figure 5-a.

2) *Post-arity*: the rule depicted in figure 4 also models the situation where the *Response* event can be repeated. If *Response* event cannot be repeated, then between any two consecutive *Response* events there must occur an *Action* event. In this way, traces containing two or more consecutive *Response* events after an *Action* event will be excluded. Figure 5-b illustrates this behavior.

3) *Immediacy*: forbidden events between *Action* and *Response* are simply added to the pattern by including the forbidden events properly. This is illustrated in figure 5-c.

4) *Precedence*: this is achieved by requiring that the occurrence of a *Response* event must always be preceded by an *Action* event. The rule in figure 5-d reflects this behavior.

5) *Nullity*: the initial scenario for the Response pattern (figure 4) covers the case where the *Action* event is not required to occur. Demanding the occurrence of the *Action* event implies that the *Action* event must occur after the beginning of the trace. This is shown in figure 5-e.

6) *Repeatability*: the initial scenario models a repeatable behavior, since it requires a *Response* event for every *Action*

event. To model the case where the pattern must be fulfilled only for the first occurrence of the *Action* event, a restriction is added so that the antecedent contemplates only the first, and not each, occurrence of the *Action* event. The antecedent in figure 5-f demands that there is no previous occurrence of the *Action* event from the beginning of the trace. Clearly, only the first *Action* occurrence will satisfy this condition.

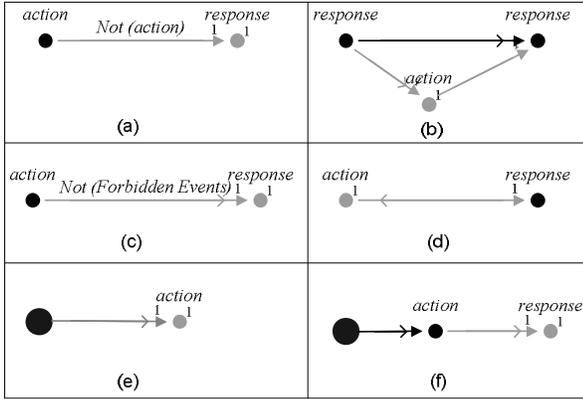


Fig. 5. Six rules covering the extended behavior

The rules presented in figure 5 model each aspect individually. Naturally, two or more aspects can be combined by simply including their corresponding rule.

### A. Modeling Scopes

The previous section describes what the work in [9] defines as the intent of the pattern, that is, the structure of the defined behavior. This notion can be completed by defining a scope for the pattern, establishing the extent of program execution over which the pattern must hold. The available scopes defined in [9] are: **Before P** (the pattern must hold before the first occurrence of an event  $P$ ), **After Q** (the pattern must hold only after the first occurrence of an event  $Q$ ), **Between P and Q** (the pattern must hold after the occurrence of  $P$  but before the occurrence of  $Q$ ), **After Q until P** (similar to the previous one, but  $P$  is not required to occur) or **Global**, which denotes the whole execution.

Scopes are introduced in FVS in the same way as any other restriction of behavior. Thus, Global scope is implicitly modeled by introducing no scope restrictions (all the rules shown up to here assumed Global scope). In what follows we model the rest of the scopes for the basic Response pattern<sup>2</sup>.

1) *Before P - After Q*: The rule in figure 6-a corresponds to the Before P scope. The occurrence of the property must precede the first occurrence of  $P$ . This is, if an Action event occurs before the first  $P$  occurrence, then a Response event must also occur before  $P$ . Similarly, the scenario in figure 6-b shows the After Q scope. That is, if an Action event occur after the first  $Q$  occurrence, then a Response event must also occur afterwards.

<sup>2</sup>Note that extra considerations such as pre-arity or post-arity can be added to the rules as shown in the previous section.

2) *After Q until P - Between P and Q*: Figure 6-c shows the After Q until P scope. In this case, an Action event occurring after  $Q$  must be followed by a Response event, but the Response event must occur before  $P$ . Note that the rule can be satisfied without the occurrence of  $P$ . Figure 6-d shows the between P and Q scope. In this case, both delimiters must occur, namely P and Q.

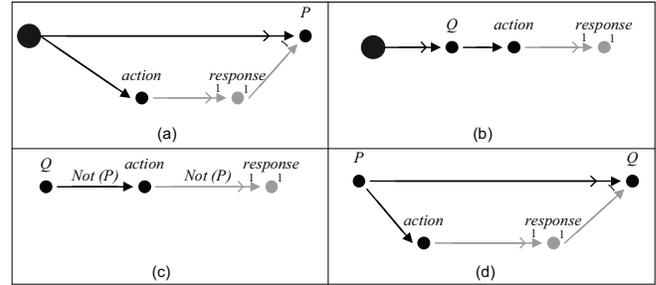


Fig. 6. Different scopes for the Response Pattern

### B. Response Chain Pattern

We now consider a generalization of the Response pattern: the *Response-Chain* pattern. In this pattern a sequence of events  $P_1, P_2, \dots, P_n$  must always be followed by a sequence of events  $Q_1, Q_2, \dots, Q_n$ . For simplicity and space reasons, we only consider one case of this pattern, where one stimuli must be followed by two responses, considering Global (figure 7-a) and Between P and Q scopes (figure 7-b).

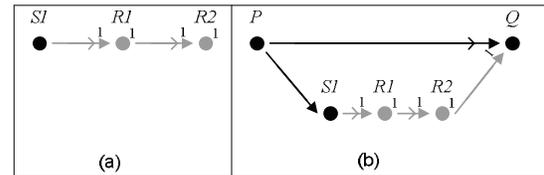


Fig. 7. One Stimuli-Two Responses Pattern

Note that the only difference with the rules describing the basic Response pattern (one stimuli - one response), shown in figures 4 (for Global scope) and 6-d (for Between scope), is just the inclusion of the second response, keeping the pattern specification simple and succinct.

### C. Constrained Chain Pattern

This pattern introduces a variant for the Response-Chain pattern. In this case, the pattern restricts user specified events from occurring between pairs of events in the chain sequences. As we have already shown, restricting behavior is added very simply in our graphical language, as a label between the events involved. The example shown in this section is an extension for the One Stimuli-Two Responses Pattern, where a  $W$  event must no occur for the property to hold. The scopes modeled are Global (figure 8-a) and Between P and Q (figure 8-b).

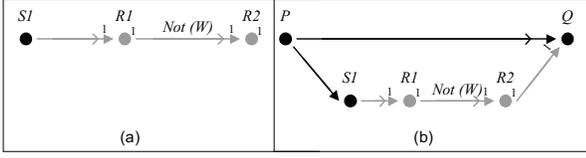


Fig. 8. Constrained Chain Pattern in FVS

#### D. Precedence Pattern

To conclude this section, we introduce another pattern: the Precedence pattern. The intent of this pattern is to describe relationships between a pair of events/states where the occurrence of the first is a necessary pre-condition for an occurrence of the second [8]. In this case, the occurrence of a  $S$  event must precede the occurrence of a  $R$  event. The scopes modeled are Global (figure 9-a) and Between  $P$  and  $Q$  (figure 9-b).

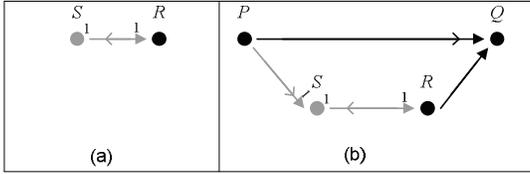


Fig. 9. Precedence Pattern in FVS

### IV. ANALYSIS AND COMPARISON

We now compare FVS against the mentioned approaches considering the three quality attributes defined initially.

1) *Succinctness*: to compare FVS against the resulting LTL formulae in [8] and Propel's notation we propose the following alternative to somehow measure the complexity of the objects expressed in the formalisms. For the LTL formulae in [8], we measure their depth of nesting (DN) and the number of operators involved (O). For Propel's automata notation we measure the number of states (St) and number of transitions (T) of the automata described in [21] and for Propel's DNL templates we measure the numbers of statements (S) needed to express the property as detailed in [21]. For FVS rules we measure the number of points (P) and the number of restrictions (R) specified, including precedence and forbidden events. We compare the basic Response pattern with the following characterization (expressed in Propel's notation):

- Core Phrase (with Pre-arity, Immediacy and Post-arity options): One or more occurrences of *Action* eventually result in one or more occurrences of *Response*.
- Nullity Phrase: *Action* may occur zero times.
- Precedence Phrase: *Response* may occur before the first *Action* occurs.
- Repetition Phrase: The behavior is repeatable.

Regarding scopes, we take into account only Global and Between scopes. Tables I and II exhibit the obtained results for the Succinctness attribute. Table I is focused on the basic Response Pattern whereas table II considers the Response

Chain pattern and the Constrained Chain pattern with one stimuli and two responses, and the Precedence pattern. In this case, FVS is only compared against LTL formulae: the first two are not available in Propel [6] whereas the Precedence Pattern is not completely described in [21].

TABLE I  
COMPLEXITY COMPARISON FOR THE BASIC RESPONSE PATTERN

Formalism	Global	Between
LTL	DN=3,O=4	DN=4,O=12
Propel-Automata	St=3,T=6	St=5,T=12
Propel-DNL	S=7	S=10
FVS	P=2,R=1	P=4,R=4

TABLE II  
COMPLEXITY COMPARISON FOR THE RESPONSE CHAIN PATTERN, THE CONSTRAINED CHAIN PATTERN AND THE PRECEDENCE PATTERN

	LTL		FVS	
	Global	Between	Global	Between
R-Chain	DN=2,O=6	DN=5,O=14	P=3,R=2	P=5,R=5
C-Chain	DN=3,O=9	DN=6,O=18	P=3,R=3	P=5,R=6
Precedence	DN=3,O=10	DN=5,O=13	P=2,R=1	P=4,R=4

One way of analyzing succinctness is to examine how the final objects grow when scope restrictions are added. As it is shown in tables I and II, the LTL formulae as given in [8] grow significantly when a more intricate scope is involved. Propel's automata representation also become more complicated, especially due to the growth of the number of transitions. Conversely, scopes in FVS are introduced seamlessly, just as any other restriction in the rules, without affecting succinctness. Propel's DNL notation handles scopes adequately. However, Propel suffers from some limitations when modeling scopes: delimiters in the scopes must be distinct, and there must be no intersection between events defining the intent and the scope of the pattern. These limitations are not present in FVS. Another interesting analysis comes up considering the Response Chain pattern with one stimuli and two responses. The only difference with the basic Response pattern is the inclusion of a second response. However, this simple modification causes a meaningful growth in the resulting LTL formulae, seriously threatening the scalability principle. On the other hand, FVS rules maintain a suitable complexity scaling appropriately. LTL formulae may be expressed more naturally or more succinctly employing modalities such as past or "from now on" operators [14], [15], but this require non trivial formulae manipulation, or even exponential blow-ups during the process [17].

2) *Ease of Validation*: in order to validate, understand and compare different patterns, the specification of the patterns expressed in the formalism must be easy to handle, manipulate and compare. These characteristics constitute the *Comparability* sub-attribute. This goal is hard to achieve when dealing with complicated LTL formulae or when comparing automata with several states and transitions. Formally, this would require testing language inclusion for automata or employing deductive simplification mechanisms for LTL formulae. The

situation is different in FVS' specifications. For example, the rule for the Response Chain pattern is the natural extension to the rule for the basic Response-pattern and this relationship can be visually depicted without extra manipulation. To mention another example, recall the Constrained Chain pattern in figure 8. The difference between the constrained version and the unconstrained version in figure 7 is clear when comparing both scenarios: the inclusion of the restriction through labeled arrows.

More elaborated analysis can also be gathered visually. As an example, recall the Response Chain pattern example considering Between scope (figure 7-b) and the Constrained Chain pattern considering the same scope (figure 8-b). In both rules antecedents are equivalent, but the consequent in figure 8-b is "stronger" than the consequent in figure 7-b, since it features more constrains. Thus, the rule depicted in figure 8-b is a specialization of the rule described in figure 7-b. The specialization relationship is a notion is similar to logical subsumption [4]. This also holds for the rules describing both patterns with Global scope (figures 8-a and 7-a). Now, when comparing rules in figure 7-a and 7-b it can be seen that although the consequent in figure 7-b is stronger than the consequent in figure 7-a, it is also the case that its antecedent is stronger too. Therefore in this case there is no specialization relationship. On the other hand, this kind of analysis is difficult to achieve when using an automaton notation or an LTL formula without proper manipulation. For example, the LTL version for the rules described in figures 7-a and 7-b presented in [8] are:  $(\Box S1 \rightarrow \Diamond (R1 \wedge X \Diamond R2))$ , for figure 7-a, and  $\Box ((P \wedge \Diamond Q) \rightarrow (S1 \rightarrow (\neg QU(R1 \wedge \neg Q \wedge X(\neg QU R2))))UQ)$ , for figure 7-b. By just looking at these formulae it is hard to recognize and understand the semantic relationship between them.

The other attribute composing the Ease of Validation attribute is *Complementariness*. FVS supports this feature by automatically generating anti-scenarios. Figure 10 illustrates two anti-scenarios for the Response pattern and Between Scope (in figure 6-d): figure 10-a models the case where the *Response* event did not occur in the trace after an *Action* event whereas in figure 10-b, the *Response* event did occur, but after the occurrence of *P*. Complementary behavior can be achieved by negating a formula or complementing an automaton, but this may involve non trivial operations.

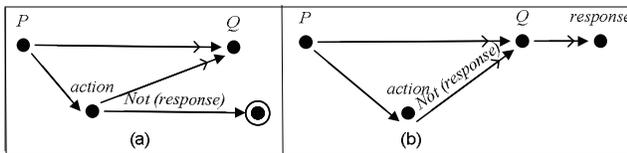


Fig. 10. Anti-scenarios for the Response pattern with Between scope

3) *Modifiability*: One interesting thing that can be observed in the Response Chain pattern is that the sequence of events involved must follow a strict order. In the example used here,

with one stimuli *SI* and two responses *R1* and *R2*, response event *R1* must precede response event *R2*. For some situations we would like to relax this condition, and to allow responses to occur in any order. This is easily achievable in FVS, since the only difference with the regular case is that no precedence restriction is present between both *Response* events. The rule in figure 11 shows this version considering Global scope.

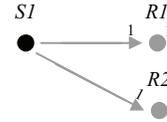


Fig. 11. A relaxed version of the One Stimuli-Two Responses Pattern

Finally, we conclude this section introducing another useful variant of the Response Chain pattern, where responses may occur before or after delimiter *Q*. In this case, the interpretation of the pattern is the following: an occurrence of stimuli *SI* between *P* and *Q* must always be followed by responses *R1* and *R2*. This version is suitable, for example, for modeling *race conditions*, a typical situation in multithreaded or concurrent systems. The rule in figure 12 reflects this behavior.

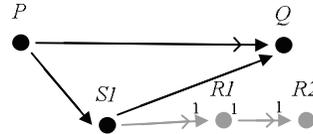


Fig. 12. A Variant of the One Stimuli-Two Responses Pattern

These examples show how patterns in FVS can be easily modified to fit in different situations or contexts. These variants of the Response Chain pattern are neither available in Propel nor in the LTL formulae available at [8]. With proper manipulation the developer could get an equivalent automaton or LTL formula, but again, this is bound to a complicated task.

## V. RELATED WORK

TimeEdit [20] and GIL (Graphical Interval Logic) [7] are two graphical specification languages based on timeline diagrams that do not feature partial ordering of events. TimeEdit is particularly focused on capturing complex chain events [2], while FVS stands for a more general approach. TimeEdit features a restricted notion of triggered scenarios using *required* events (events that are required to occur if all previous events have occurred). This limitation makes properties about past events, or events occurring in a certain scope, harder to specify and understand. GIL provides search operators to locate end points of intervals, similar to *next* and *previous* in FVS. However, it *previous* operator can not be applied freely as in FVS: interval recognition starts always forward a generic (or the first) point in the enclosing interval. Thus, easily expressible situation in FVS like freshness, correlation constrains or asserting the existence of a past in general can not be stated in GIL. Finally, specification of complex properties involving

several events in GIL requires nesting or stacking operators, threatening succinctness, ease of validation, and modifiability quality attributes. Other worth-mentioning approach is PSC (Property Sequence Chart) [2], which is inspired in UML 2.0 Interaction Sequence Diagrams. PCS's notation is also validated modeling property specification patterns. As said by the authors, it might be difficult to directly express properties in this language, and some automated assistance tool may still be need to help the developer [3]. Denoting complex constrains between events may require textual annotations. In addition, properties in PCS are described as anti-scenarios (e.g [1]) and not as conditional or triggered scenarios.

Other visual formalisms based on Message Sequence Charts such as [19], [22], [11] have been proposed for scenario-based specifications. We share with them the idea of using partial orders to describe scenarios. However, our work differs in several aspects. Our language is meant to express properties to be checked against a model or an implementation under analysis; we do not focus on creating an executable modeling language for different phases of the development process. FVS's trigger notation is distinguishable too: in our approach, the antecedent pattern is not required to predicate on a prefix of the behavior. Our consequents can refer to events occurring before the trigger or interleaved with its events.

Finally, to our best knowledge, none of the previously mentioned approaches is equipped with deductive features for comparability or complementariness reasoning.

## VI. CONCLUSIONS AND FUTURE WORK

Property specification is one of the most challenging task for the transference of software verification techniques. In this sense, the use of patterns has been proposed in order to hide from the developer the complicated handling of formal languages. However, when validating whether the pattern correctly expresses the desired property, the developer might face the translated version of the pattern into some specification's formalism. In this respect we identify three desirable quality attributes for the underlying formalism: *succinctness*, *ease of validation*, and *modifiability*. We show that typically used formalisms such as temporal logics or automata fail at some extent to support these characteristics. We propose FVS as a possible alternative to specify behavioral properties and we assess its performance by comparing it with two known approaches using one of the most commonly used pattern, the Response Pattern, and several variants of it. Other patterns such as the Constrained Chain pattern and the Precedence pattern are also considered. Regarding future work, we are considering enhancing FVS's expressivity power to enable expressing arbitrary  $\omega$ -regular languages. We are also working on defining a synthesis algorithm for FVS's rules, enabling the possibility of elaborated automatic analysis. Finally, since VTS can express real time properties, we would like to explore real time specification patterns [13], an extension to the patterns introduced by [9] considering timing requirements.

## ACKNOWLEDGMENTS

This work was partially funded by PICT 32440, PAE-PICT-2007-02278:(PAE 37279), PIP 112-200801-00955, UBACyT X021 and STIC-AmSud project TAPIOCA. Both authors are also affiliated to CONICET.

## REFERENCES

- [1] A. Alfonso, V. Braberman, N. Kicillof, and A. Olivero. Visual timed event scenarios. In *Proceedings of the 26th International Conference on Software Engineering*, page 177. IEEE Computer Society, 2004.
- [2] M. Autili, P. Inverardi, and P. Pelliccione. Graphical scenarios for specifying temporal properties: an automated approach. *Automated Software Engineering*, 14(3):293–340, 2007.
- [3] M. Autili and P. Pelliccione. Towards a graphical tool for refining user to system requirements. *Electronic Notes in Theoretical Computer Science*, 211:147–157, 2008.
- [4] V. Braberman, D. Garbervestky, N. Kicillof, D. Monteverde, and A. Olivero. Speeding Up Model Checking of Timed-Models by Combining Scenario Specialization and Live Component Analysis. In *FORMATS*, page 72. Springer, 2009.
- [5] V. Braberman, N. Kicillof, and A. Olivero. A scenario-matching approach to the description and model checking of real-time properties. *IEEE TSE*, 31(12):1028–1041, 2005.
- [6] R. Cobleigh, G. Avrunin, and L. Clarke. User guidance for creating precise and accessible property specifications. In *Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering*, page 218. ACM, 2006.
- [7] L. Dillon, G. Kutty, L. Moser, P. Melliar-Smith, and Y. Ramakrishna. A graphical interval logic for specifying concurrent systems. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 3(2):131–165, 1994.
- [8] M. Dwyer, G. Avrunin, and J. Corbett. "Specification Patterns Web Site". In <http://patterns.projects.cis.ksu.edu/documentation/patterns.shtml>.
- [9] M. Dwyer, G. Avrunin, and J. Corbett. Patterns in property specifications for finite-state verification. In *Proceedings of the 21st International Conference on Software Engineering ICSE*, volume 99, 1999.
- [10] D. Giannakopoulou and J. Magee. Fluent model checking for event-based systems. In *Proceedings of the 9th European software engineering conference held jointly with 11th ACM SIGSOFT international symposium on Foundations of software engineering*, page 266. ACM, 2003.
- [11] D. Harel and R. Marelly. Playing with time: On the specification and execution of time-enriched lscs. In *MASCOTS '02*, pages 193–202. IEEE Computer Society.
- [12] G. Holzmann. The logic of bugs. *ACM SIGSOFT Software Engineering Notes*, 27(6):87, 2002.
- [13] S. Konrad and B. Cheng. Real-time specification patterns. In *Proceedings of the 27th ICSE*, pages 372–381. ACM, 2005.
- [14] F. Laroussinie, N. Markey, and P. Schnoebelen. Temporal logic with forgettable past. In *Proceedings- Symposium on Logic in Computer Science*. pp. 383-392. 2002, 2002.
- [15] N. Markey. Temporal logic with past is exponentially more succinct. *EATCS Bull*, 79:122–128, 2003.
- [16] D. Paun and M. Chechik. Events in linear-time properties. In *Proceedings of 4th International Conference on Requirements Engineering*. Citeseer, 1999.
- [17] M. Pradella, P. San Pietro, P. Spoletini, and A. Morzenti. Practical model checking of LTL with past. In *ATVA03: 1st Workshop on Automated Technology for Verification and Analysis*. Citeseer, 2003.
- [18] R. W. R. and K. Viggers. Implementing protocols via declarative event patterns. In *ACM Sigsoft International Symposium on FSE(FSE-12)*, pages 158–169, 2004.
- [19] B. Sengupta and R. Cleaveland. Triggered message sequence charts. In *SIGSOFT FSE*, pages 167–176, 2002.
- [20] M. Smith, G. Holzmann, and K. Etessami. Events and constraints: A graphical editor for capturing logic requirements of programs. In *Proceedings of the 5th IEEE International symposium on Requirements Engineering*, pages 14–22. Citeseer, 2001.
- [21] R. Smith, G. Avrunin, L. Clarke, and L. Osterweil. Propel: An approach supporting property elucidation. In *ICSE*, volume 24, pages 11–21, 2002.
- [22] S. Uchitel, J. Kramer, and J. Magee. Negative scenarios for implied scenario elicitation. In *Proc. of FSE '02*, pages 109–118. ACM Press, 2002.

# A Context Conceptual Model for a Distributed Software Development Environment

Ana Paula Chaves  
*System Analysis and Development*  
*Faculdade Integrado Campo Mourao - Brazil*  
*chavesana@gmail.com*

Elisa H. M. Huzita  
*Dept. of Informatics*  
*UEM - Brazil*  
*elisa@din.uem.br*

Vaninha Vieira  
*Dept. of Comp Sciences*  
*UFBA - Brazil*  
*vaninha@ufba.br*

Igor Steinmacher  
*Coord. of Informatics*  
*UTFPR - Brazil*  
*igorfs@utfpr.edu.br*

**Abstract**—Organizations seeking lower costs, quality software and specialized resources started to use an approach called Global Software Development (GSD). However, this approach also brought some drawbacks imposed by the physical distribution, including issues related to communication, cooperation and coordination. There is a need to improve user awareness related to the process and the context in which collaborative artifacts are generated. In this context, conceptual modeling is relevant since conceptual specifications can be used to support understanding, problem-solving, and communication among stakeholders about a given subject domain. This paper presents a conceptual model focused on contextual information for DiSEN, a distributed software engineering environment. Main contributions found on the paper are: (i) the identification of a set of information that compose environment entities context ; and (ii) the definition of a model that facilitates domain semantics comprehension, reducing communication gaps.

**Keywords**-Context modeling; Global software development; conceptual modeling

## I. INTRODUCTION

Global Software Development (GSD) was introduced to attend software industry quality requirements in a competitive way. Allowing collaborative interaction among geographically distributed development teams became an alternative to organizations looking for competitive advantages, such as time to market, better resource usage, increased productivity and reduced costs. However, GSD also brings many new challenges such as contextual, cultural, organizational, geographical, temporal, and political differences [1]. In order to minimize these differences, one fundamental aspect to focus is the communication. To ensure that geographically distributed individuals are collaborating, it is essential to have an infrastructure that guarantees information and knowledge exchange among all involved parties.

According to Pozza [2], interaction frequency among geographically distributed individuals occurs according to the task they are working on, when there are methods to offer awareness over activities performed by other team members. However, to have a full knowledge on the contributions generated by other team members it is necessary to know not only the collaborative artifact itself, but the way and the context it was produced as well.

DiSEN (Distributed Software Engineering Environment) [3] is an environment aiming to provide an infrastructure to

support distributed software development. One of the goals of DiSEN is to provide meaningful and appropriate information regarding entities context during artifacts production. In this sense, DiSEN-CSE (DiSEN-Context Sensitive Environment) [4] was proposed. DiSEN-CSE is integrated to DiSEN and is focused on providing elements to capture, represent, process and disseminate context information to interested individuals. But, to allow context sharing, it is necessary to know what information is part of the context of a given action, how these information are related and how they affect environment behavior.

In order to facilitate contextual information handling, some representation models were created, focusing on offer information to different entities (people, software or devices) in a way that any of them would have the same semantic comprehension of what was informed [5]–[7]. Conceptual modeling is relevant on application design process, since conceptual specifications are used to support understanding, problem-solving and communication among stakeholders about a given subject domain. Based on this, an alternative is Context Metamodel [8], a domain independent context metamodel to support contextual model creation based on UML extensions.

This paper presents a context conceptual model focused on context information for a distributed software engineering environment. The context conceptual model presented here uses as conceptual base the context metamodel proposed in [8]. This model allows identifying the set of contextual information relevant to each environment entity, how these information are related among themselves and how context and environment behavior are related.

The rest of this paper is organized as follows: section II presents the Context Metamodel used to create context conceptual models, section III introduces the context conceptual model created for DiSEN, section IV presents a preliminary proof of concept, section V presents some related works and, finally, some conclusions and final considerations are presented on section VI.

## II. CONTEXT METAMODEL

After analyzing different existing context models, Vieira [8] checked that, in general, there is no formalism for

creating context models that associate contextual information and its usage. In this sense, existing models do not relate contextual information and its use. Based on that observations, Context Metamodel was proposed. Context Metamodel is a domain independent metamodel to support contextual model creation. This metamodel is an UML 2.0<sup>1</sup> extension.

Within Context Metamodel *Contextual Element* is defined as any piece of data or information that enables to characterize an entity in a domain; and *Context* of an interaction between an agent and an application, in order to execute some task, is the set of instantiated contextual elements necessary to support the task at hand. Other than that, contextual element is stable and can be defined at design time, while a context is dynamic, and must be constructed at runtime, when an interaction occurs.

Context metamodel is divided into two main packages that organize the concepts in two categories: structure, which describes the concepts related to the conceptual and structural elements of a context sensitive system (context structural model); and behavior, which contains the concepts related to the behavioral aspects of a context sensitive system (context behavior model).

Context structural model identifies contextual elements; the entities that contain them; relationships among them; and the way they are acquired. Context behavior model focuses on modeling how the system must react under a specific context, identifying the set of condition responsible for a system behavior variation. To support the design of the context sensitive system behavioral part, the context metamodel uses the concepts defined in the formalism of Contextual Graphs, defined in [9]. Contextual graphs are based on semantic networks, and support task models representation. They consider the relation between procedures established by an organization for executing a given task, and how contextual information influences this task execution. Each path on the contextual graph contains the rationale used to execute a task. It contains the sequence of triggered actions, conditions activated for each action, and context elements related to each condition. So, this kind of representation allows one to compose rules graphically, based on the paths described on the graph.

### III. MODELING CONTEXT IN DiSEN

#### A. DiSEN Environment

DiSEN (Distributed Software Engineering Environment) [3] is an environment aiming to provide an infrastructure to support distributed software development. DiSEN design is composed of three main Managers : (i) Object Manager – comprises all environment entities, involving Artifacts,

Processes, Resources, Projects, Tools, Users and Local Managers; (ii) Agent Manager – responsible for creating, registering, finding, migrating and destroying software agents; and (iii) Workspace Manager – provides shared workspaces, corresponding to environment instances where users can cooperate during tasks execution. DiSEN Object Manager has a Resource Manager responsible for managing physical resources (machines, paper, media) and also Places, Users and Tools. These three entities are responsible to provide contextual information to the environment. Users (a person or a group) are the most active entities within the environment capable to change states and generate information for the whole environment.

#### B. DiSEN-CSE

To allow contextual information sharing among several workspaces involved on a cooperative work, DiSEN-CSE (DiSEN-Context Sensitive Environment) was proposed. To reach this goal, every time a contextual information is generated by an Object Manager entity (Place, User or Tool) at any workspace within DiSEN, this information is captured by workspace manager. Within workspace manager, DiSEN-CSE transforms them into another information that can be understood by all team members geographically distributed, using a representation model based on a domain ontology. Then, DiSEN-CSE shares it with other workspaces, including the one that generated it.

DiSEN-CSE model has four essential elements: Capture Support, Context Representation, Processing Support and Awareness Mechanisms. The model also has a repository responsible for storing context information. Capture Support is responsible for recognizing context changes occurred during task execution. These changes can be captured by human agents or software agents. Context Representation is responsible for mapping information coming from Capture Support to a formal representation model – based on ontologies – and relates them to other information available within the repository. Processing Support corresponds to reasoning mechanisms, able to infer contextual information implicit on that captured beforehand, and fix possible inconsistencies, based on existing relationship among sets of information created by Context Representation. Finally, Awareness Mechanisms are responsible for identifying (automatically) what are the workspaces interested on the information, the methods that can be used to show them (defined by Pozza [2]) and for spreading them to workspace instances, using communication infrastructure.

So, DiSEN-CSE goal is to capture, manipulate and share contextual information available within DiSEN environment. However, to reach this goal, it is necessary to know what are contextual information available on the environment and what are the relationships among them. To achieve this, we propose a Context Conceptual Model, which is described in the next section.

<sup>1</sup>Unified Modeling Language: Superstructure, Version 2.1.2, In: <http://www.omg.org/spec/UML/2.1.2/>

### C. The DiSEN Context Model

According to [8] context is composed by elements related to an entity that is useful when supporting a problem resolution. To identify the elements with this feature within DiSEN domain, a conceptual model was created using Context Metamodel to specify main environment functionalities, defining contextual elements and the relationships among them, and the possible environment actions.

Vieira [8] says that the first step when building a context model using the context metamodel is to identify the main *foci* to be considered in the system. *Focus* is what enables to determine which Contextual Elements should be instantiated and used to compose the context. Brezillon [10] considers it to be a step in a task execution, in a problem solving, or in a decision making. In the context metamodel, *focus* is determined by the association between a task (*Task*) and the agent who is executing it (*Agent*).

All *foci* considered are depicted on Fig. 1 and are determined by DiSEN functionalities. In this paper we will use the *focus* (User, Acquiring Knowledge) to illustrate context modeling, chosen because it is an easy to understand *focus* and has all elements needed to explain the model.

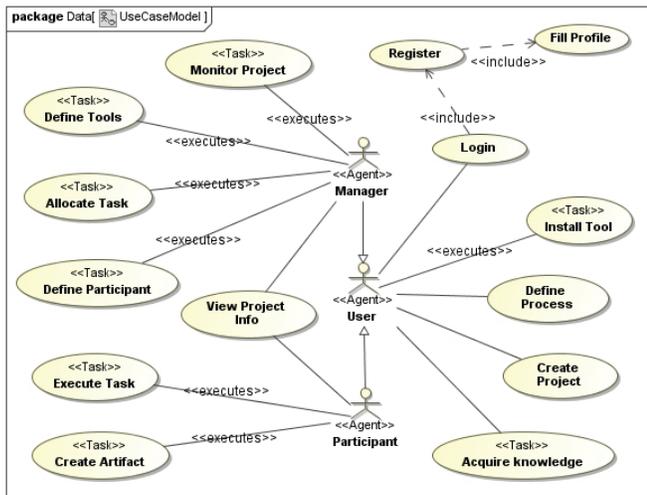


Figure 1. Use Case Model

1) *Building the Context Structural Model:* Inside DiSEN environment, an user can acquire knowledge in two different situations: when working on a project or when attending to a training. So, the following behavior variations were verified for the focus Acquiring Knowledge:

1. Users have knowledge. They can acquire knowledge when attending to trainings and when working on a project.

2. When an user attends to a training, the skillset acquired on that training and the level are defined. If an user attends to a basic level training, acquired knowledge is basic. If the training is an intermediate level, the skill offered is

intermediate. And if the training is advanced, the skill level offered is advanced.

3. When an user participates on a project, he/she may not have all required skills for that project. But, at the end of the project he/she may have acquired some of them. Users must inform to the system all new skills acquired.

Based on these variations, it is possible to verify that the elements that interfere on *focus* context are: User.doATraining, UserTraining.level, Participant.isOnProject, User.hasKnowledge, project.Status, Project.requireKnowledge, UserKnowledge.level. These elements represent contextual elements for this *focus* and are depicted on Contextual Structural Model of Fig. 2. Each context element is tagged with *ContextualElement* stereotype. An entity with at least one contextual element is a contextual entity, tagged with *ContextualEntity* stereotype. In this sense, the behavior that indicates user knowledge is represented by the contextual element User.hasKnowledge. The behavior that shows that users can attend to a training is specified by the contextual element User.doATraining and so on.

Context Structural Model also represents the way the information is acquired. Each contextual element has its own acquisition properties, indicating where the information can be found, the type of acquisition and the frequency the information is updated. Optionally, it can have a formation expression, showing a new rule to transform an obtained information to the expected format. These properties are represented by an acquisition relationship among a contextual entity (contextual element container) and the context source (the one which indicates the information source).

Fig. 2 depicts acquisition relationships. PersistenceService and InterfaceGUI entities are tagged with *ContextualSource* stereotype, representing context sources. For each contextual element there is a relationship with its own source, indicating acquisition configuration. These relationships are tagged with *acquisitionAssociation* stereotype.

Context structural model presents the system structure for a given *focus*, showing up contextual entities and elements, relationships among them and their acquisition way. However, when it is desired to design a Context Sensitive System, it is necessary to know system behavior, determined by the context. In this sense, behavioral model was created. It consists on contextual graphs able to determine possible behavior variation for each context. Context behavior model and some rules will be detailed on the following sub-section.

2) *Building the Context Behavior Model:* The contextual graph that defines the behavior of Acquiring Knowledge *focus* is depicted on Fig. 3. Behavioral variation is represented as a decision node, tagged with *ContextualNode*. When an action occurs (tagged with *Action* stereotype), it reaches a *ContextualNode*, and the environment must decide, according to the context, what is

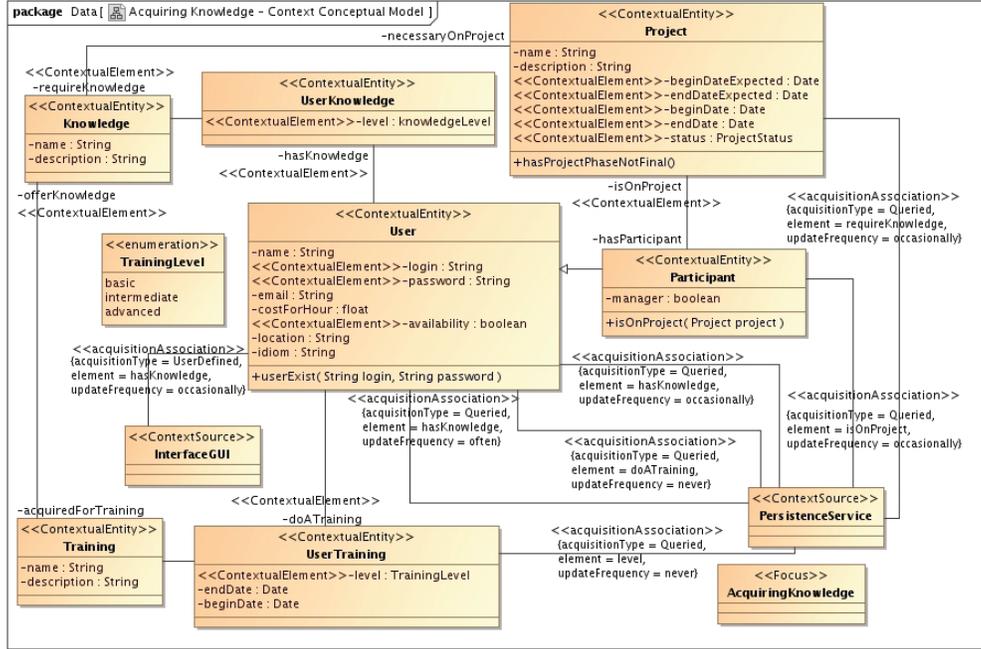


Figure 2. Context Structural Model – Focus: Acquiring Knowledge

the path to be followed. Possible paths are tagged with  $\langle\langle ContextualBranch \rangle\rangle$  stereotype. At the end, after performing all actions related to chosen path, there is a  $\langle\langle RecombinationNode \rangle\rangle$ , responsible to ensure graph consistency. Recombination node indicates that all actions related to that context were concluded, regardless what is the chosen branch.

The paths depicted by the graph can be converted to compose the rules in the context conceptual model. Some examples for Acquiring Knowledge *focus* are depicted on the right side of Fig. 3. The rules presented indicate necessary conditions to allow a given user to acquire a new knowledge.

Structural and behavioral context models made it possible to determine which information influences the context and how the changes in the interactions affect environment behavior for each focus. Section IV describes how the context conceptual model was implemented to support information sharing within DiSEN environment.

#### IV. PROOF OF CONCEPT

The context conceptual model presented here was implemented on DiSEN environment, during the development of an event notification manager called DiSEN-Notifier. DiSEN-Notifier is a responsible by receiving the information captured (Capture Support), represented (Context Representation) and processed (Processing Support) and deciding who should be notified about the events occurred. In addition, this manager prepares a message and sends it to any available Awareness Mechanism, which notifies the users in an appropriate way.

The decision on who should receive a notification at a given time is based on context behavioral models. Thus, DiSEN-Notifier becomes context-sensitive as it implements the conceptual model presented in this paper and can, based on the model, find out what people are interested on a specific focus and therefore should be notified on events occurred. Moreover, as behavioral patterns define the context under which the notifications should be sent, when those predefined circumstances occur, the system triggers the notifications automatically, without any action performed by the user requesting to send or to check for new incoming information. More information about the DiSEN-Notifier and how to use the conceptual modeling of context for reporting information can be found in [11].

In addition, an ongoing research is using the context conceptual model defined here to support processing rules definition for Context Representation element on DiSEN-CSE. These rules will be used to make inferences from information captured, based on an ontological model. The results of this research will be presented in future work.

#### V. RELATED WORK

Many proposals had been made in order to allow context modeling. However, most of these proposals focus on ubiquitous or pervasive environments [7], or collaborative applications in general, so far there are few studies related to context modeling for Global Software Development. This section presents some researches related to this paper.

Regarding collaborative applications, Rosa et. al [12] proposed a conceptual framework to identify and classify

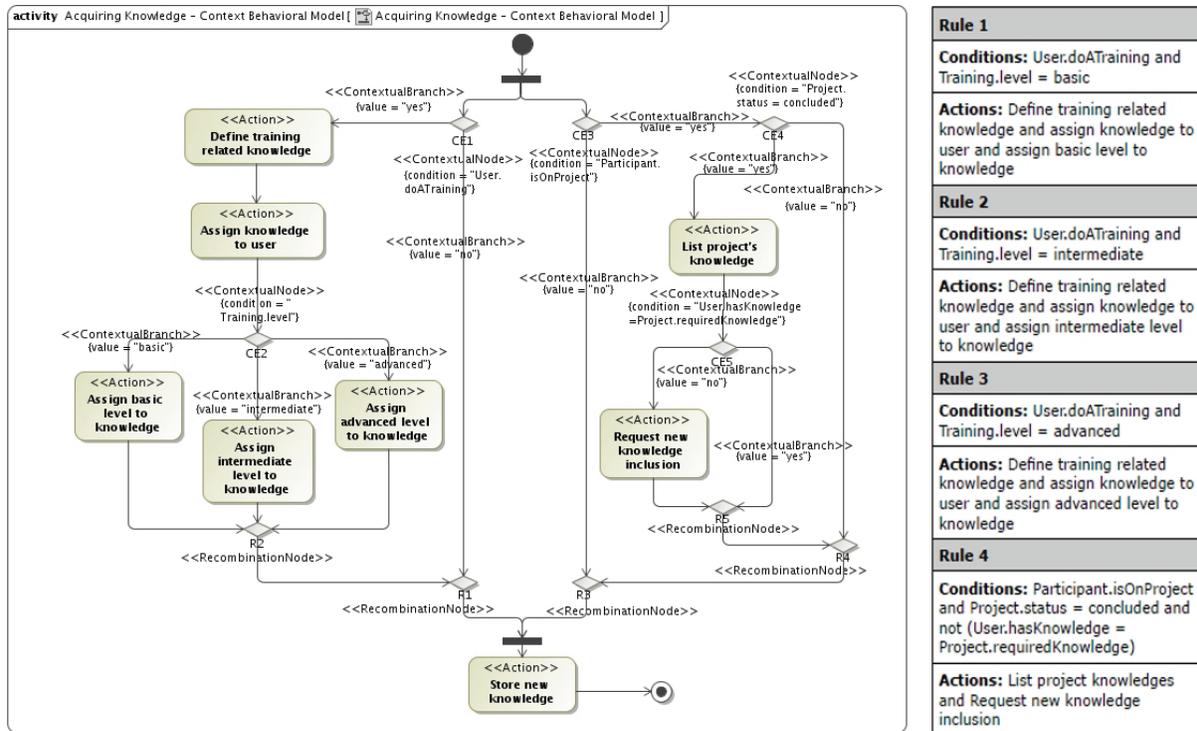


Figure 3. Context Behavior Model and some rules extracted – Focus: Acquiring Knowledge

most common contextual information on groupware tools. This framework classifies the information into five types: group members, tasks, relationship among people and tasks, setting and completed tasks. Based on these information, this framework keeps important information for groupware users be aware of the influence of their interactions. Vieira et. al [13] propose an ontology to formally represent context in groupware systems. The idea is to identify which information presented in groupware applications could be classified as context and which kinds of context should be represented. Nunes et al. [14] presented a model for managing context-based knowledge, which addresses the creation, storage and reuse of contextual knowledge, encompassing the representation, capture, storing, comparison, and presentation of knowledge in the setting where the work process activity is performed. This knowledge management model aims to establish organizational memory, including the results of performing the work process activity and the context through which results were produced. However, none of these studies bring integrated solutions to support context structural and behavioral aspects, nor are related to GSD.

Specifically for context modeling in GSD, Eckhard [15] developed a language called NSL (Notification Specification Language) to support NOTICON, a tool that allows context based notification. The NSL language used to define context was created specifically for NOTICON and requires a specialist to model context. For each focus a file containing

context specification is generated, making it difficult to reuse already modeled *foci*.

Veiel [16] defined a context model to adapt collaboration in shared workspaces. This context model addresses scenarios in which several actors collaborate to achieve a shared goal and thereby captures basic concepts of co-located and distributed collaboration. This model identifies classes and relationship needed to model configuration of shared workspaces and tools, and to capture the current context at runtime. Based on the relationships created at a given moment, adaptation rules capable to use context are defined to adapt user workspaces involved on the interaction. The rules that define the adaptive behavior are textually described, and do not follow any formalism or standardized language. Furthermore, behavior is described textually, without any model to support understanding and reuse.

Conceptual modeling, as presented on this paper, allows representing elements within a given application domain and their possible relationships. The use of conceptual modeling to identify contextual information on GSD domain brings as advantage the ability to represent structural (Context structural model) and behavioral (Context behavior model) viewpoints. In this sense, this work differs from the ones presented here by presenting a way to facilitate communication among geographically distributed individuals, considering dynamic context, differing context and contextual elements. The model also offers a standard way of context representa-

tion that allows model reuse and easy understanding, since UML is well established on industry and academy.

## VI. CONCLUSION

A well known problem caused by geographical distribution on GSD environment is the communication gap. This is mainly caused by mistrust among team members, lack of disposal to help people involved on the same task, cultural and organizational differences. Because of this, a context-awareness based model was proposed to share contextual information among several workspaces on a distributed software development environment, DiSEN. However, to offer contextual information it is necessary to know entities and what set of information about these entities are important to create the context of a given focus. In this sense, this paper presented a context conceptual model for DiSEN.

To create the context conceptual model context metamodel [8] was used. The use of Context metamodel improved model semantic comprehension and made easier to design context concepts in context sensitive environment DiSEN. The metamodel also supports dynamic context understanding, brings up the difference among context and contextual elements and presents how context can be used to model behavior variation on different domain areas. Acquiring Knowledge focus was used to demonstrate the context conceptual model produced. Thus, the main contribution of this paper is a context conceptual model for GSD, integrating structural and behavioral aspects by: (i) identifying which set of information will compose environment entities context; (ii) identifying how these information can be used to design system behavior based on the context; and (iii) facilitating domain semantics comprehension and reuse of context models. In addition, this model was used as a base to implement an event notification manager called DiSEN-Notifier.

There are some future work in progress related to this research: (i) evaluate context information transformation, and its behavior within DiSEN when considering context information present on conceptual model; (ii) link conceptual information identified by conceptual model and the ontology based representation model, to verify consistency for the considered domain; (iii) explore sharing mechanisms for captured contextual information; and (iv) use the context conceptual model presented here to support processing rules definition for Context Representation on DiSEN-CSE.

*Acknowledgments:* The author Ana Paula Chaves thanks CAPES for financial support.

The author Vaninha Vieira thanks the National Institute of Science and Technology for Software Engineering (INES), funded by CNPq grant 573964/2008-4, for their financial support.

## REFERENCES

- [1] M. Ivcek and T. Galinac, "Aspects of quality assurance in global software development organization," in *31st International Convention MIPRO*, 2008, pp. 150–155.
- [2] R. S. Pozza, "Proposta de um modelo para cooperacao baseado no gerenciador de workspace no ambiente disen," Master's thesis, PCC, UEM, 2005.
- [3] E. H. M. Huzita, T. F. C. Tait, T. E. Colanzi, and M. A. Quinaia, "Um ambiente de desenvolvimento distribuido de software – disen," in *I WDDS*, Joao Pessoa - PB, 2007, pp. 31–38.
- [4] A. P. Chaves, I. S. Wiese, C. A. da Silva, and E. H. M. Huzita, "Um modelo baseado em context-awareness para disseminacao de informaces em um ambiente de desenvolvimento distribuido de software," in *CLEI 2008*, Argentina, 2008.
- [5] T. Strang and C. Linnhoff-Popien, "A context modeling survey," in *Workshop on Advanced Context Modelling, Reasoning and Management, in 6th International Conference on Ubiquitous Computing*, Inglaterra, 2004, pp. 34–41.
- [6] T. Gu, X. H. Wang, H. K. Pung, and D. Q. Zhang, "An ontology-based context model in intelligent environments," in *Proceedings of CNDS*, USA, 2004, pp. 270–275.
- [7] M. Baldauf, S. Dustdar, and F. Rosenberg, "A survey on context-aware systems," in *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 2. Inderscience, 2007, pp. 263–277.
- [8] V. Vieira, "CEManTIKA: A domain-independent framework for designing context-sensitive systems," Ph.D. dissertation, CIn – UFPE, Recife, 2008.
- [9] P. Brezillon, "Task-realization models in contextual graphs," in *CONTEXT*, ser. Lecture Notes in Computer Science, vol. 3554. Springer, 2005, pp. 55–68.
- [10] P. Brezillon and J. C. Pomerol, "Contextual knowledge sharing and cooperation in intelligent assistant systems," *Le Travail Humain*, vol. 62, no. 3, pp. 223–246, 1999.
- [11] A. P. Chaves, E. Huzita, and V. Vieira, "Context-based notification supporting distributed software teams management and coordination," in *SBSC 2009*. IEEE SC, 2009, pp. 80–89.
- [12] M. Rosa, M. Borges, and F. Santoro, "A conceptual framework for analyzing the use of context in groupware," *Lecture Notes in Computer Science*, pp. 300–313, 2003.
- [13] V. Vieira, P. Tedesco, and A. Salgado, "Towards an ontology for context representation in groupware," vol. 3706. Springer, 2005, pp. 367–375.
- [14] V. T. Nunes, F. M. Santoro, and M. R. Borges, "A context-based model for knowledge management embodied in work processes," *Information Sciences*, pp. 2538 – 2554, 2009.
- [15] B. Eckhard, "Context-aware notification in global software development," Master's thesis, Institut fr Softwaretechnik und interaktive Systeme – Technischen Universitt Wien, 2007.
- [16] D. Veiel, J. Haake, and S. Lukosch, "Extending a shared workspace environment with context-based adaptations," in *CRIWG*, ser. Lecture Notes in Computer Science. Springer, 2009, pp. 174–181.

# Service Automation Architecture as adopted by Unified Communication Audit Tool

Shadan Saniepour Esfahani  
Cisco Systems  
Advanced Services  
Unified Communications Practice  
ssaniepo@cisco.com

Talal Siddiqui  
Cisco Systems  
Advanced Services  
Unified Communications Practice  
tsiddiqu@cisco.com

**Abstract-** This article elaborates on the architecture developed for the Unified Communications Audit Tool (UCAT) to eliminate manual processes for the Cisco Unified Communication audit service, and automate the audits to the possible extent. The rule-based system built into this architecture, allows conducting intelligent analysis to produce a comprehensive report. The modular nature of the architecture makes the system very flexible and allows distributing the collection agent piece freely to the customers, while keeping the analysis and report generation component, which is the intelligent portion, internal. The tracking feature provided in this architecture, allows providing realistic statistics on distribution of Cisco products in the customer networks. The statistics gathered in the tracking database is used for trend analysis of deployments, architectures, issues, and to generate alerts in very customer specific context. This tracking information also allows the users of this database to proactively inform customers of any critical information that can be pertinent to them, and hence increase the customer satisfaction. Since this is a web-based architecture, services can be launched remotely, eliminating the need to be physically at the customers' location to conduct the audit. This article presents a few use cases for UCAT and provides the cost savings figures.

## I. INTRODUCTION

From an enterprise point of view, there are two main challenges with delivering services, namely; the quality and the cost. As an enterprise grows in size and becomes a global entity, these challenges become even more accentuated.

Services' computing is a new research field that goes beyond traditional computing disciplines as it includes not only architectural, programming, deployment, and other engineering issues, but also management issues such as business component modeling, business process design, and service delivery [1]. Web services are gaining more popularity [2][3][4][5][6], and Web 2.0 is becoming a typical element of the common workplace. These are all attempts to cut the costs and increase the quality.

In a global organization, delivering a uniform quality of service to customers all around the world is not trivial. Even with a predefined process and framework, depending on the skill set and expertise in different locations, the quality of service might vary. Automating

the services can help provide a uniform quality of service, by leveraging a single knowledge base [7][8]. While the local staffs are required to customize the deliverables in the form of a report, an automated service would help promote the quality, because it can:

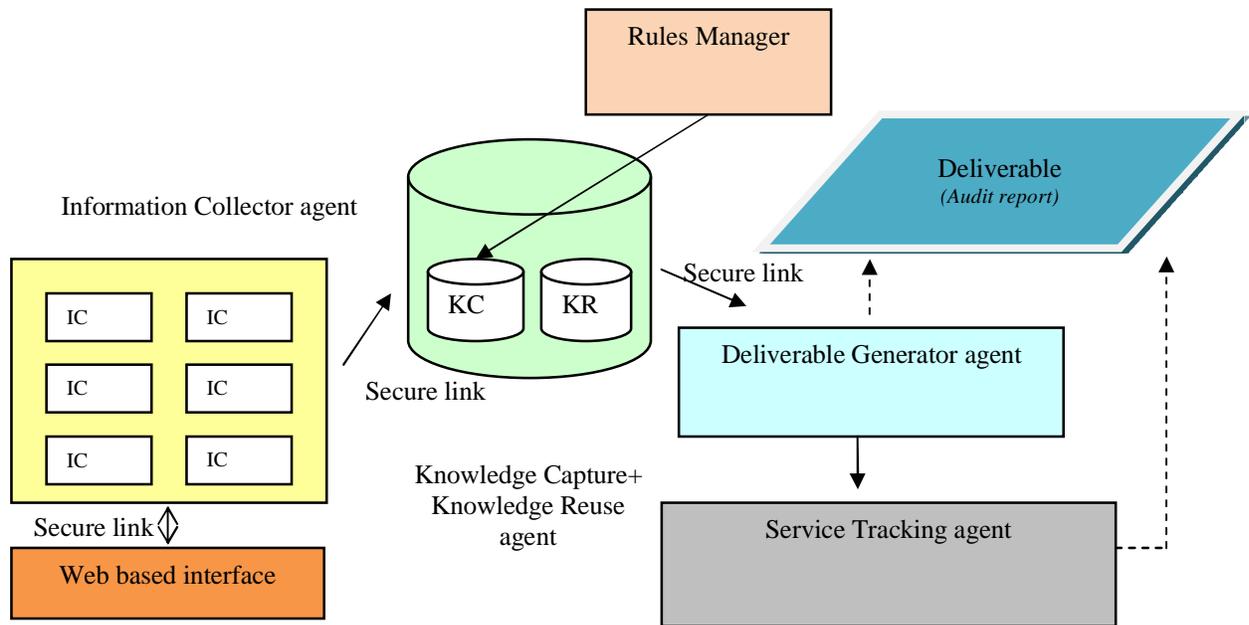
- 1) Increase the accuracy of the deliverable, by eliminating human errors
- 2) Reuse the previously captured knowledge from a global team of experts, by utilizing a common knowledge base
- 3) Save time and effort by way of automation
- 4) Help deliver a standard high quality service to customers all around the globe

To this end we have been developing tools that can automate our services. In this article we introduce the architecture that has been adopted by the Unified Communications Audit Tool (UCAT) designed to audit enterprise's Unified Communication applications including IP telephony, collaboration applications, and contact center environment and the underlying infrastructure. We elaborate on the internals and operations, and study this architecture in practice.

## II. SYSTEM ARCHITECTURE

The main components of the system are defined as: Information Collector agent, Knowledge Capture agent, Knowledge Reuse agent, Deliverable Generator agent, Service Tracker agent. Fig. 1 illustrates this architecture.

The Information Collector agent gathers all the information required for the service. For the Unified Communications Audit Tool (UCAT) this information is typically configuration and inventory data, but can also be network performance attributes such as latency or jitter. A multi-threaded architecture is applied to this agent, to increase the performance, by spawning multiple threads that collect different information simultaneously. The Information Collector agent communicates with a web based user interface, allowing access to the system remotely, via World Wide Web. This eliminates the need for engineers to be physically in the customers' location to provide the service.



**Fig. 1. Architecture**

The Knowledge Capture agent captures all the intelligence that is required for the service. In the context of Unified Communications Audit Tool (UCAT) this refers to the best practice rules applied to UC networks. This knowledge base is constantly updated with the experience that our experts gain from the field. The Rules Manager feeds new rules to the Knowledge Capture agent.

The Knowledge Reuse agent applies this intelligence (rules) to customers’ deliverables. In the context of UCAT this intelligence is referred to rules that are applied over the audit report, and any deviation from these rules in the report is highlighted.

The Deliverable Generator agent generates the final audit report to be presented to the customer. The final deliverable of UCAT is a MS word document that lists all the configuration and inventory data and makes recommendation based on UC and network design best practices based on collective experience of Cisco engineers.

The Service Tracking agent keeps track of information that pertains to customers who receive the service. Hence, the tracking server is able to identify if at a later time, after the final deliverable is submitted, there is any new information that can be of the interest of the customers who receive the service. In the context of Unified Communications Audit Tool (UCAT) this can be hardware or software end of life/end of support information or PSIRTs (Product Security Incident Responses).

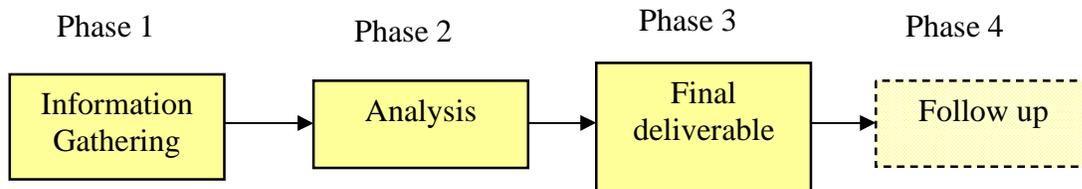
This architecture has proven to be very efficient. The modular design of this architecture provides flexibility, and allows breaking the tasks into simpler ones for independent but coordinated execution. It also keeps the intelligent module separated from others, hence enables us to distribute other modules to the customers freely, while keeping the intellectual property internal. Finally, the Tracking agent allows the system to identify any information that might be of interest to each customer, even after submitting the final deliverable.

### III. SYSTEM OPERATIONS

As Fig. 2 illustrates, we have divided the tasks required for conducting a service into a few phases: information gathering, analysis, and final deliverable.

In the first phase, the information required to perform the task is collected. The collected data is then passed on to be analyzed. In the analysis phase the intelligence is applied over the collected data and the final results are produced. The next phase prepares the result to be delivered to the customer as the final deliverable, and the final phase is to follow up with customers if there is any information that might be of interest to them. Automation can be done in any of these phases to increase accuracy and promote efficiency.

In the next sections we describe how this architecture is applied to Unified Communications Audit Tool and explain the benefits associated with the tool.



**Fig. 2. Operations**

#### IV. UNIFIED COMMUNICATION AUDIT TOOL

Unified Communications Audit Tool (UCAT) provides means for auditing Unified Communication (UC) network components (e.g., Cisco Communications Manager, Unity, Cisco Unified Contact Center, Gateways, Switches, etc) in a very efficient manner. This tool collects configuration and inventory data from UC network, as the input, and provides an audit report as the output. The report contains the collected data, displayed in a very comprehensive manner. The tool also compares the collected information against Cisco Advanced Services (AS) best practices, and highlights the report if it finds any deviation from the best practices. Fig. 3 demonstrates UCAT overview.

The overwhelming amount of data collected in the data collection phase will take a few weeks, if gathered manually. The tool collects all the data remotely using different methods, from SNMP, and WMI, to web based AXL/SOAP, hence eliminating the need to manually log on to the devices one by one. Table 1 lists all the data collection methods used by UCAT for different UC platforms. The intelligence embedded to the data collector allows the tool to switch to alternate collection methods wherever needed, and automatically detect the best method of data collection, hence making the data collector very effective and accurate. The collected data is in XML format and is saved in an XML database. The collected data is encrypted and compressed for secure transport over Internet or VPN from customer's network back to a central location where the intelligent analysis and report generating modules reside.

**Table 1 Data collection Methods**

UC Platform	Data Collection Method
Cisco Communications Manager (CUCM)	AXL/SOAP, SNMP, SQL, CLI
Unity	SQL, Registry, GUSI, remote execution of commands (NSLookup, DNSlint, Netstat, etc)
Cisco Unified Contact Center (CUCC)	WMI, SQL, Registry, remote execution of commands (NSLookup, MSinfo, etc)
Network Infrastructure (Gateways and Switches)	SNMP, CLI (Telnet, SSH)

UCAT is a web-based application, allowing users to access the web interface to launch the service. Therefore it eliminates the need for going to customers' location to provide the service. Typically the data collector piece of the application is freely distributed to the customers. Once the tool is installed at the customers' premises, users can access it via World Wide Web to launch the service and collect the data.

A database of best practice rules makes the tool intelligent enough to highlight any part of the report that does not fully comply with the Cisco design best practices. The best practices rules cover design rules for network resiliency, high availability, scalability, and security. This database is constantly being updated with the experience gathered from the field. Rules are applied to the XML data using xqueries. The new rules are reviewed by subject matter experts for final analysis and fed to the tool using the rules manager. UCAT also provides an interface for users to customize the rules, if needed. Although manual work is required to customize the final deliverable for every customer, the accuracy of this database guarantees to highlight every potential issue in the customers' network.

The Report (Deliverable) Generator component of UCAT is the module that provides the final audit report in MS Word format.

The audit information is also securely archived on a tracking database, and is used at a later time, to proactively inform the customers about relevant information such as security advisory notification, product end of life (EoL), end of service (EoS), to their network. This information also allows further data analysis for determining the distribution of products in the customers' network or identifying different architectures and combination of UC and network products adopted by different customers.

UCAT provides a web based, user interface, to access the data collector remotely. Typically the data collector is installed on the customers' premises, but users can access it from anywhere using a browser. The information exchange facilitated by this tool is very secure. All the communications between the browser and the data collector are encrypted, using HTTPS. The

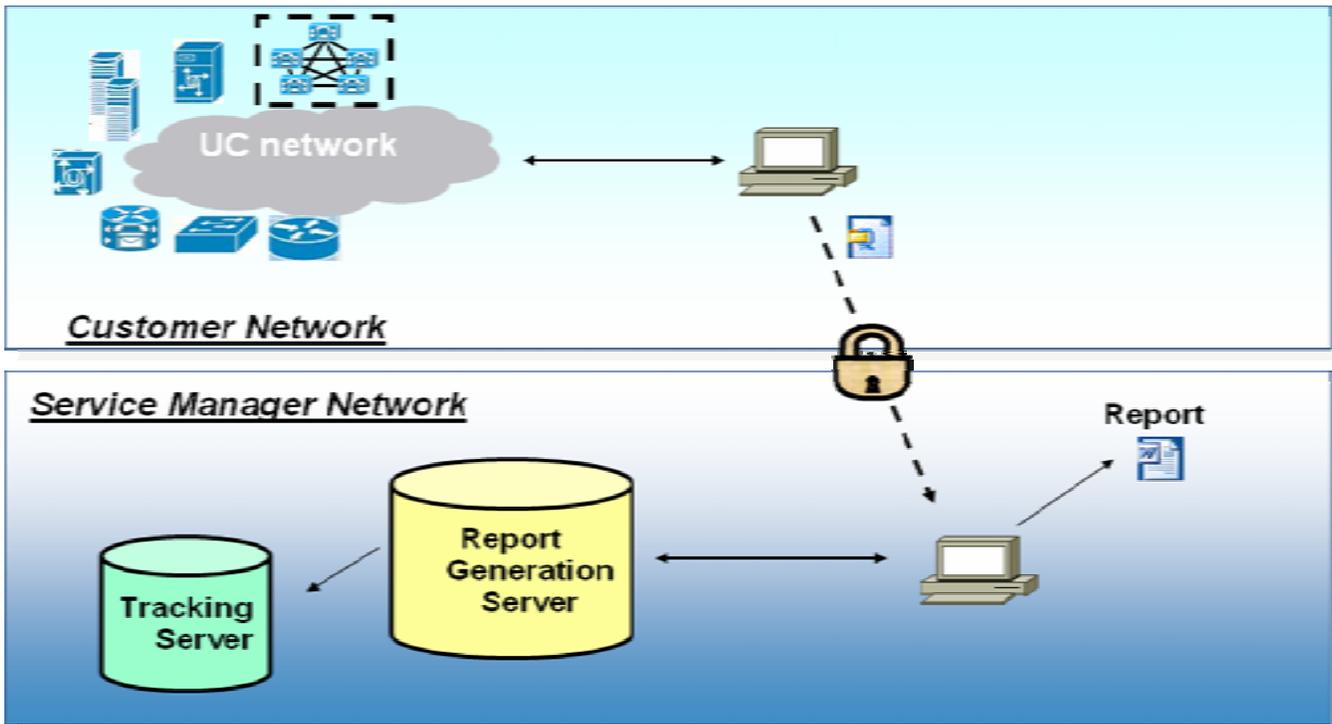


Fig. 3 Unified Communication Audit Tool (UCAT)

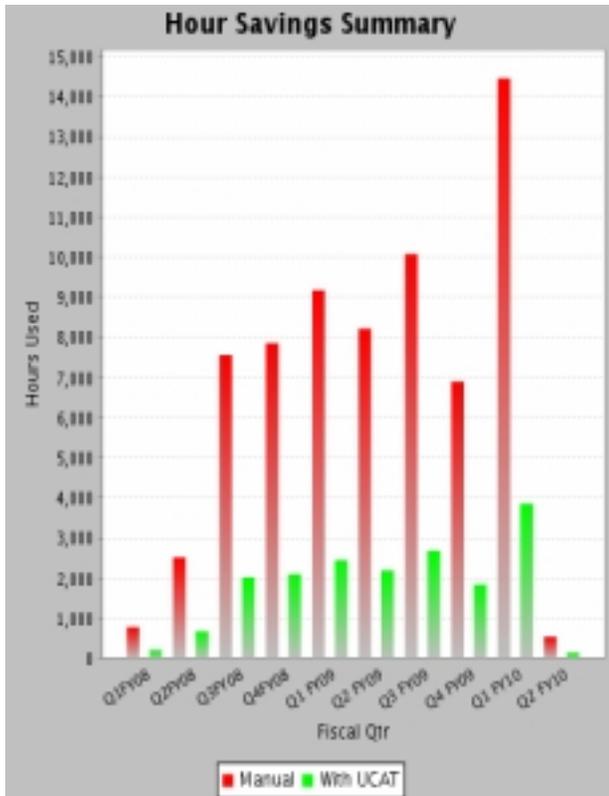


Fig. 4 Hour savings by UCAT

credentials provided to the tool are also encrypted and they will be removed after data collection is completed.

Referring to the Fig. 2, all four phases are automated in UCAT. There are two remarkable advantages with using UCAT: 1) accuracy of the report,

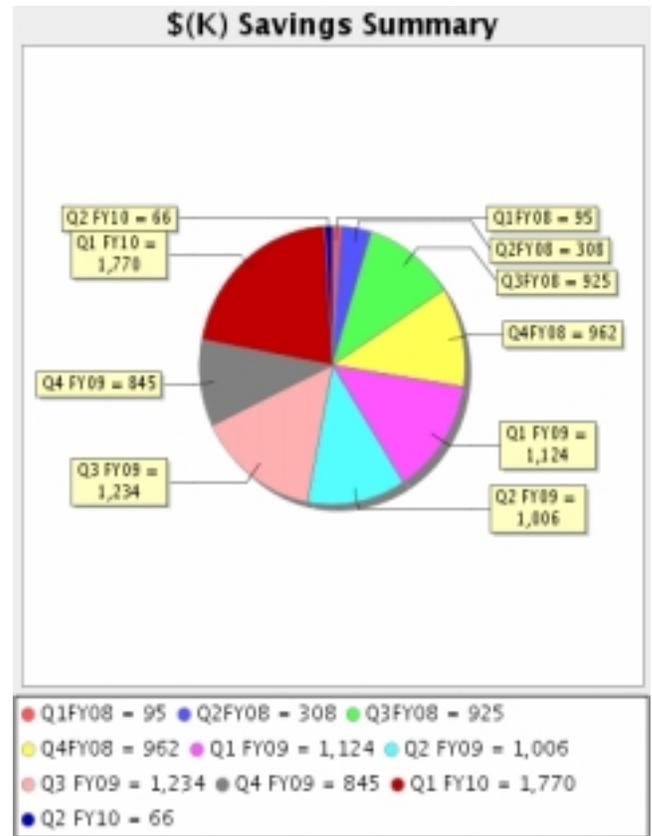


Fig. 5 Costs Savings by UCAT

by eliminating human misses and automating the data collection process, and 2) cost saving in terms of cutting the manual work significantly. Fig. 4 shows the hour

savings and Fig. 5 shows the cost savings reported for UCAT.

## V. USE CASES

UCAT has number of different use cases. Some of these use cases include (but not limited to):

- 1- Troubleshooting: since the UCAT report identifies any deviation from best practices, reviewing the report would tell users, in a glance, the areas that may potentially be causing problems.
- 2- Optimization: the recommendations, based on the best practices in the UCAT report, are very good bases for identifying areas of enhancement.
- 3- Migration: UCAT provides a comprehensive list of information, such as node license information. This information can be taken into account before the migration, to be considered in the planning stage.
- 4- UCAT tracking server has a number of use cases as well, some of which were mentioned in the previous section. Examples of UCAT tracking server include, but not limited to: proactively inform customers of information pertinent to them, find the most popular products in the field, find the most commonly adopted combination of products, etc.

In addition, UCAT offers all its features through web services. The web service API is readily available to Cisco AS partners. A third party application can use this web service and build a customized service around it.

There are ongoing development efforts on the UCAT tool to support additional features. One of the features that is going to be on the roadmap is support of "UC Green" feature, where the UCAT report would provide recommendations on how customers can tune their network by, for example, shutting down redundant phones, to help the green and help with cost savings.

## VI. SUMMARY

This article explains how Unified Communication Audit Tool (UCAT) is designed and used to automate the Cisco Unified Communication Audit process. The architecture of UCAT is described in detail. The merits of this architecture which include flexibility, modularity, embedded intelligence, and ability to track information are also discussed. Some use cases of the tool are listed, and the cost saving figures are presented.

## References

- 1) J. Leon Zhao, Mohan Tanniru and Liang-Jie Zhang, Services computing as the foundation of enterprise agility: Overview of recent advances and introduction to the special issue, Information Systems Frontiers Journal, Volume 9, Number 1 / March, 2007
- 2) Therani Madhusudan<sup>1</sup>, A web services framework for distributed model management,

- Information Systems Frontiers Journal, Volume 9, Number 1 / March, 2007
- 3) Seung-Hyun Rhee, Hyerim Bae and Yongsun Choi, Enhancing the efficiency of supply chain processes through web services, Information Systems Frontiers Journal, Volume 9, Number 1 / March, 2007
- 4) A. Dogac, Y. Kabak, G. Laleci, S. Sinir, A. Yildiz, S. Kirbas, Yavuz Gurcan, Semantically enriched web services for the travel industry, SIGMOD Record, Vol. 33, No. 3, September 2004
- 5) Yao-Min Fang, Li-Yu Lin, Chua-Huang Huang and Tien-Yin Chou, An integrated information system for real estate agency-based on service-oriented architecture, Expert Systems with Applications, Volume 36, Issue 8, October 2009, Pages 11039-11044
- 6) Sudeep Mallick, Anuj Sharma, B. V. Kumar and S. V. Subrahmanya, Web services in the retail industry, Sadhana Journal, Springer India, in co-publication with Indian Academy of Sciences, Volume 30, Numbers 2-3 / April, 2005
- 7) C.F. Cheung, Y.L. Chan, S.K. Kwok, W.B. Lee, W.M. Wang, knowledge-based service automation system for service logistics, Journal of Manufacturing Technology Management, Volume 17, Issue 6, 2006, Pages 750-771, ISSN: 1741-038X
- 8) C. F. Cheung, W. B. Lee, W. M. Wang, K. F. Chu and S., A multi-perspective knowledge-based system for customer service management, Expert Systems with Applications, Volume 24, Issue 4, May 2003, Pages 457-470

# Knowledge Based Service Oriented Architecture for M&A

Debasis Chanda<sup>1</sup>, Dwijesh Dutta Majumder<sup>2</sup>, Swapan Bhattacharya<sup>3</sup>

<sup>1</sup> Jadavpur University, Kolkata 700032, India

<sup>2</sup> Professor Emeritus, Electronics & Communications Sciences Unit, Indian Statistical Institute, Kolkata,  
& Director, Institute of Cybernetics Systems & Information Technology, 203 BT Road, Kolkata 700035, India

<sup>3</sup> Director, National Institute of Technology, Durgapur 713209, West Bengal, India & Professor, Department of Computer Science & Engineering, Jadavpur University, Kolkata 700032, India

E-mail: cdebasis04@yahoo.co.in, ddmdr@hotmail.com,  
director@nitdgp.ac.in

## ABSTRACT

For creating virtual enterprise (VE), which is in general the collaborative partnership between business partners in value chains, the business processes (services) of the resulting organization need to be composed of the individual business processes of the participating organizations. This can be realized by means of Service Oriented Architecture (SOA). In our paper we

propose a SOA Framework based on Knowledge Bases.

**Keywords:** Knowledge Base, Predicate Calculus, Service Oriented Architecture, Merger & Acquisition

## 1 INTRODUCTION

The goal of this paper is to present a new service oriented modeling framework for the virtual enterprise (VE), which is focused on process composition. This framework uses Predicate Calculus Knowledge Bases. The rest of this paper is organized as follows: In Section 2 we furnish literature review of articles published in the relevant areas. In Section 3, we introduce our modeling framework. Finally, Section 4 provides some conclusions.

## 2 RELATED WORK

A typical service-oriented architecture (SOA) has three main parts: a provider, a consumer and a registry [1]. Our approach proposes a Knowledge Base which is a repository of processes / services; discovery is affected by pattern search & unification/substitution.

The paper [2] suggests a framework for designing agile and interoperable VEs, which supports enterprise modeling. Our Paper proposes a framework/architecture for consolidation of business services that may be adopted on enterprise-wide basis.

Claus Pahl's paper [3] presents ontology-based transformation and reasoning techniques for layered semantic service architecture modeling. We propose a Knowledge Based framework which considers reasoning.

Arroyo et al. [4] describe the practical application of a semantic web service-based choreography framework; we propose a consolidation / choreography framework

that adopts Knowledge Based approach for Banking System.

Jung et al [5] propose an architecture for integrating knowledge management systems (KMSs) and business process management systems (BPMSs) to combine the advantages of the two paradigms. We propose a Knowledge Based framework that supports the capture of business process knowledge.

Rezgui [6] argues that a role based authorization approach to service invocation is necessary in order to enhance and guarantee the integrity of the transactions that take place in the business environment of a VE. Our framework takes into account authentication, and can be extended for role based authorization.

Luger [7] captures the essence of artificial intelligence. We apply artificial intelligence concepts for developing knowledge bases towards modeling business processes in our work.

## 3 KNOWLEDGE BASED SERVICE ORIENTED MODELING FRAMEWORK

For our purpose, the representative domain of discourse considered is the deposit function of two banks, for which the knowledge can be represented as a set of Predicate Calculus expressions.

The realization of each business process for the consolidated bank (following the merger of the two representative banks) requires the orchestration/composition of the business processes for each of the two banks into

consolidated business process. The orchestration / composition of the business processes is achieved by means of our proposed approach.

### 3.1 CONSOLIDATED SCENARIO: KNOWLEDGE BASED PROCESS COMPOSITION

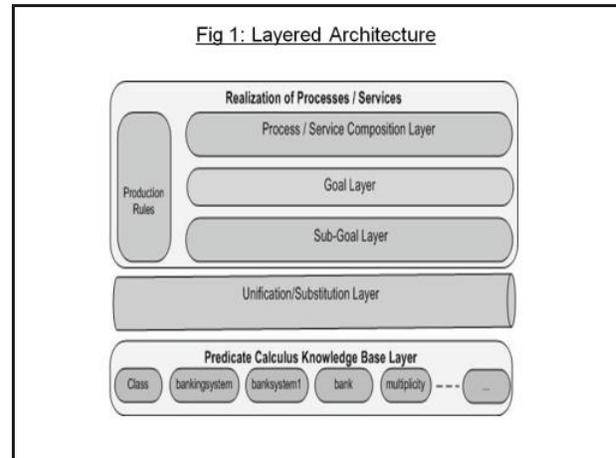
The goal for the consolidated scenario is represented by:  $\text{displaybalance}(\text{bank1}) \vee \text{displaybalance}(\text{bank2}) \rightarrow \text{displaybalance}(\text{bank\_newbank})$ , where newbank is the bank, post consolidation. The goal gives a valid result if the customer of the new bank is either a customer of Bank 1 OR if he is a customer of bank 2, but he is simultaneously not a customer of both the banks.

Now if the customer is having an account in both the banks (Bank1 & Bank2), the above goal representation to evaluate to a business fault condition. However, if we have the following goal representation for the ‘display balance’ service, the fault condition would not occur:  $\text{displaybalance}(\text{bank1}) \wedge \text{displaybalance}(\text{bank2}) \rightarrow \text{displaybalance}(\text{bank\_newbank})$

The new service  $\text{displaybalance}(\text{bank\_newbank})$  is an orchestration of the two services  $\text{displaybalance}(\text{bank1})$  &  $\text{displaybalance}(\text{bank2})$ . Atomic services as well as composite services (which are arrived at by orchestrating atomic/composite services) can be well realized by means of Goals & Sub goals.

### 3.2 LAYERED ARCHITECTURE

The Layered Architecture diagram for our proposed Knowledge Based Architecture has been furnished in Fig 1.



The architecture comprises of three horizontal layers viz.

- Knowledge Base Layer (lowermost layer): comprises of the Predicate Calculus expressions
- Unification/Substitution Layer (middle layer): comprises of the set of unifications that satisfies each sub goal
- Realization of Processes / Services layer (upper layer)

The upper layer (Realization of Processes / Services layer) in turn comprises of the following layers

- Horizontal Layers
  - Sub-Goal Layer: comprises of the various sub-goals

- Goal Layer: comprises of the inferences
- Process / Service Composition Layer: comprises of consolidated services
- Vertical Layer
  - Production Rules Layer: comprises of rule based expert system

## 4 CONCLUSION

The unique contributions of this paper are the following:

- To present a new Knowledge based process modeling framework for the VE that is an alternative approach to prevalent UML & Petri Net based modeling approaches. Though Knowledge Based approach has been used in many domains, the novelty is in the new domain of service oriented architecture (SOA)
- Our approach provides benefits (comprehensive modeling of static structure dynamic features of a system) & additional architectural features over existing approaches and thus advances the state of the art

## References

- [1] Marco Crasso, Alejandro Zunino, Marcelo Campo, Easy web service discovery: A query-by-example approach, *Science of Computer Programming* 71 (2008) 144–164
- [2] Tae-Young Kim, Sunjae Lee, Kwangsoo Kim, Cheol-Han Kim, A modeling framework for agile and interoperable virtual enterprises, *Computers in Industry* 57 (2006) 204–217
- [3] Claus Pahl, Semantic model-driven architecting of service-based software systems, *Information and Software Technology* 49 (2007) 838–850
- [4] Sinuhe Arroyo, Miguel-Angel Sicilia, Juan-Manuel Dodero, Choreography frameworks for business integration: Addressing heterogeneous semantics, *Computers in Industry* 58 (2007) 487–503
- [5] Jisoo Jung, Injun Choi, Minseok Song, An integration architecture for knowledge management systems and business process management systems, *Computers in Industry* 58 (2007) 21–34
- [6] Y. Rezgui, Role-based service-oriented implementation of a virtual enterprise: A case study in the construction sector, *Computers in Industry* 58 (2007) 74–86
- [7] George F Luger, *AI Structures and Strategies for Complex Problem Solving*, Pearson Education, Fourth Edition, 2006

# ISE – Integrated Service Engineering: Applying an Architecture for Model to Model Transformations

Hao Hu

Siemens AG - Corporate Technology  
Information & Automation Technology - Knowledge Management  
Otto-Hahn-Ring 6, 81739 Munich, Germany  
Email: hao.hu@siemens.com

Gregor Scheithauer, and Guido Wirtz

University of Bamberg  
Distributed Systems Group  
Feldkirchenstraße 21, 96052 Bamberg, Germany  
Email: scheithauer@acm.org, guido.wirtz@uni-bamberg.de

**Abstract**—The Integrated Service Engineering (ISE) framework supports planning, designing and implementing tradable business services. For doing so, it utilizes various models with different intentions on multiple levels of abstraction. Consequently, consistency and synchronization between these models is an issue of high importance. Model-Driven Architecture (MDA) describes a software design approach that utilizes model-to-model and model-to-code transformations to remedy such issues and in turn results in improved software in terms of portability, productivity, and quality. This paper introduces the ISE framework and a generic MDA architecture that is used to show how to apply MDA successfully to the service engineering domain. Additionally, the paper investigates currently available model transformation technology in order to realize the proposed architecture. Lastly, a data transformation example from a real-world scenario is presented to show the applicability of the approach.

**Keywords:** MDA, service engineering, modeling

## I. INTRODUCTION

Model-Driven Development (MDD) is a well-accepted software design approach that intends to improve software projects with respect to portability, productivity, and quality [8]. Traditionally, software architects gather business requirements in a specification document that software developers implement, which is impeded by the different mind sets of software architects and software developers, the time that the requirement implementation takes, and the inability to react to rapid requirement changes [20]. In contrast, MDD prescribes the utilization of models in order to specify requirements as well as to generate valid software artifacts automatically based on these models. A formalization of the MDD approach is the Model Driven Architecture (MDA) [8] proposed by the OMG.

The Integrated Service Engineering (ISE) framework [17], [18] provides a comprehensive, interdisciplinary approach to develop tradable business services. In the ISE context, however, analysts as well as architects use and produce numerous artifacts throughout the service engineering process. Maintaining consistency for all artifacts and their relationships manually is time-consuming and prone to errors. In order to tackle this issue, the intention is to apply MDA in the ISE framework by defining service artifacts as formal models and the relationships between them as model transformations in order to ensure the correctness, consistency and quality of artifacts.

The main question that drives this research is whether MDA and its model transformation technology can be successfully integrated into the ISE framework in order to improve service development time, to reduce implementation mistakes, and to advance the ability to incorporate rapidly changing requirements. For doing so, this paper presents a general architecture for MDA, a list of available technologies to build an MDA-tool chain for service engineering [17] as well as a valid transformation example.

This paper is structured as follows: section II provides a brief overview of MDA, the ISE framework, and related work. Whereas section III presents a generic architecture for MDA, section IV discusses available technologies in order to realize the architecture proposed for service engineering and shows an example of a model transformation in the service engineering process. Section V concludes this work with prospects for future work.

## II. PRELIMINARIES

Before introducing the generic MDA architecture and its application for service engineering, this section briefly sketches the ISE framework, MDA, and highlights related work.

### A. The Integrated Service Engineering (ISE) Framework

The ISE framework [17] supports service engineering in terms of *planning, designing and implementing services, which are traded over the Internet, in addressing stakeholders from business & IT, and acknowledgment of different service aspects*. Figure 1 shows that ISE relies on the Zachman framework [22] and follows a divide and conquer approach. ISE is not only limited to computing services. Rather, it targets business services, such as insurance & financial services, civil services, marketing services, and telecommunication services. The vertical axis in figure 1 represents four perspectives of the engineering process and is named *service perspectives*. Each perspective relates to a specific role with appropriate skills and offers different sets of tools and methods. It also implies the chronology of the framework. Additionally, the perspectives are linked to phases of the service engineering process. The horizontal axis (aspects) in figure 1 allows for five different descriptions of a service. Each description is valid for each perspective. Any intersection in the matrix is a placeholder

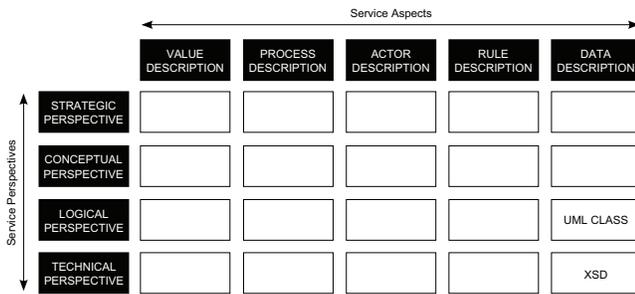


Fig. 1. ISE Framework [17]

for a *model*, a *notation*, and a *modeling technique* which is appropriate for the respective perspective and the modeling aspect. For a complete introduction to the ISE framework we refer to previous work [17], [18].

### B. Model-Driven Architecture (MDA)

MDA [8] is a software design approach utilizing models during requirements engineering and software development. However, the term *architecture* does not refer to the architecture of the system being built, but rather to the architecture of technologies under the umbrella of the OMG. Figure 2 gives an overview of these technologies by fitting them into the MDA landscape. To enable the model-driven approach, the fundamental technology is *Meta Modeling*. In the OMG context, the corresponding standard is the Meta-Object Facility (MOF) [10], which serves as the basis for metadata repository management and a common meta modeling language to define other modeling languages. MOF consists of two main packages, i.e., *Essential MOF* (EMOF) and *Complete MOF* (CMOF). EMOF is a subset of CMOF, which allows metamodels to be defined using only a few concepts. MDA does not constrain the types of models, as long as these models are expressed in a MOF-based language. This guarantees that the models can be stored in a MOF-compliant repository and transformed by MOF-compliant tools.

### C. Related Work

This subsection highlights related work in the area of applying MDA to specific domains.

López-Sanz et al. [6] present a model-driven approach in order to generate valid styles for service-oriented architectures. They argue that software requirements should be specified in a generic architectural style that is free of specific styles, such as service-orientation or component-orientation. Hence, they propose a model-driven technique to transform generic architectural designs into service-oriented architecture designs.

Bitsaki et al. [1] propose a model-driven architecture for generating business process stubs from service network models. They interpret service networks as service interactions between network partners and use a service network notation for modeling service networks. Each partner in this network, however, needs compliant business processes in order to interact with network partners. Consequently, Bitsaki et al. produce an architecture in order to generate valid processes from a given service network diagram.

Hahn et al. [5] introduce a model-driven approach that aligns multi-agent systems and semantic web services. Semantic web service technology utilizes ontologies for meaningful web service descriptions and improved service discovery. They introduce platform-independent models for agent systems and semantic web services, and present transformations between these models as well as to technical artifacts in the area of semantic web services and agent systems.

Similar to the approach presented in this paper, each aforementioned approach aims at integrating MDA to a specific domain. This approach, however, presents a generic MDA architecture for any domain and shows a valid extension for the service engineering process.

## III. GENERAL MDA ARCHITECTURE

Following the introduction of the ISE framework and MDA, this section presents a generic MDA architecture that is valid for any MDA implementation (cf. figure 2). The following subsections elaborate on its main components: (A) modeling technologies, (B) DSL technologies, (C) transformation technologies, and (D) an MDA repository.

### A. Modeling Technologies

Figure 2 presents a generic architecture for MDA. Its first component is the **Modeling Technology** that modeling experts employ to develop models. In line with MDA, modeling technologies can be grouped in two main sets, the *Generic Modeling Technologies* and the *Domain-specific Modeling Technologies*. The generic modeling technologies are industry standards elaborated or adopted by OMG, which include *Generic Modeling Languages* and *Generic Textual Languages*. Generic modeling languages are graph-based languages such as the Unified Modeling Language (UML) [14]; on the contrary, the generic textual languages such as UML Human-Usable Textual Notation (HUTN) [9] support a textual representation to conform to human-usability criteria. *Graphical Domain-specific Languages* (graphical DSLs) and *Textual Domain-specific Languages* (textual DSLs), on the other hand, are those modeling technologies that are dedicated to particular problem domains and created by individuals on demand. A valid approach to implement domain-specific languages is to use UML profiles [19].

In order to be integrated into the MDA landscape, both, generic languages and domain-specific languages need to be defined in MOF to ensure their interoperability. By providing a mapping from MOF to XML, OMG defines a standard format, the XML Metadata Interchange (XMI) [11], to facilitate the *Model Persistency* within MDA. XMI is a significant component for modeling technologies, which allows for models expressed in MOF-based languages (e.g. in UML) as well as the metamodels, which represent modeling languages themselves, to be rendered into an XML-based format for serialization. In the context of MDA, technologies such as *Model Query* and *Model Validation* are also required to underpin modeling technologies. The OMG defines the *Object Constraint Language* (OCL) [15] for this purpose. By using OCL expressions,

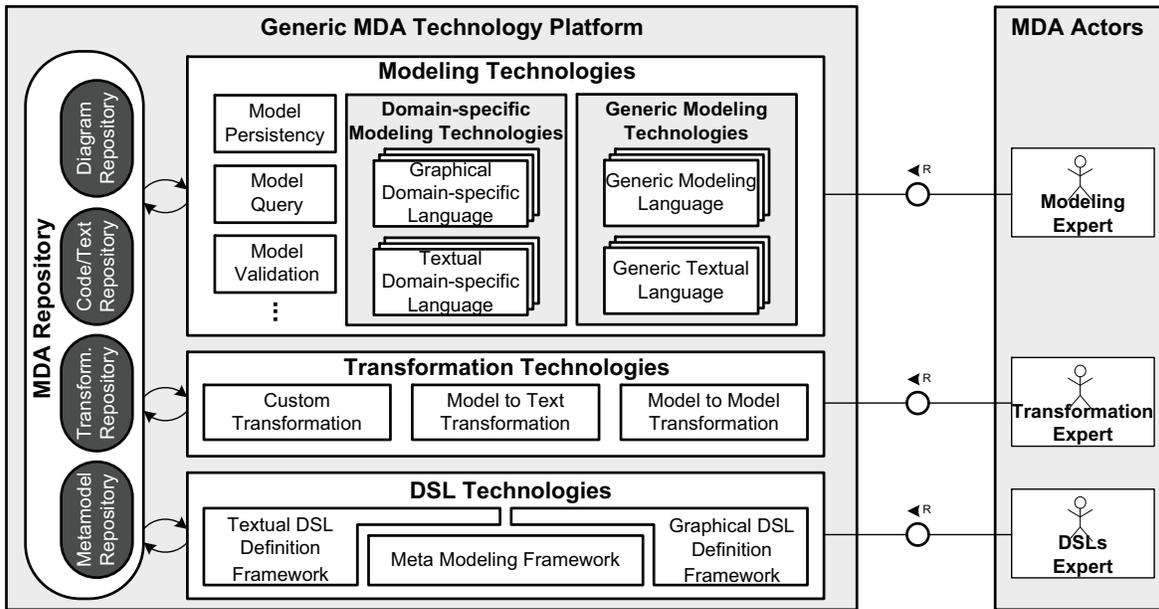


Fig. 2. Generic MDA Architecture

modeling experts describe additional constraints in their UML models and MOF-based models (in that case, only a subset of OCL is applicable) in order to validate them, as well as to specify queries on models without side effects.

### B. DSL Technologies

Figure 2 shows the **DSL Technologies** as the second part of the generic MDA architecture that a DSL expert utilizes to implement domain-specific languages (DSLs). The *Meta Modeling Framework* serves as the foundation for this architecture component by providing a core technology to develop the abstract syntax for DSLs. Within MDA, MOF is the dedicated standard for meta modeling, both for domain-specific languages and standardized modeling languages. For a domain-specific language, not only the abstract syntax is needed, but also a concrete syntax, which is defined as textual or graphical notation, so that it can be used by humans. Since in MDA the abstract syntax of a modeling language refers to a metamodel specified in MOF, the concrete syntax has to be mapped to it, in order to permit modeling experts to develop models by means of a text-based or graphical-based notation. As corresponding standards to specify the concrete syntax for modeling languages are missing in the OMG landscape, the generic MDA architecture has to rely on proprietary solutions, which are referred to as *Graphical DSL Definition Framework* and *Textual DSL Definition Framework* in figure 2.

### C. Transformation Technologies

MDA prescribes that modeling experts use the aforementioned modeling technologies for capturing platform-independent and platform-specific models [8]. Transformations between these models are often required for finally generating desirable artifacts (e.g. software code) that result in improved portability and quality, and reduced complexity. For this purpose, the **Transformation Technologies** are recognized as the key part

of the generic MDA architecture. Just like other technologies involved in MDA, model transformation also relies on the standard metamodel concept – MOF, which implies that the models (both, the source model and the target model) need to be expressed in MOF-based languages.

As shown in figure 2, transformation technology distinguishes three categories. *Model to Model Transformation* is dedicated to transforming information between models. The OMG specifies Query/Views/Transformations (QVT) [13] for this purpose, which relies on other OMG standards (e.g., OCL) in order to define queries and constraints on models during transformation. *Model to Text Transformation*, on the other hand, addresses how to transform information from models to text artifacts, including software code, reports, and documents. For this purpose, the OMG specifies the MOF Models to Text Transformation Language (MOFM2T) [12]. *Custom Transformation* refers to QVT's *black box implementation* that allows writing custom transformations in other languages (e.g. Java) for complex transformations that cannot be expressed in QVT itself.

### D. MDA Repository

The final part of the generic MDA architecture shown in figure 2 is the **MDA Repository**, which stores the MDA artifacts that are developed throughout the entire model-driven development process. It serves as the central component for data exchange and, hence, allows for the different architecture components of the generic MDA architecture to work together as a whole.

## IV. APPLYING MDA TO ISE

This section discusses which of the MDA technologies currently available are suitable for applying MDA to service engineering in a realistic setting. Based on an example from a case study reported in [18], the implementation of transforming an ISE data description illustrates the practicability of the approach proposed in section III.

TABLE I  
APPLYING MDA TO SERVICE ENGINEERING

GENERIC MDA PLATFORM	MDA FOR SERVICE ENGINEERING
<b>Modeling Technologies</b>	<b>Service Engineering Modeling Tools</b>
Generic Modeling Technologies	UML Modeling Tool UML Profile Modeling Tool BPMN Modeling Tool SBVR Modeling Tool
Domain-specific Modeling Techn.	Business Services Modeling Tool (cf. [19]) Conceptual Service Modeling Tool (cf. [19]) Business Activity Modeling Tool (cf. [18])
Model Persistency Model Query / Model Validation	XML Metadata Interchange (XMI) [11] Object Constraint Language (OCL) [15]
<b>DSL Technologies</b>	<b>DSL Toolset</b>
Meta Modeling Framework Graphical DSL Definition Framework Textual DSL Definition Framework	Eclipse Modeling Framework (EMF) Graphical Modeling Framework (GMF) Textual Modeling Framework (former oAW)
<b>Transformation Technologies</b>	<b>Transformation Toolset</b>
Model to Model Transformation	Procedural/Declarative QVT [13] ATLAS Transformation Language (ATL) [3]
Model to Text Transformation	Acceleo Xpand (former oAW)
Custom Transformation	Black box Implementations for QVT [13]
<b>MDA Actors</b>	<b>Service Engineering Actors</b>
Modeling Expert	Business Strategist Business Analyst IT Architect IT Developer
Transformation Expert DSL Expert	Transformation Expert DSL Expert

### A. MDA for Service Engineering

In order to improve the service engineering process, the generic MDA architecture is applied to the ISE framework. Because MDA specifies merely a set of standards and guidelines rather than concrete implementations and tools, a technical tool chain is needed to support ISE modeling and transformation. In this paper, *Eclipse* is used as an enabling platform.

Table I shows a valid mapping between MDA specifications (figure 2) and concrete technologies. A vantage point is the *Eclipse Modeling Project* (EMP). This is a collection of projects related to modeling technologies within the Eclipse ecosystem. The heart of the project is the *Eclipse Modeling Framework* (EMF), which includes the meta modeling language *Ecore* as a near-standard implementation of the OMG’s metamodel concept EMOF [4], as well as other related features such as model persistence, model query and model validation as implementations of appropriate OMG standards. By means of *EMF*, models can be expressed in *Ecore*-based languages on the Eclipse platform to enable an MDA approach.

Table I shows that the modeling technologies refer to **Service Engineering Modeling Tools** (SEMT) in order to develop service artifacts with supporting modeling languages (or model types) defined in the ISE framework. A set of these tools such as the *UML Modeling Tool* and the *BPMN Modeling Tool* are built on the basis of existing generic modeling languages, which have been chosen and tailored to suit the particular requirements of service engineering. For supporting these modeling languages, SEMT relies on the Model Development Tools (MDT) within EMP. The subset

of MDT such as BPMN and UML can be adopted by SEMT to support these standardized modeling languages.

The other set of tools such as the *Business Services Modeling Tool* and the *Conceptual Service Modeling Tool* [19] are based on domain-specific languages, which are specifically defined for the ISE framework and thus are only applicable in the service engineering domain. To support domain-specific languages as part of SEMT, the **DSL Toolset**, an implementation of MDA’s DSL technologies on the Eclipse platform, can be utilized. It relies on EMF to provide abstract syntax-development capabilities. For developing a concrete syntax there exist the *Graphical Modeling Framework* (GMF) and *Textual Modeling Framework* (TMF) within EMP [2]. Using GMF, one can develop a graphical concrete syntax for a DSL, based on the abstract syntax defined using EMF that together result in an Eclipse-based editor. TMF provides a textual concrete syntax and produces a textual editor, complete with syntax highlighting and code completion. The main component of TMF is *xText*, which originates from the *openArchitectureWare* (oAW) project [16].

Within EMP, both, standardized modeling languages as well as domain-specific languages are defined for the ISE framework based on the same *Ecore* metamodel concept, which makes model transformation possible without further integration efforts. The implementations of MDA’s transformation technologies, as shown in table I, are available as a **Transformation Toolset** on the Eclipse platform. For model to model transformation, *Procedural/Declarative QVT* serves as the implementation of OMG’s QVT standard; additionally, there exists also a QVT-like hybrid model transformation language – the *ATLAS Transformation Language* (ATL). For model to text transformation, one can use *Acceleo* as an implementation of the OMG’s MOFM2T standard, and a statically-typed template language – *Xpand*, which was originally developed as part of the *openArchitectureWare* project [16] before it became an Eclipse component.

The tools introduced previously can be integrated into a single **Service Engineering Workbench** (SEW) based on the *Eclipse Plug-in Framework*. It serves as the tool chain to underpin the ISE framework throughout the service engineering process. Moreover, the actors defined for MDA can also be adapted to the context of the ISE framework, namely as **Service Engineering Actors**, in that the different roles are assigned to modeling experts in terms of different

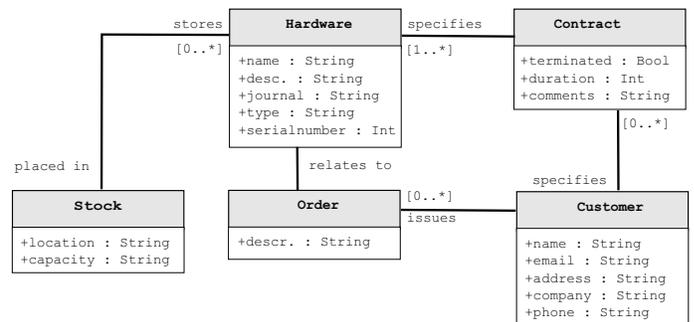


Fig. 3. ISE Data Artifact (cf. [18])

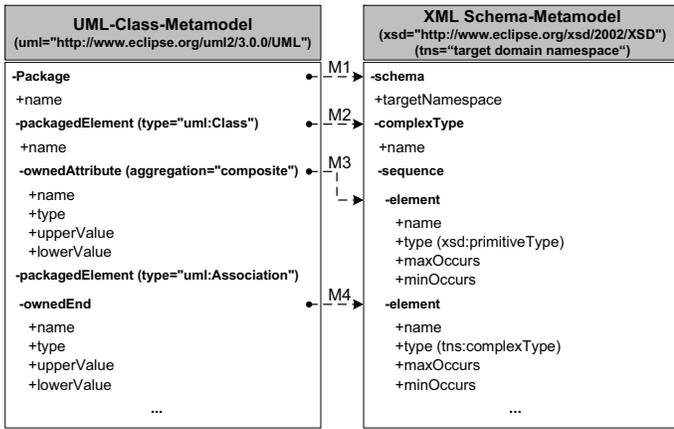


Fig. 4. UMLClass2XSD Mapping

service perspectives (cf. figure 1). Each of these roles owns appropriate skills and can utilize different sets of tools offered by the SEW to develop service artifacts.

### B. Transformation Implementation

This section illustrates an example for a data description transformation in the ISE framework using a case study in the IT outsourcing domain (for details, see [18]). This implementation should show whether MDA has a positive impact on development time, consistency between artifacts, and the ability to react to rapid changing requirements in the service engineering process. The transformation scope is to transform the logical data description into the technical data description automatically (cf. figure 1), whereas logical data descriptions utilize UML class models and technical data descriptions XML Schema Definitions (XSD). Figure 3 shows the UML class model, which an IT architect has developed using the UML modeling tool provided by the SEW.

Listing 1. UMLClass2XSD Transformation

```

1  modeltype UML uses "http://www.eclipse.org/uml2
   /3.0.0/UML";
2  modeltype XSD uses "http://www.eclipse.org/xsd
   /2002/XSD";
3  ...
4  transformation UMLClass2XSD(in clazz : UML, out
   xsd : XSD);
5  main() {
6    ...
7  }
8  — M3:property2element transformation
9  mapping Property::property2element():XSDParticle{
10     var xsdElement := object xsd::
       XSDElementDeclaration{
11         name := self.name;
12         typeDefinition := getXSDType(self.
           type.name);
13     };
14     — upperValue2maxOccurs transformation
       result.maxOccurs := self.upper;
15     — lowerValue2minOccurs transformation
       result.minOccurs := self.lower;
16     result.content := xsdElement;
17 }
18 }
19 }

```

Figure 4 shows an excerpt of the *vertical* mapping [7] between UML class models and XSD. This mapping is used afterwards for implementing the transformation with available transformation languages. Whereas the UML class model conforms to the *UML-Class-Metamodel* and the XSD file is conform to the *XML Schema-Metamodel*, both are defined on

the basis of Ecore to guarantee the interoperability during transformation.

Listing 1 illustrates an excerpt of the transformation implemented in Procedural QVT as supported by the SEW. The transformation accepts UML class models that conform to the UML metamodel as source and produces XML-schema that conform to the XSD metamodel. The function `main` in line 5 serves as the entry point of the transformation. For each mapping defined in figure 4 there exists a *mapping operation* as a corresponding implementation in QVT. For example, *M3* in figure 4 defines the mapping from `ownedAttribute` to `element`. This is implemented as the mapping operation `property2element` in line 9. It says that for each found `ownedAttribute` under `packagedElement` of type `uml:Class` in the UML class model, a corresponding `element` should be generated for an XSD `complexType`. The execution of this transformation with the source model depicted in figure 3 results in the XSD shown in listing 2.

Sometimes, defining a one-size-fits-all transformation in the context of the ISE framework is insufficient, especially when the design decisions have to be made during the model transformation process. In this case, the simplest solution is to supply these variable factors with a hard-coded default value in a transformation, or to extend the source model with extra data – both of which are undesirable. In order to tackle this issue without polluting the original source model, Vara et al. [21] propose a *Weaving Model* [3] as a way to annotate source models. In doing so, the annotation model captures these design decisions and is part of the transformation process. Every time when a design decision has to be made, the transformation process consults the annotation model to generate corresponding elements in the target model. This approach requires that the transformation engine can accept more than one source model.

Due to space limitations, the transformations implemented in declarative QVT and ATL are not shown. The comparison between these transformation languages is carried out in different scenarios to test their applicability within the scope of service engineering. It turns out, based on a taxonomy of model transformations [7], that the declarative QVT language is appropriate in the case of endogenous transformations (same metamodel), whereas the procedural QVT language is more applicable when the transformation is exogenous (different metamodels) and the source models and target models are strongly heterogeneous. ATL, on the other hand, because of its property of combining both, procedural and declarative transformation approaches, is positioned in between.

## V. CONCLUSION & FUTURE WORK

MDA is a well-accepted software design approach that intends to improve software projects with respect to portability, productivity, and quality. Whereas MDA concepts and corresponding specifications are available for some time, the adoption and implementations of these specifications, especially for model transformation languages, are just emerging. This paper sheds some light on this matter in that it presents modeling

transformation languages and a general architecture for MDA. Moreover, a valid mapping to existing MDA technologies for applying MDA in a service engineering process [17] as well as a valid transformation example from the service engineering domain are presented.

Listing 2. The generated XSD

```

1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <xsd:schema xmlns:tns="DataDescription" xmlns:xsd="
  http://www.w3.org/2001/XMLSchema" targetNamespace
  ="DataDescription">
3 <xsd:complexType name="Stock">
4 <xsd:sequence>
5 <xsd:element maxOccurs="1" minOccurs="1" name="
  capacity" type="xsd:string"/>
6 <xsd:element maxOccurs="unbounded" minOccurs="0"
  name="location" type="xsd:string"/>
7 <xsd:element maxOccurs="unbounded" minOccurs="0"
  name="hardware" type="tns:Hardware"/>
8 </xsd:sequence>
9 </xsd:complexType>
10 <xsd:complexType name="Hardware">
11 <xsd:sequence>
12 <xsd:element maxOccurs="1" minOccurs="1" name="
  name" type="xsd:string"/>
13 <xsd:element maxOccurs="1" minOccurs="1" name="
  serialnumber" type="xsd:integer"/>
14 <xsd:element maxOccurs="1" minOccurs="1" name="
  journal" type="xsd:string"/>
15 <xsd:element maxOccurs="1" minOccurs="1" name="
  type" type="xsd:string"/>
16 <xsd:element maxOccurs="1" minOccurs="1" name="
  description" type="xsd:string"/>
17 <xsd:element maxOccurs="1" minOccurs="1" name="
  stock" type="tns:Stock"/>
18 </xsd:sequence>
19 </xsd:complexType>
20 <xsd:complexType name="Customer">
21 ...
22 </xsd:complexType>
23 <xsd:complexType name="Contract">
24 ...
25 </xsd:complexType>
26 <xsd:complexType name="Order">
27 ...
28 </xsd:complexType>
29 </xsd:schema>

```

The MDA implementation of the service engineering architecture shows that MDA is applicable to service engineering. In terms of ISE, it can be said that procedural QVT is the most fitting transformation language, since ISE features rather heterogeneous models. Furthermore, applying MDA to service engineering has a positive impact on development time, implementation mistakes, and model consistency. Future case studies, like the one presented in [18], have to show whether MDA has a positive impact on the ISE framework's performance in terms of rapid changing requirements. The conclusion about transformation languages is currently limited to the data transformation example presented. More transformation implementations will follow in future work. The tool chain is under development; so far, the Eclipse environment is capable of supporting the task of combining the different tools and implementing the MDA architecture proposed for the service engineering domain.

#### ACKNOWLEDGMENTS

This project was funded by means of the German Federal Ministry of Economy and Technology under the promotional reference "01MQ07012". The responsibility for the content of this publication lies with the authors.

#### REFERENCES

- [1] BITSAKI, M., DANYLEVYCH, O., VAN DEN HEUVEL, W.-J., KOUTRAS, G., LEYMAN, F., MANCIOPPI, M., NIKOLAOU, C., AND PAPAZOGLU, M. P. Model transformations to leverage service networks. In *ICSOC Workshops* (2008), pp. 103–117.
- [2] ECLIPSE GMP|TMP. Graphical|textual modeling framework - homepage. <http://www.eclipse.org/modeling/gmf|tmf/>, 2010.
- [3] FABRO, M. D. D., BEZIVIN, J., JOUAULT, F., BRETON, E., AND GUELTAS, G. Amw: a generic model weaver. In *Proceedings of the 1re Journe sur l'Ingnerie Dirige par les Modles (IDM05)* (2005).
- [4] GRONBACK, R. C. Eclipse modeling project: A domain-specific language (dsl) toolkit, March 2009.
- [5] HAHN, C., NESBIGALL, S., WARWAS, S., FISCHER, K., AND KLUSCH, M. Model-driven approach to the integration of multiagent systems and semantic web services. In *EDOCW '08: Proceedings of the 2008 12th Enterprise Distributed Object Computing Conference Workshops* (Washington, DC, USA, 2008), IEEE Computer Society, pp. 317–324.
- [6] LÓPEZ-SANZ, M., VARA, J. M., MARCOS, E., AND CUESTA, C. E. A model-driven approach to weave architectural styles into service-oriented architectures. In *MoSE+DQS '09: Proceeding of the first international workshop on Model driven service engineering and data quality and security* (New York, NY, USA, 2009), ACM, pp. 53–60.
- [7] MENS, T., CZARNECKI, K., AND GORP, P. V. A taxonomy of model transformations. In *Language Engineering for Model-Driven Software Development* (2004), J. Bzivin and R. Heckel, Eds., vol. 04101 of *Dagstuhl Seminar Proceedings*, Schloss Dagstuhl, Germany.
- [8] OBJECT MANAGEMENT GROUP (OMG). Specification: MDA Guide V1.0.1. <http://www.omg.org/mda/>, June 2003.
- [9] OBJECT MANAGEMENT GROUP (OMG). Specification: UML Human-Usable Textual Notation (HUTN), Version 1.0. <http://www.omg.org/spec/HUTN/1.0/PDF>, August 2004.
- [10] OBJECT MANAGEMENT GROUP (OMG). Specification: Meta Object Facility (MOF), Version 2.0. <http://www.omg.org/spec/MOF/2.0/PDF>, January 2006.
- [11] OBJECT MANAGEMENT GROUP (OMG). Specification: MOF 2.0/XMI Mapping (XMI), Version 2.1.1. <http://www.omg.org/spec/XMI/2.1.1/PDF>, December 2007.
- [12] OBJECT MANAGEMENT GROUP (OMG). Specification: MOF Model to Text Transformation Language (MOFM2T), Version 1.0. <http://www.omg.org/spec/MOFM2T/1.0/PDF>, January 2008.
- [13] OBJECT MANAGEMENT GROUP (OMG). Specification: MOF Query/View/Transformation(QVT), Version 1.0. <http://www.omg.org/spec/QVT/1.0/PDF/>, April 2008.
- [14] OBJECT MANAGEMENT GROUP (OMG). Specification: Unified Modeling Language (UML), Version 2.2. <http://www.omg.org/spec/UML/2.2/Infrastructure/PDF>, February 2009.
- [15] OBJECT MANAGEMENT GROUP (OMG). Specification: Object Constraint Language(OCL), Version 2.2. <http://www.omg.org/spec/OCL/2.2/PDF>, February 2010.
- [16] OPENARCHITECTUREWARE. openArchitectureware - homepage. <http://www.openarchitectureware.org/>, 2009.
- [17] SCHEITHAUER, G., AUGUSTIN, S., AND WIRTZ, G. Business Modeling for Service Engineering: Toward an Integrated Procedure Model. In *Proc. of the 21st Intern. Conf. on Software Engineering & Knowledge Engineering (SEKE'2009)* (Boston, MA, USA, July 1-3, 2009).
- [18] SCHEITHAUER, G., KETT, H., KAISER, J., HACKNER, S., HU, H., AND WIRTZ, G. Business Modeling for Service Engineering: A Case Study in the IT Outsourcing Domain. In *SAC 2010, 25th Symp. On Applied Computing, Enterpr. Eng. Track* (Sierre, Switzerland, 2010).
- [19] SCHEITHAUER, G., AND WIRTZ, G. Business Modeling for Service Descriptions: A Meta Model and a UML Profile. In *APCCM 2010, 7th Asia-Pacific Conference on Conceptual Modelling* (Brisbane, Australia, January, 18-21 2010), pp. 79–88.
- [20] SCHEITHAUER, G., WIRTZ, G., AND TOKLU, C. Bridging the Semantic Gap between Process Documentation and Process Execution. In *Proc. of the 20th Intern. Conf. on Software Engineering & Knowledge Engineering (SEKE'2008)* (Redwood City, CA, USA, July, 1 - 3 2008).
- [21] VARA, J. M., DE CASTRO, M. V., DEL FABRO, M. D., AND MARCOS, E. Using Weaving Models to automate Model-Driven Web Engineering proposals. In *Proc. ZOCO'08: Integracin de Aplicaciones Web. Gijn* (2008), pp. 86–95.
- [22] ZACHMAN, J. A. A Framework for Information Systems Architecture. *IBM Systems Journal* 26, 3 (1987), 276–292.

# A Model-based Business Process Diagnosis Method in Service-Oriented Architecture\*

Soo Ho Chang and Soo Dong Kim  
Department of Computer Science  
Soongsil University, Seoul, Korea  
sooho.chang@gmail.com sdkim@ssu.ac.kr

## Abstract

*Service-Oriented Architecture (SOA) has been widely accepted as a development paradigm for application systems. An application system in SOA is composed of heterogeneous and distributed service components and it provides integrated functionalities to end users. When abnormal situation happens in the service system, it is complicated to identify origins that caused the abnormality. That is, fault diagnosis at runtime is one of challenges in service system management. In this paper, we present a method to diagnose service faults at runtime with MBR approach. A target service to be diagnosed is composed of more than one web services based on a business process. The method in this paper receives input values of the target business process, observed values on the business process execution, and output value, and then analyzes suspicious services in the business process based on the predefined business process model.*

## 1. Introduction

Service-Oriented Architecture (SOA) has been widely accepted as an application system development paradigm which provides concepts of service providers, consumers, and their interactions. An application system in SOA is composed of heterogeneous and distributed service components and it provides integrated functionalities to end users. In the architecture, there are two layers of service provider-and-consumer relationships; *end-users and integrated services and the integrated services and (atomic) service components*

Since there are several parties, their requirements and roles, and various situations on service executions, it is complicated to manage the integrated services. More specifically, services are composed into a business process with appropriate order and service level agreement (SLA), and the business process is provided to end users. When abnormal situation happens, it is not easy to identify the origin that caused the abnormality. That is, fault diagnosis at runtime is one of challenges in service system management.

Model-Based Reasoning (MBR) is a formal approach to

diagnosing faults by deducting possible faulty sets based on predefined system models and faulty models. Business process in SOA provides a meta-model which can fully describe the structure of the business process including control flows as well as data flows. We expect that MBR can be effectively utilized to diagnose faults in business process executions.

This research is to provide a method to diagnose service faults at runtime with MBR approach [1]. A target service to be diagnosed is composed of more than one web services based on a business process. The method in this research receives input values of the target business process, observed values on the business process execution, and output value, and then analyzes suspicious services in the business process based on the predefined business process model.

For the value of this research, it extends MBR-based diagnosis into a business process diagnosis in service system. Pure MBR can be generally applied into a simple system structure such as circuit models, because it needs to be completely defined as a model that is the basis in MBR. Once the model defined, MBR provides an inference system and it does not require historical data which is one of limitations of probabilistic reasoning. However, business processes in SOA have complicated structure and characteristics for MBR to be applied into the business process diagnosis. This research represents an extension of MBR to diagnose business processes in SOA.

## 2. The Diagnosis Method Overview

In this section, we identify essential artifacts in general MBR-based diagnosis and refine required artifacts for business process diagnosis in SOA. Based on the defined artifacts in this section, diagnosis method in section 4 will be represented.

### 2.1. Simple Example of MBR-based Diagnosis

In MBR-based diagnosis, circuit examples are commonly referred to explain the concept of the model and the observation in MBR because they are simple enough to make the concept clear. Figure 1 shows a simple circuit example of MBR-based diagnosis.

---

\* This research was supported by the National IT Industry Promotion Agency (NIPA) under the program of Software Engineering Technologies Development.

The a) shows a circuit of Half Adder which consists of a *XOR* and an *AND* gates, gets two bits(A, B in the figure) of inputs, and puts two bits (S, C). When the circuit is executed, there are four possible cases of input values ((0,0), (0,1),(1,0),(1,1)). For each case, outputs will be either Normal ( $\neg Ab$ ) or Abnormal (Ab).

The b) shows the expected function results of the two gates, depending on the pair of input values. If *XOR* and *AND* gates are normal then they put 1 and 0. If they are abnormal, they put 0 and 1 respectively. Therefore, with input and output values from executions, we can infer whether the gates are normal or abnormal.

Entailment in c) is to show what status of the gates results in observed data. In a), we have two inputs 1 and 0, and two outputs 0 and 0. Based on the expected results model in b), normal *AND* gate and abnormal *XOR* gate can entail the output 0 and 0. The second line in c) shows the gates set entailing the output values with the input values. Therefore, we infer that the *XOR* gate is abnormal when it gets 1 and 0 and puts 0 and 0.

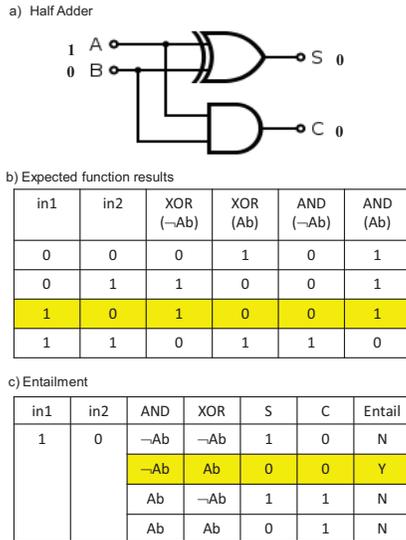


Figure 1. A Simple Circuit Example for Model-based Diagnosis

From the circuit example, we can identify essential artifacts in MBR, which can be used for our diagnosis method as followings;

- *System Structure*, a) in the Figure 1, showing components in the system and their relationships: In this example, it has two gates, *AND* and *XOR*, their connections, and two inputs and outputs.
- *Expected Results Description*, b) in the Figure 1, showing how the components will work when either it is normal or abnormal: The b) includes a fault model showing values when the gates are abnormal, and a truth model showing values when they are normal.
- *Input Values* to the system execution and observed *Output Values* from the execution

## 2.2. MBR-based Business Process Diagnosis

In SOA, services are composed into a coarse-grained unit called Business Process (BP). End users send a request to a BP, and then the BP sends messages to its web services and returns execution results to the end users. Our target to diagnose here is the BP just like the example of the gates set in the section 3.1.

In order to describe business process diagnosis, we define artifacts that are derived from general MBR and refined for the business process diagnosis in SOA.

The first artifact required is a system structure and its description where the system structure is about the composite service, business process.

Def 1. Let *Knowledge Base (KB)* be a system description that contains a business process.

Business process consists of more than one service and their control flow. It gets input from end users, sends requests to web services with appropriate input parameters, and finally returns output values to the end users. To interact with end users and web services, BP defines service level agreements (SLA); SLA between end users and BP, and between BP and web services. SLA usually provides contracts for Quality of Service (QoS) such as response time or throughput.

Def 2. Let *Settings* be initial values with which BP is executed.

Settings include input data to BP and initial values for expect QoS such as start time of the BP to calculate end-to-end execution time. Depending on the types of SLAs, different initial values can be included into the Settings.

Def 3. Let *Observation* be a set of observed values while BP is executing and after the execution.

Observation in MBR generally includes output values from an execution of the target system, such as 0 and 0 in the circuit example in the section 3.1. However, the observation on BP executions includes return values, QoS values of each web service execution, and states of service components before/after the executions.

Def 4. Let  $\{\alpha\}$  be a set of causes deriving abnormality of BP execution.

When an abnormal situation occurs in a BP execution, it needs to clarify the causes. In this paper, we assume that services on BP are the unit that may contain the causes deriving the abnormal situation. Therefore,  $\{\alpha\}$  is a set of web services which probably cause the unexpected BP execution such as unsatisfied QoS or return values out of expected type.

Def 5.  $KB \cup Settings \cup \{\alpha\} \Rightarrow Observation$

We define Diagnosis as a set of steps to find  $\{\alpha\}$  in Def 5 which accounts for the Observation on KB and Settings.

## 3. Process for Business Process Diagnosis

Based on the definition of diagnosis in section 3, we now represent a diagnosis process to identify  $\{\alpha\}$ . The process includes two phases, *preparation phase* and *diagnosis phase*,

and there are five steps on the two phases.

For understandability, we use a simple business process which contains two web services as shown in the Figure 2. The BP,  $BP_{simple}$ , gets two inputs, returns one output, and consists of two web services each of which has two inputs and one output.

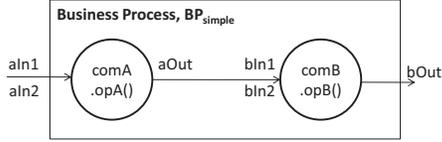


Figure 2. A Simple Example of Business Process

### 3.1. Preparation Phase

**Step 1. Building KB.** This step is to represent models of the target business process and its required service components providing web services of the business process. The KB includes 1) services, 2) data and control flows among the services, 3) SLAs, and 4) service components providing functionalities of the services.

*Service* is described with operation prototypes and URLs where an operation prototype includes an operation name, input types, and an output type. The input and output types are used to analyze abnormality of input and return values for web services in diagnosis steps. (\* means multiple elements)

$$\text{Service} = \{OpName, InType^*, OutType\}$$

In the  $BP_{simple}$  example, there are two services; Service A = {opA(), {aIn1:int, aIn2:String}, aOut:Boolean} and Service B = {opB(), {bIn1:Boolean, bIn2:String}, bOut:String}.

*Control Flow* shows a sequence of service invocations whereas *Data Flow* represents relationships of input and output data to/from web services.

$$\text{ControlFlow} = \{PrevService \rightarrow NextService^*\}$$

Control flow in the example is represented as {beginning  $\rightarrow$  opA(), opA() $\rightarrow$  opB(), opB() $\rightarrow$ end} and data flow is {aIn1  $\rightarrow$  bIn2, aIn2 $\rightarrow$ bIn2, aOut $\rightarrow$ bIn1}

*SLA* includes two parties and their agreements represented with quality attributes, measures, and expected threshold.

$$SLA_{Party1, Party2} = \{Party1, Party2, \{QualityAttribute, MeasureType, ExpectedValue\}\}$$

For example, the  $BP_{simple}$  defines response time of the service A such as  $SLA_{BP, ServiceA} = \{BP, ServiceA, \{Performance, response\ time, less\ than\ 3sec\}\}$

*Service component* provides system information where *Services* in a BP are deployed and served. Possible information by the Service Component can be represented with different types of *States* such as ‘Available’, ‘Overloaded’, ‘Unknown’, ‘Not responding’, and so on. Moreover, it can also be classified into ‘Pre’ and ‘Post’ conditions based on service execution time point.

**Step 2. Acquiring Settings.** This step is to acquire initial input values for a business process execution. The initial

values include input values from end users and values for getting SLAs.

**Step 3. Acquiring Observations.** This step is to acquire execution results including return values from business process, QoS values for SLAs, and service components’ status.

A business process is usually executed on Enterprise Service Bus (ESB) and the ESB can intercept execution values from the business process executions. Therefore, the *Settings* and *Observations* can be acquired by the ESB or other platforms, which is an underlying middleware to integrate services executions.

### 3.2. Diagnosis Phase

Once KB is specified, and Settings and Observation are acquired, we can diagnose the target business process. In this approach, we define two types of faulty services in service diagnosis, *absolute faulty services* and suspicious faulty services, and we describe diagnosis steps with the classification.

**Step 4. Finding set of absolute faulty services.** Absolute faulty service is the services whose faults can be directly identified. From the observed value, some faulty services can be directly identified for instance, if  $responseTime(s) > expectedResponseTime(s)$  for a service, s, then the s is faulty. This step is to find the absolute faulty services in a business process execution.

For the purpose, we define a comparison table which shows how the observation is going with the predefined business process model, i.e. ‘entailed’ or ‘consistent’ in MBR. However, the comparison table in this approach is adapted to SOA diagnosis with characteristics from business process executions. In pure MBR, when ‘entailment’ in abductive diagnosis or ‘consistency’ in consistency-based diagnosis is derived, they cannot acquire input and output values of each function in a whole process such as a gate of the circuit example in the section 3.1. Thereby, they deduct the input and output values of each function based on predefined system model (i.e. truth model) and fault model, and identify normal and abnormal sets of functions in a whole process.

However, it is possible for business process diagnosis to monitor input and output values of each service even though not all of them are possible due to the limited expenses. Therefore, it is more straightforward to identify faulty services in this approach. However, diagnosis in the business process, we have more criteria to compare with in order to decide its abnormality such as QoS values for SLA and status of service components, as well as input and output values of each service.

For the  $BP_{simple}$  example, we may assume that we get following observations;

- Observation on Output:  $out(opB) == \text{Data Exception}$
- Observation on monitored states:
  - ServiceComponent(CompA)  $\wedge$  postState(opA) = “available”
  - ServiceComponent(CompB)  $\wedge$  postState(opB) =

“unavailable”

- Observation on QoS values:  
 $ServiceComponent(CompA) \wedge responseTime(opA) = 2 \text{ sec}$   
 $ServiceComponent(CompB) \wedge responseTime(opB) = 7 \text{ sec}$

With the example, we can represent a comparison table as shown in the Table 1. Columns for  $\neg Ab$  (i.e. not Abnormal) and  $Ab$  (i.e. Abnormal) show conditions to be satisfied on their left side and the value of the conditions on their right side where T, U, and F mean True, Unknown, and False respectively. Each condition on left side for each case is from KB while the values are from comparison with Observation.

Table 1. Comparison Table for Diagnosis

$\neg Ab(opA)$		$Ab(opA)$	
$in1(opA) \in String$	T	$in1(opA) \notin String$	F
$in2(opA) \in String$	T	$in2(opA) \notin String$	F
$out(opA) \in String$	U	$out(opA) \notin String$	U
$responseTime(opA) \leq 3$	T	$responseTime(opA) > 3$	F
$preState(CompA) == \text{“available”}$	U	$preState(CompA) != \text{“available”}$	U
$postState(CompA) == \text{“available”}$	T	$postState(CompA) != \text{“available”}$	F
Result	U	Result	U
$\neg Ab(opB)$		$Ab(opB)$	
$in1(opB) == aOut$	U	$in1(opB) != aOut$	U
$in2(opB) == (in1(opA) + in2(opA))$	U	$in2(opB) != (in1(opA) + in2(opA))$	U
$out(opB) \in String$	F	$out(opB) \notin String$	T
$responseTime(opB) \leq 4$	F	$responseTime(opB) > 4$	T
$preState(CompB) = \text{“available”}$	U	$preState(CompB) != \text{“available”}$	U
$postState(CompB) = \text{“available”}$	F	$postState(CompB) != \text{“available”}$	T
Result	F	Result	T

In the business process, there are two services, opA and opB, so we need to figure out four cases of comparison that are  $\neg Ab$  and  $Ab$  cases for the two services. For  $\neg Ab$ , the all conditions must be satisfied if the service is normal, while if one condition of the list of conditions is satisfied, the service is abnormal. For example, opB is abnormal because some of the conditions are unknown and two conditions are satisfied with KB. As a result, the opB is the absolutely abnormal service in the business process.

**Step 5. Finding set of suspicious faulty services.** Suspicious faulty services are the services which are not clear to identify their abnormality due to the limited observation value. This step is to find such suspicious services in order to recommend more investigation.

In this step, we take unknown service in the comparison table from the Step 4, and explore relationships with absolutely faulty services. The most representative relationships can be data relationship and deployment relationships. Suppose that we have two services, s1 and s2 in a business process.

- Data relationship: If s1.out is used for s2.in, then they are

related in terms of data relationship.

- Deployment relationship: If s1 and s2 are deployed in the same service component and the service provider, they are related in terms of deployment relationship.

Depending on the unsatisfied conditions of the absolutely faulty services in step 4, if unknown services are related to the services with the same criterion, we chose them as suspicious faulty services.

In the simple example, the output of opA and inputs of opB which are related are unknown. However, the output of opB is data exception. Therefore, the input of opB which is originally from opA may be the root cause of the data exception. So, more investigation on opA may be required.

From the diagnosis steps, we result that opA is suspicious and opB is absolutely abnormal. For the suspicious services, we may need to gather more information which is unknown in the comparison table. For the absolutely abnormal services, we may need to find out original causes deriving the abnormal services such as network problem, service download and overload, or application abrupture. We do not cover the issues in this paper.

#### 4. Concluding Remarks

Due to the non-conventional characteristics of SOA, services management presents hard challenges which are not presented in conventional systems management. A key problem in services management is to diagnose various faults and determining their causes, so that they can be eventually remedied. A desirable approach is to apply autonomous way of diagnosing the observed symptoms and to determine their causes at runtime without human administrators' intervention.

Model-based reasoning (MBR) is an effective diagnosis approach; however, its reasoning scheme is limited to diagnosing symptoms with output values. In this paper, we presented a diagnosis method extended the basic MBR to be able to diagnose faults found in various SOA components. Our method uses three different inference sources for diagnosis, *QoS*, *Component States*, and *input and out data*. We also defined *Absolute faulty services* and *Suspected faulty services* which are derived from difference probability of diagnosis.

We expect that this diagnosis method can be extended not only for end-to-end business process diagnosis but for deeper diagnosis into service components and their system which may be the very root cause of web service faults [2].

#### References

- [1] Bernhard Peischl and Franz Wotawa, “Model-Based Diagnosis or Reasoning from First Principles,” *IEEE Intelligent System*, Vol 18, Issue 3, Page(s): 32 – 37, IEEE, 2003.
- [2] Soo Ho Chang and Kwei-Jay Lin, "A General QoS Error Detection and Diagnosis Framework for Accountable SOA," *In the Proceedings of IEEE International Conference on e-Business Engineering (ICEBE) 2008*, Oct. 22-24, 2008.

# Ontology-Based Dependency-Guided Service Composition for User-Centric SOA

Wei-Tek Tsai<sup>\*+</sup>, Peide Zhong<sup>\*</sup>, Jay Elston<sup>\*</sup> and  
Yinong Chen<sup>\*</sup>

<sup>\*</sup>Computer Science and Engineering Department  
Arizona State University  
Tempe, U.S.A

{Wei-Tek.Tsai, Peide.Zhong,jelston,yinong}@asu.edu

Xiaoying Bai<sup>+</sup>

<sup>+</sup>Computer Science and Technology Department  
Tsinghua University  
Beijing, China  
baixy@tsinghua.edu.cn

**Abstract**—Service-Oriented Architecture (SOA) is characterized by dynamic service discovery and composition. For user-centric SOA, not only services, but workflows and application templates also can be published and discovered for composition. This paper proposes a two-steps composition process, ontology-based dependency-guided service composition (OBDG). In the first step, users compose their templates based on domain ontology or using existing templates. In the second step, users submit the templates to the OBDG system and let the system complete the dependencies by choosing and finalizing the selection of services and workflows from a set of candidate services or workflows based on user preferences. In this paper, services are described as interface service and implementation service. Interface service is like a service specification that describes service interface and its description, which is used to build domain ontology. Implementation services implement interface services. OBDG maintains a one-to-many relationship between them plus test scripts and cases. Templates are composed by interface services or test script so they can be instantiated by implementation services or test scripts and cases. In this way, application templates and testing templates can be same, which means OBDG can instantiate same template as application or testing by choosing implementation service or test script.

**Keywords**—Ontology, Service, Composition, Two-Steps, Interface service, Implementation service

## I. INTRODUCTION

Service discovery and composition are two important problems in Service-Oriented Architecture (SOA). Service discovery identifies a set of services that meet the specified requirements. Composition creates a composite service or an application that meets the requirements by reusing published services. To compose a composite service or an application, various SOA techniques such as orchestration, choreography, and coordination can be used.

Many service composition techniques have been proposed including model-based approaches [1-3], semantic-based approaches [4-6], template-based approaches [7, 8] or quality of service-driven approaches [9-11]. Model-based approaches mainly focus on how to model the process of service composition. Semantic-based approaches are mainly focused on how to describe services semantically so they can be easily found when do service composition. It is hard for them to build a workflow to compose discovered services. Template-based approaches help users to accelerate service composition process. Users can easily select or revise current template to

accomplish their tasks. It is difficult for them to match services needed in templates to real services. Template instantiation is a big problem. QoS driven approach focuses on how to select best service or best fit service for service composition. It has same problem with semantic based approach. Both of them are hard for users to change composition workflow.

Current service composition approaches are almost all provider centric. They can do very well in some aspects but are not suited for other aspects, which they either focus on the dynamic aspects of composition, or the static aspects, but these approaches tend not to handle both at the same time. However, both dynamic and static aspects are very important for composition. When users do not know which services they need, what they know is the goal they want. In this condition, dynamic approach will be very useful. And dynamic approach is easy to change services if there is any service out of work or error happens. But, if users know which service they need, static approach can be useful. If an approach can do dynamic and static service composition, it will be very useful because it can make use of dynamic and static characters.

In traditional SOA, service composition, service testing and users' requirements may need to be presented in different program language. These models are not unified, and each must be developed and maintained separately. This causes additional work in creating and maintaining compositions. To save users' time and effort, this paper proposes an ontology-based user-centric service composition (OBDG) approach. This approach allows users to build a single model that describes the requirements, the composition and the testing. Furthermore, templates for this approach allow users to compose applications or services quickly based on existing templates. With the help of domain knowledge, users can easily select fit templates or change them, which includes application template, workflow template, collaboration template and, etc. Domain ontology will be used to help compose and revise template. There are some dependency relationships among them and these dependency relationships will be used to compose template by OBDG. In this way, the composition process is now divided into two steps:

i. *Identification and Discovery*: Based on requirements, the user searches for a suitable template from among those that have been previously published. The identified templates come with a set of associated services, workflows and tests. These templates are the candidates for composition. Domain Ontology can be used for guiding user to search templates. That is, the templates from the user's application domain are

given more weight as being applicable to the user's needs. Information in the ontology can be used to compose or revise template. The OBDG system will complete the template with dependencies information and validate the consistency of the resulting template.

ii. *Composition and Instantiation*: Based on user preference, dependency information and other factors, the user can choose an appropriate set of services and workflows from the set of candidate services and workflows. Finally, OBDG instantiates the interface services or workflows from those implementation services are active.

This paper is organized as follows. Section II briefly reviews related SOA and service composition; Section III discusses how to compose OBDG domain ontology; Section IV illustrates how to do OBDG templates; Section V depicts transferring between use scenario and OBDG templates. Section VI presents a case study to illustrate the proposed composition approach; Section VII concludes this paper.

## II. RELATED WORK

Dependency information is often used for compiler optimization and change management [12]. This paper further extends dependency with likelihood information to assess the weight of dependency relationships. The information is useful in identifying those items that are most likely to be selected for composition.

Service composition has been a difficult task. Dustdar [13] classified composition strategies into five categories: 1) static and dynamic composition strategies, 2) model driven service composition, 3) business rule driven service composition, 4) declarative composition; and 5) automated and manual service composition.

Static and dynamic composition concerns the time when services are composed. Static composition occurs at design time. Services are chosen, combined together, and finally compiled and deployed. Sun [14] defines Microsoft Biztalk and Bea WebLogic as examples of static composition engines and Stanford's Sword and HP's eFlow as examples of dynamic service composition. In static composition, it is difficult to replace services with equivalent newer services. Dynamic composition was introduced as a method that allows service composition to replace services dynamically. However, dynamic composition is difficult and one issue is identification of appropriate services at runtime.

Orriens [3] introduced model-driven dynamic service composition where UML is used to provide a high-level of abstraction that can be directly mapped to other standards, such as BPEL4WS. They use OCL (Object Constraint Language) to express business rules and describe the process flow. Gronmo [2] proposed a model-driven semantic web service composition. They use OWL-S and WSMML as semantic web service description languages, and their method guides developers to compose services through four phases, starting with the initial modeling, and ending with a new composite service that can be deployed and published.

Ontology-based service composition approach is introduced in papers [15, 16]. Tasic in his paper [17] discussed the need for requirements for ontology, and provided ontology systems

for the management of services and for Quality of Service (QoS) metrics.

Tsai's paper [18] propose a service composition approach, which is Dependency-Guided service composition for User-Centric SOA. In this paper, ontology systems are used to express domain information and ontology systems cross reference each other with dependency relationships. This paper also illustrates how to express dependency information and how to use this dependency information to help service composition processes.

Ontology is often used for knowledge representation, sharing, classification, reasoning, and interoperability. Ernestas in his paper [19] proposed a method of transforming ontology representation from OWL to relational databases and algorithms for transformation of domain ontology to relational databases. Bianchini [20] described an ontology design approach defined in the framework of the VISPO (Virtual-district Internet-based Service Platform) project to support knowledge sharing and service composition in virtual districts. Kim [21] presented a task dependency approach for web service composition driven by business rules statically.

Current SOA composition emphasizes publishing and discovering services, and most of support is for service providers. UCSOA (User-Centric SOA) and CCSOA (Consumer-Centric SOA) [22, 23] are extensions to SOA that provide support to service consumers. These SOA allow various items such as user requirements to be published so that service providers can discover and supply the needed services or workflows. UCSOA also allows end users to compose and share applications in a community. In this way, a non-technical person can compose applications easily like they use mashup [24].

## III. OBDG DOMAIN ONTOLOGY

Domain ontology is often used to express domain information and it often represents entities, relationships and constraints. A service ontology example is illustrated in Figure2.

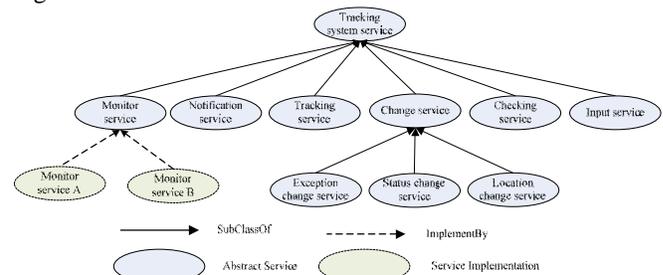


Figure 1 Shipping Domain Service Ontology

There are two related definitions to help represent domain ontology:

*Nodes*: A node represents an interface or concrete implementation of an interface service, workflow or test. Every interface node has corresponding implementations that have been verified to be correct. In this paper, each node's implementation must have same input and output as the interface node they are linked to so they can be dynamically replaced more readily.

*Relationships:* A relationship is a connection between two nodes in the domain ontology. Relationships are directional. A relationship consists of the dependent node, the relationship, and the target node.

Each interface service node is described by a service description [22, 23, 25]. Here, PSML-S [26] will be used to describe each node. It will present interface service Interface, Scenario and Constraints same as [25]. OBDG will maintain a *one-to-many* relation between an interface service and its implementations, as shown in Figure 1. When more than one instance exists, the one chosen is based on its ranking. There are many ways to rank services, such as best QoS, least expensive, most reliable, etc. Ranking mechanisms can be found in Tsai's paper [27].

As interface services have defined the interfaces that are shared by all of its implementations, test scripts can be defined that can be applied to all an interface service's implementations. Test scripts and cases relationships with service interfaces can be maintained in the domain ontologies using relationships. OBDG also needs to maintain one-to-many relationships to manage test scripts and cases. OBDG can perform static or dynamic testing of interface implementation services by calling the test scripts and test cases. *Implementation service providers* can publish their test scripts and test cases to illustrate the fitness of their services. For instance, providers can devise tests that only their services pass and publish those tests to show that their services are better than those from other providers. Users can also create tests that are representative of their specific usage scenarios and publish their test cases so only those implementation services that pass their test scripts will be candidates for their compositions. As with services, test scripts and cases can be reused and ranked.

Creating a domain ontology is a problematical problem because different domains have different ontologies and nobody can know all knowledge of all domains. With OBDG, domain ontology will be built up by using Web2.0 principles. Users can submit interface nodes to their domain and domain experts will check and make decisions if those interfaces are correct. After nodes have been created, all providers will be notified and they can see if they can provide implementations of the interface services. After they finish the implementations, they can register them. Once the implementation passes OBDG tests, they are published.

Domain ontology also maintains dependency information among their nodes. This dependency information will be helpful when users compose templates. Dependencies can be identified by A, B, and C as proposed by Tsai [18].

#### IV. OBDG TEMPLATES

OBDG can use templates for static, dynamic, or hybrid composition approaches. Templates can be populated by implementation services or interface services that come from service nodes of domain ontology. To be useful for actual execution, a template must be populated only by implementation services. If a template is populated only by implementation services, the composition is static. That is, it

can be run without any further interaction by OBDG. If a template consists of only interface services, OBDG will need to create an executable template by replacing interface services with implementation services as part of executing the template. This is a dynamic composition. Finally, if a template is populated by a mixture of implementation and interface services, a hybrid composition approach is taken where only the interface services are replaced with implementation services.

An OBDG template for dynamic composition is represented by domain ontology and control structures as illustrated in Figure 2. One benefit of this template-based approach is that service and test scripts or cases share the same template and it can be automatically completed with dependency support. OBDG will dynamically replace interface services with implementation services or test scripts based on dependency and service or test script rank, simulate and execute it. The interface services that are used to compose the template can also be a template, as long as this template has been registered, passed all tests and a one-to-many relationship exists between one node of the domain ontology and the template. As a template is composed by interface service, users with limited programming knowledge can revise it. They can change the control flow in the template by adding or removing control structures or replace interface services with their implementations. In OBDG, templates can be described by PSML-S [26]. PSML-S will provide many control constructs to help users to revise templates easily such as condition, parallel and sequence. Users can revise the templates by drag and drop. One PSML-S template is illustrated as Figure 3. It is very similar to the template in Figure 2 except it has a start point.

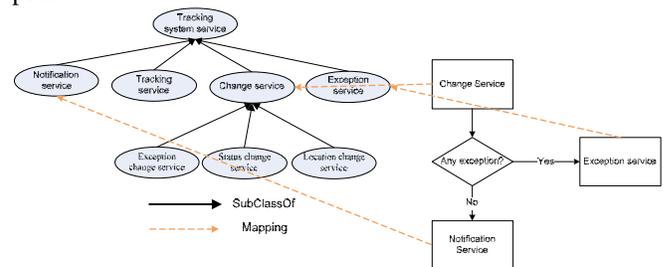


Figure 2 Service Template Mapping

As an interface service in a template can have many associated implementation services and test scripts, OBDG will choose one of implementation services or test scripts and replace it dynamically. This selection can be performed based on ranking mechanisms such as A, B, and C as introduced in Tsai's [18, 27].

#### V. USE SCENARIO AND OBDG TEMPLATE

A scenario is a semi-formal description of system functionality. It is a sequence of events expected during operation of system products which includes the environment conditions and usage rates as well as expected stimuli (inputs) and response (outputs). The use scenario is an extension to UML's use case and David Parnas' concept of use. It specifies how a service or system is used by other services or systems.

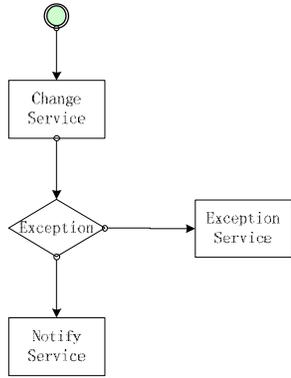


Figure 3 Template in PSML-S

Tsai [26, 30] propose a use scenario syntax, specification and tools to support it. Based on that, a use scenario can be easily translated to a template. Figure 5 shows a mapping between a use scenarios and a templates.

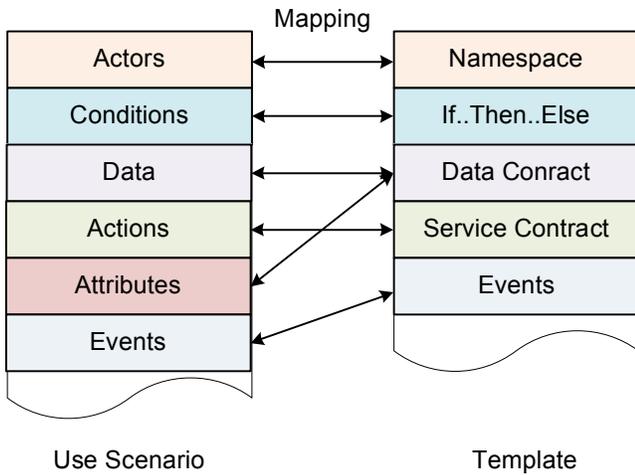


Figure 5 Use Scenario and OBDG Template Mapping

So, users can model *use scenario* that the system translates to an OBDG template. OBDG can perform model checking, verification & validation (V&V), simulation, and code generation on the resulting template. All these functions have been developed by Software Research Lab at Arizona State University.

## VI. CASE STUDY – SHIPPING DOMAIN TRACKING SYSTEM

This section uses an example system from the shipping domain to illustrate the OBDG composition process. We get real requirements from three different industry companies, which we incorporate with them. According to applications' requirement, three different application systems have been developed. A lot of services, application template, workflows and ontology have been published in Tsai's book [31] and one master thesis. For illustrating how to use OBDG system, a fourth company's requirements have been provided. The system consists of four participants, company manager who wants to see statistics data such as what is the profit of the company in this month, system administrator who will manage the system to make sure that system works well, carrier who wants to change the status of shipment and user who wants to track the status of shipment.

### A. Existing Items

Forty-six services, fifty-eight workflows and sixteen application templates have been published, which table 1 illustrates how they distribute.

	Company A	Company B	Company C
Service	16	17	13
Workflow	20	21	17
Application Template	5	6	5

Table 1 Existing Items in Different Applications

Shipping domain ontology has been presented as Figure1. Dependency information of the service ontology can be illustrated as Table 2.

Service Name	Dependency Service	Likelihood	Domain
Notification Service	Exception Service	40%	Shipping Domain
Notification Service	Tracking Exception	90%	Shipping Domain
Notification Service	System Exception	30%	Shipping Domain
Change Service	Notification Service	100%	Shipping Domain
Location Change Service	Notification Service	100%	Shipping Domain
Exception Change Service	Notification Service	100%	Shipping Domain
Status Change Service	Notification Service	100%	Shipping Domain
Tracking Service	Exception Service	20%	Shipping Domain
Tracking Service	Tracking Exception Service	50%	Shipping Domain
Tracking Service	System Exception Service	5%	Shipping Domain

Table 2 Dependency Information on Shipping Domain Ontology

### B. Specifications

The mission is to let a manager named Jerry to quickly compose an application according to his requirements. Jerry wants to get informed by cell phone of any tracking exception that occurs in his company's shipping system. Jerry does not know about programming, and the system does not provide any available service or application that he can use. Jerry will use PSML-S to publish his requirements and compose his new application. The process he uses is described below:

- i. Jerry logs into system with a role of manager.
- ii. Jerry checks the notification workflow for shipment exception.
- iii. Jerry updates the notification workflow.

The process how to use the PSML-S can be illustrated as follows:

- i. Search applications or templates from OBDG ontology to see if they satisfy their requirements

- ii. Revise selected template to satisfy user's requirements.
- iii. Ask OBDG system to do model checking and simulation.
- iv. Generate code based on user's preference.

C. Notification Workflow

The notification workflow can be constructed by use scenario tool and OBDG ontology. As user and manager are different roles in this case study, they may have different workflows. For demonstration, manager's notification workflow is illustrated in Figure 6.

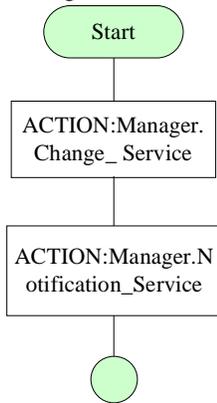


Figure 6 Notification Workflow

As manager does not need to consider any dependency information, he can focus on what he wants to do. OBDG system analysis the Change\_Service and find its dependency information. So, OBDG will automatically choose its dependency with Change\_Service for the workflow. It can be illustrated as Figure 7.

According to the description before, Change\_Service, Exception\_Service and Notification\_Service can be mapped to OBDG ontology, which is illustrated as Figure 8. Model-driven analyses

The PSML-S tool support lots of analyses including Completeness and Consistency (C & C) checking, policy specification, analysis, and enforcement, dependency analysis and ripple effect analysis and, etc. The first step is to perform C&C analysis on the specification to check the completeness and consistency [32]. If the specifications are supplemented to be complete, the SOA application builder can generate sequence diagrams for path analysis, usage analysis and concurrent analysis or perform dependency analysis.

D. Interface service Mapping to Implementation service

If there is not any violation after OBDG finishes the model-driven analyses, the OBDG system will select implementation services. For static composition, the OBDG system will choose the implementation services according to their ranks and the user's preferences and generate code a static templates populated entirely by implementation services. For dynamic compositions, the OBDG system will choose sets of fit services for you. When OBDG runs the workflow, it will choose fit services from the sets to run. This is shown in Figure 8.

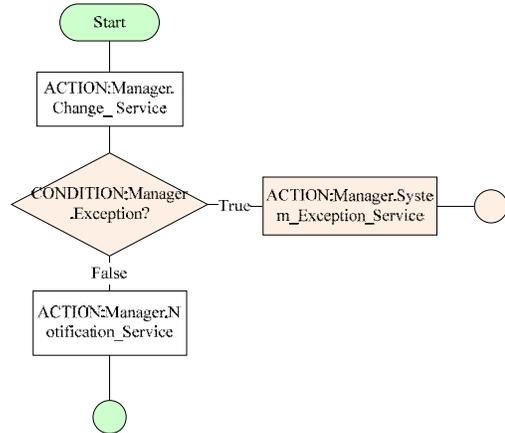


Figure 7 Notification Workflow with Its Dependency

If the manager prefers Company A's services, OBDG service will select interface services provided by company A if they are available. Thus, OBDG will automatically change Figure 7's workflow to Figure 8's workflow.

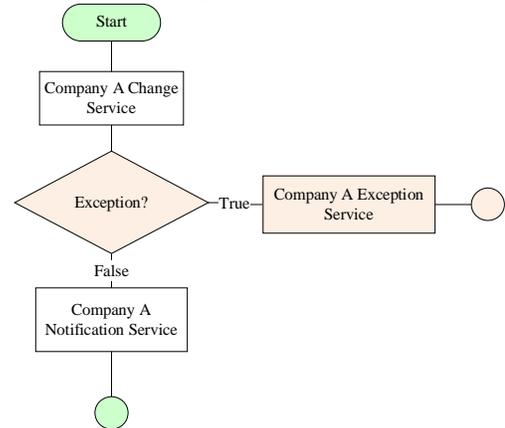


Figure 8 Revised Notification Workflow

As described as before, all implementation service must follow the interface service interface and the workflow needs to pass C&C check. So, the revised notification workflow can be simulated by PSML-S tools and generate code for the manager now.

E. Simulation

The OBDG simulation process is illustrated in Figure 9. The red color abstract action shows it is executed at this moment and it will change to original color after it has been executed. At same time, three implementation services available for ACTION:Manager.Change\_Service and Company A Change Service will be chosen in this scenario. If it is failed, PSML-S will choose Company B Change Service as its replacement.

VII. CONCLUSION

This paper proposes a user-centric, two-step ontology-based dependency-guided service composition (OBDG) process to assist people to compose applications. In the first step, users compose their templates based on domain ontology or use existing templates. In the second step users can submit them to the system and let the system complete dependencies choosing and finalize the selection of services and workflows

from a set of candidate services or workflows based on their preference.

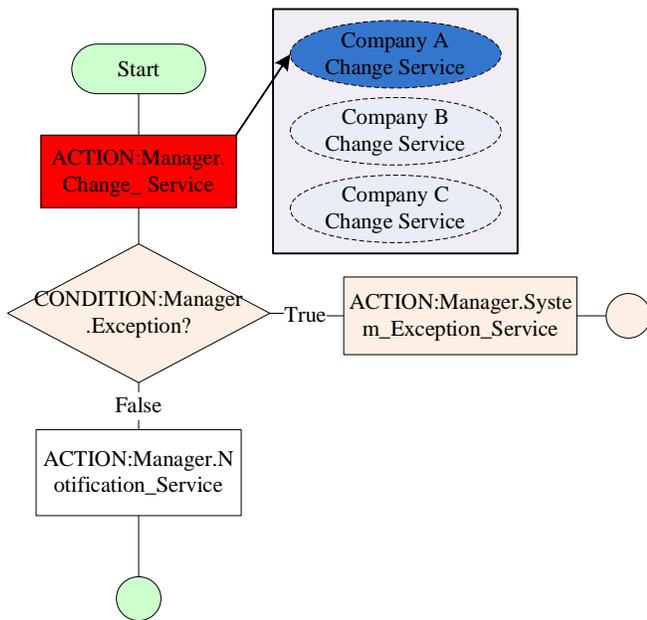


Figure 9 PSML-S Simulation Process

In this paper, services are described as interface service and implementation service. A domain ontology is used to organize and manage interface services and maintain a one-to-many relationship with instances services. Users use interface services to do service composition and let OBDG system choose implementation services for the composed service, which can alleviate users programming burden because users only need to concern logic business of program but not which services need to be selected. At last, a case study is used to illustrate the approach of this paper works well.

### VIII. ACKNOWLEDGMENT

The work is supported by U.S. Department of Education FIPSE project, and Korean ETRI project. The opinions expressed are of the authors only, and should not be taken as the position of U.S. Government.

### IX. REFERENCES

[1] L. Chen, *et al.*, "ECA Rule-Based Workflow Modeling and Implementation for Service Composition," *IEICE Transactions*, vol. 89-D, 2006, pp. 624-630.

[2] R. Gronmo and M. C. Jaeger, "Model-Driven Semantic Web Service Composition," *APSEC*, 2005, pp. 79-86.

[3] B. Orriens, *et al.*, "Model Driven Service Composition," *ICSOC*, 2003, pp. 75-90.

[4] K. Fujii and T. Suda, "Semantics-Based Context-Aware Dynamic Service Composition," *TAAS*, vol. 4, 2009.

[5] S. Kona, *et al.*, "USDL: A Service-Semantics Description Language for Automatic Service Discovery and Composition," *Int. J. Web Service Res.*, vol. 6, 2009, pp. 20-48.

[6] A. Brogi, *et al.*, "Semantics-based composition-oriented discovery of Web services," *ACM Trans. Internet Techn.*, vol. 8, 2008.

[7] E. Sirin, *et al.*, "Template-based composition of semantic web services," 2005, pp. 85-92.

[8] K. Geebelen, *et al.*, "Dynamic reconfiguration using template based web service composition," *MW4SOC*, 2008, pp. 49-54.

[9] Y. Dai, *et al.*, "QoS-Driven Self-Healing Web Service Composition Based on Performance Prediction," *J. Comput. Sci. Technol.*, vol. 24, 2009, pp. 250-261.

[10] L. Yang, *et al.*, "Performance Prediction Based EX-QoS Driven Approach for Adaptive Service Composition," *J. Inf. Sci. Eng.*, vol. 25, 2009, pp. 345-362.

[11] S. Meng and F. Arbab, "QoS-Driven Service Selection and Composition," *ACSD*, 2008, pp. 160-169.

[12] B. Ramesh and M. Jarke, "Towards Reference Models for Requirements Traceability," *IEEE Trans. Softw. Eng.*, vol. 27, January 2001.

[13] S. Dustdar and W. Schreiner, "A Survey On Web Services Composition," *IJWGS*, vol. 1, 2005, pp. 1-30.

[14] H. Sun, *et al.*, "Research and Implementation of Dynamic Web Services Composition," *APPT*, 2003, pp. 457-466.

[15] V. Tosic, *et al.*, "On Requirements for Ontologies in Management of Web Services," *WES*, 2002, pp. 237-247.

[16] R. Fileto, *et al.*, "POESIA: An Ontological Workflow Approach for Composing Web Services in Agriculture," *VLDB J.*, vol. 12, 2003, pp. 352-367.

[17] V. Tosic, *et al.*, "WSOL - Web Service Offerings Language," *WES*, 2002, pp. 57-67.

[18] W. T. Tsai, *et al.*, "Dependency-Guided Service Composition for User-Centric SOA," *2009 IEEE International Conference on e-Business Engineering*, vol. 0, pp. 149-156.

[19] V. Ernestas and N. Lina, "Transforming Ontology Representation From OWL To Relational Database," *INFORMATION TECHNOLOGY AND CONTROL*, vol. 35, 2006.

[20] D. Bianchini and V. D. Antonellis, "Ontology-based Integration for Sharing Knowledge over the Web," *DiWeb2004 - 3rd International Workshop on Data Integration over the Web*, June 2004.

[21] K. Jong Woo and J. Radhika, "Web Services Composition with Traceability Centered on Dependency," in *In the 38th Hawaii International Conference on System Sciences*, 2005, pp. 89-89.

[22] W. T. Tsai, *et al.*, "Consumer-Centric Service-Oriented Architecture: A New Approach," *Proc. of IEEE 2006 International Workshop on Collaborative Computing, Integration, and Assurance (WCCIA)*, April 2006 pp. 175-180.

[23] M. Chang, *et al.*, "UCSOA: User-Centric Service-Oriented Architecture," *IEEE International Conference on e-Business Engineering*, Oct. 2006, pp. 248-255.

[24] ["http://en.wikipedia.org/wiki/Mashup\\_\(web\\_application\\_hybrid\)."](http://en.wikipedia.org/wiki/Mashup_(web_application_hybrid))

[25] W. Tsai, *et al.*, "Services-Oriented Dynamic Reconfiguration Framework for Dependable Distributed Computing," *COMPSAC*, September 2004, vol. 554-559.

[26] W. T. Tsai, *et al.*, "PSML-S: A Process Specification and Modeling Language for Service Oriented Computing," *The 9th IASTED International Conference on Software Engineering and Applications (SEA)*, Phoenix, November 2005, pp. 160-167.

[27] W. Tsai, *et al.*, "Role-Based Trust Model for Community of Interest," *To be on IEEE International Conference on Service-Oriented Computing and Applications (SOAC'09)*, 2009.

[28] T. Rademakers and J. Dirksen, *Open-Source ESBs in Action*: Manning Publications; illustrated edition edition, October 28, 2008.

[29] Wikipedia, "Cloud computing," [http://en.wikipedia.org/w/index.php?title=Cloud\\_computing&oldid=338511038](http://en.wikipedia.org/w/index.php?title=Cloud_computing&oldid=338511038).

[30] W. Tsai, *et al.*, "Verification framework for dynamic collaborative services in service-oriented architecture," *QSIC*, 2006, pp. 313-320.

[31] Y. Chen and W. Tsai, *Distributed Service-Oriented Software Development*: Kendall Hunt Publishing; 1 edition (June 30, 2008).

[32] W. T. Tsai, *et al.*, "A robust testing framework for verifying web services by completeness and consistency analysis," *IEEE international workshop on service-oriented system engineering (SOSE)*, Beijing, 2005, pp. 151-158.

# Feature Modeling for Service Variability Management in Service-Oriented Architectures

Mohammad Abu-Matar, Hassan Gomaa, Minseong Kim, and Ahmed Elkhodary  
George Mason University  
{mabumata, hgomaa, mkim12, aelkhoda}@gmu.edu

**Abstract** – Service Oriented Architecture (SOA) has emerged as a model for distributed software development that promotes flexible deployment and reuse. Software product lines (SPL) promote reusable application development for product families. Service oriented systems change to respond to changing clients’ requirements. As they change, service oriented systems can be modeled as service families similar to the SPL concept. Some SPL development techniques rely on feature models to describe the commonality and variability of member applications. We use SPL feature modeling techniques to model the variability of service families. In this paper, we introduce a platform independent approach to model SOA variability based on feature modeling. We develop a UML meta-model to describe features. Then, we describe SOA variability scenarios using the newly released OMG standard SoaML. Finally, we develop a meta-model that maps features into SoaML. We believe that such an approach facilitates variability management of service families in a systematic and platform independent way.

## 1. Introduction

Software Product Lines (SPL) are families of software systems that share common functionality, where each member has variable functionality [1]. The main goal of SPL is the rapid development of member systems by using reusable assets from all phases of the development life cycle. This goal is similar to the goal of Service Oriented Architecture (SOA) where flexible application development is a common theme.

An essential modeling phase in Software Product Lines Engineering (SPLE) is Commonality and Variability Analysis (CVA) where the common and varying features of SPL member applications are outlined. CVA is commonly expressed in Feature Models based on the SPL common, optional, and alternative use cases.

Since services in SOA could be used by different clients with varying functionally, we believe that SOA variability modeling can benefit from SPL variability modeling techniques. Service oriented systems can behave like service families, similar to the concept of software product lines.

However, variability modeling in SOA has different challenges to deal with than non-SOA software product lines, because service consumers

are decoupled from services providers and application development is usually done by assembling services rather than developing components and code. Therefore, the following two aspects of variability in SOA must be thoroughly taken into account:

- Variability of consumer services must remain independent from the provider services.
- Service oriented collaboration of multiple enterprises happens in a decentralized and federated manner. This federation, also called choreography, creates a very loose coupling with no central point of authority. Thus, variability of federated enterprises has to be taken into account as well.

Existing approaches to handling variability in SOA [6, 15, 16, 17] have used SPL concepts to model variability in service families (this will be discussed in detail in the Related Work section), however none has produced a treatment of all the variability aspects in SOA mentioned above. In addition, existing research addresses centralized variability without considering choreography, which is decentralized collaboration. In particular, existing research mainly treats SOA variability issues in a platform specific way by focusing on Web Services and orchestration languages like Business Process Execution Language (BPEL).

In this paper, we introduce an approach that addresses all variability aspects mentioned above in a *unified manner*. In particular, our approach handles decentralized as well as centralized variability scenarios in a platform independent way.

In doing so, we follow a platform independent approach by using the newly released OMG standard SoaML [11]. SoaML is a UML extension released by the Object management Group (OMG). SoaML is “a standard way to architect and model SOA solutions using UML” [11]. SoaML introduces new elements that model SOA concepts based on existing UML elements by using the Profile extension mechanism in UML. In this paper, we only use the SoaML modeling elements that are relevant to our current research.

Thus, the key concepts of our approach are as follows, in which we combine SPL variability modeling [1] concepts with SOA concepts, as represented in SoaML:

1. Feature modeling – where we use SPL’s feature modeling techniques to model variability on a meta-modeling level for services.
2. Choreography variability – where we exploit UML and SoaML’s support for Contracts and SPL feature and variability modeling to model the variability of collaborating entities in decentralized environments.
3. Orchestration variability – where we exploit feature modeling to model the variability of service coordination and relate this to SoaML support for workflows.
4. Interface Variability – where we exploit SPL feature modeling and variable interface modeling [1] to model the variability of SoaML service interfaces.
5. Implementation Variability – where we use SPL feature modeling and class modeling variability [1] to model the variability of SoaML service implementations.

We use UML extension mechanism (UML Profiles) to model feature models based on our previous work [12]. Then, we describe SOA variability scenarios using the newly released OMG standard SoaML. Finally, we develop a meta-model that maps SPL feature and variability modeling to SoaML.

The rest of the paper is structured as follows. In section 2, we present a motivating example. Feature oriented service variability, and the feature

to service mapping meta-model are discussed in section 3 and 4. We detail related work in section 5, and conclude the paper in section 6.

## 2. Motivation

In a typical SOA environment, each community has a central body or federation in charge of defining business contracts and policies. This process encompasses defining federation-level business contracts as well as organization-level services. For example, the *E-Commerce B2B Network* (EBN) in Figure 1, a central authority typically defines business-to-business contracts (**Service Contracts**) and business services (**Service Interfaces**) required in order to conduct commercial activities. Each *Service Contract* prescribes generic roles for the organizations participating in it (**Participants**).

For example, in Figure 1a, an organization that aims to play the *Seller* role in the *Purchasing* supply chain must be able to implement and advertize the service interface *Ordering Service*. In this way, each organization defines its own business processes while satisfying federation-wide business contracts.

The variability of the *EBN* can take place at either the federation-level or the organization-level. This can be manifested by introducing new Service Contracts, Service Interfaces, and/or Participants. We address the following four variability scenarios:

- Service Contract (Choreography) Variability - Some contracts may be introduced to the architecture to satisfy the needs of particular realizations of the supply chain. For example, in Figure 1a, the Credit Checking service

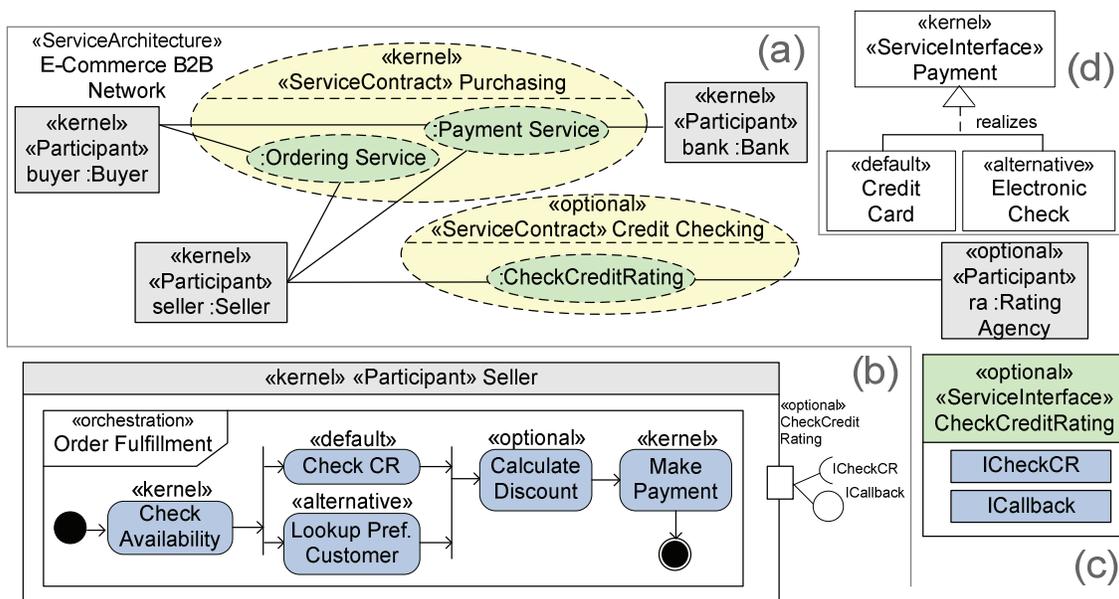


Figure 1: The e-Commerce Motivating Example

contract is introduced, if EBN is operating within a zone of non-trusted buyers and sellers.

- **Orchestration Variability** – A business process is composed of a sequence of activities, and this sequence is called an *orchestration*. Participants may need to change their internal business processes without affecting other participants in the contract. For instance, a *Seller* in EBN may be able to provide some discounts to its buyers to beat the competition. To do so, the *Order Fulfillment* orchestration, as shown in Figure 1b, must introduce a new *Calculate Discount* activity to the workflow.
- **Participants Interface Variability** – New service interfaces may be introduced to a participant so that it can be involved in new choreographies. For example, by introducing a *CheckCreditRating* service interface in Figure 1c, a *Seller* can participate in a *Credit Checking* contract in Figure 1a.
- **Service Implementation Variability** – An application can be configured to have one or more payment options. In EBN for example, the *Seller*'s infrastructure can allow more than one payment option as given by the *Make Payment* service, in Figure 1d, which is specialized to allow *Direct Deposit* or *Credit Card* implementation logic. A given *Seller* may be configured to support either or both of these options, with *Credit Card* as default.

### 3. Feature Oriented Service Variability

In the proposed approach, through feature modeling, we can analyze and model commonality and variability in service families with respect to the variability of services.

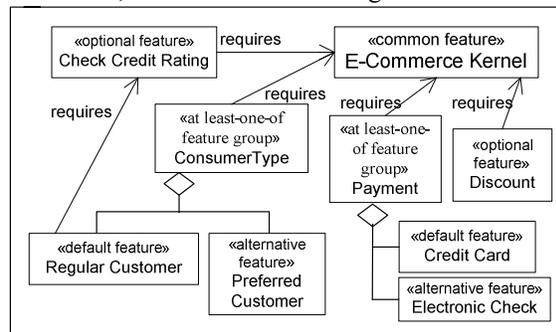
Feature modeling is rooted in the seminal work of Kang et al. [5] in the Feature Oriented Domain Analysis (FODA) method. It is the activity of identifying externally visible characteristics of products in terms of features and organizing them into a feature model. In a software product line approach, feature models are used to express and manage similarities and differences among different family members in a product line. Features are analyzed and categorized as common, optional, or alternative. Common features among products in a product line are mandatory or kernel features, while different features among them may be optional and alternative. Related features can be grouped into feature groups, which constrain how features are used by a product of a product line. A feature group is specialized to “zero-or-more-of”, “at-least-one-of”, “exactly-one-of”, or “zero-or-one-of” [1].

The commonalities and differences between members in a service family are therefore expressed in terms of features. A feature is a reusable requirement that can be selected by any member of the service family.

Referring to our running example, after initial analysis, it was determined that the B2B Network could vary in the following ways:

- The B2B Network could have a credit checking service contract for regular customers, which would then require checking credit rating; however, if the business always deals with trusted buyers, this capability is not required. This requirement can be modeled as a ‘Check Credit Rating’ <<optional feature>>.
- The Order Fulfillment Process could offer two types of payments: credit card, which is default, and electronic check. This requirement can be modeled as a ‘Payment’ <<at least-one-of feature group>> with a ‘Credit card’, which is a default feature stereotyped with <<default feature>>, and an ‘Electronic check’, which is an optional feature, <<optional feature>> in the feature model.
- The Seller Participant could offer a discount capability that can be selected seasonally. This requirement can be modeled as a “Discount” feature, which has the <<optional feature>> stereotype.
- The Order Fulfillment process could offer a ‘Preferred Customer’ capability for customers with existing credit records to speed up their order processing. This requirement can be modeled as a ‘Consumer Type’ <<at least-one-of feature group>> with a ‘Preferred Customer’ <<alternative feature>> and a ‘Regular Customer’ default feature in the feature model.

Based on the aforementioned variability scenarios, we create the following feature model:



**Figure 2: The E-Commerce Feature Model**

The following sub-sections describe how feature-oriented service variability has been achieved in SoaML via meta-modeling.

### 3.1 SoaML Meta-Model

Meta-modeling is a way to define models [3]. A meta-model defines what elements can exist in a modeling language, e.g. the UML meta-model defines concepts like Classes, Packages, and Associations [2]. In addition, the meta-model describes the relations among the modeling elements in the language. All elements in a modeling language are defined in a meta-model for that language.

Before we describe variability in the SoaML meta-model, we briefly define the relevant SoaML elements that have been extended to support variability:

- *ServiceArchitecture* – Specifies a community of Participants who collaborate together to achieve some goals. This element extends the UML *Collaboration* element.
- *ServiceContract* – Specifies the agreement between providers and consumers, and may include interfaces, policies and role descriptions. This element extends the UML *Collaboration* element.
- *ServiceInterface* – Models the service interface concept in SOA. It specifies provided and required interfaces by participants in addition to the interaction protocol that describes how provided and required interfaces should be called. This element extends the UML *Class* element.
- *Participant* – Specifies providers or consumers of services. This element extends the UML *Component* and *Class* elements.

### 3.2 Variability Meta-Model

Since neither UML nor SoaML has native support for variability modeling, we use a UML based feature meta-model based on [12]. The variability meta-model (Figure 3) specifies how the different meta-classes relate to each other. The meta-classes modeled include service related meta-classes such as service contract, service interface, and participant, as well as SPL meta-classes, in particular the feature meta-class. Each of these meta-classes is specialized using SPL concepts into kernel, optional, and variant meta-classes. The top right side of Figure 3 below depicts a feature meta-model. Features are specialized into kernel, optional, alternative, and default depending on the characteristics of the requirements, that is, commonality and variability.

Kernel features are requirements common to all members of systems, that is, required by all members of a product line. Optional features are required by only some members of a product line.

An alternative feature is an alternative of a kernel or optional feature to meet a specific requirement of some systems. Default features are chosen by default when they are part of a feature group. Feature groups refer to constraints on the selection of a group of features (e.g., preventing selection of mutually exclusive features). Feature dependencies represent relationships between features.

In the following section, we map feature models into SoaML models at the meta-level to cope with variability in service families.

## 4. Feature to Service Mapping

In this section, we provide meta-modeling mapping between a feature model and SoaML. In other words, we establish mapping between feature meta-model elements and the relevant SoaML meta-model elements based on the service variability scenarios discussed in the previous section.

### 4.1 Service Contract Feature Mapping

In Figure 3, a *Feature* maps to a set of *ServiceContracts*. For example, when feature *Check Credit Rating* is selected, service contract *Credit Checking* will be activated and enforced in the community. The variability stereotype on a *ServiceContract* dictates the type of feature it may map to. For instance, an optional feature (e.g., *Check Credit Rating*) can only contain optional service contracts (e.g., *Credit Checking* service contract). Similarly, an alternative feature may contain *Variant* service contracts only).

### 4.2 Service Orchestration Feature Mapping

In Figure 3, *Feature* maps to a set of *Activities* in the Participant's orchestration. For example, when the *Discount* optional feature is selected in EBN, which means that the system changes to provide the 'Discount' capability, the *Calculate Discount* activity becomes part of the *Order Fulfillment* process. Thus, the *Discount* <<optional feature>> is mapped to <<optional>> *Calculate Discount* activity.

### 4.3 Service Interface Feature Mapping

In Figure 3, a *Feature* maps to a set of *ServiceInterfaces*. For example, if the *Check Credit Rating* optional feature is selected, the *Seller* participant has to provide a new interface that can interact with a credit rating agency. Thus, the *Check Credit Rating* << optional feature>> is mapped to <<optional>> *Check Credit Rating* *ServiceInterface*.

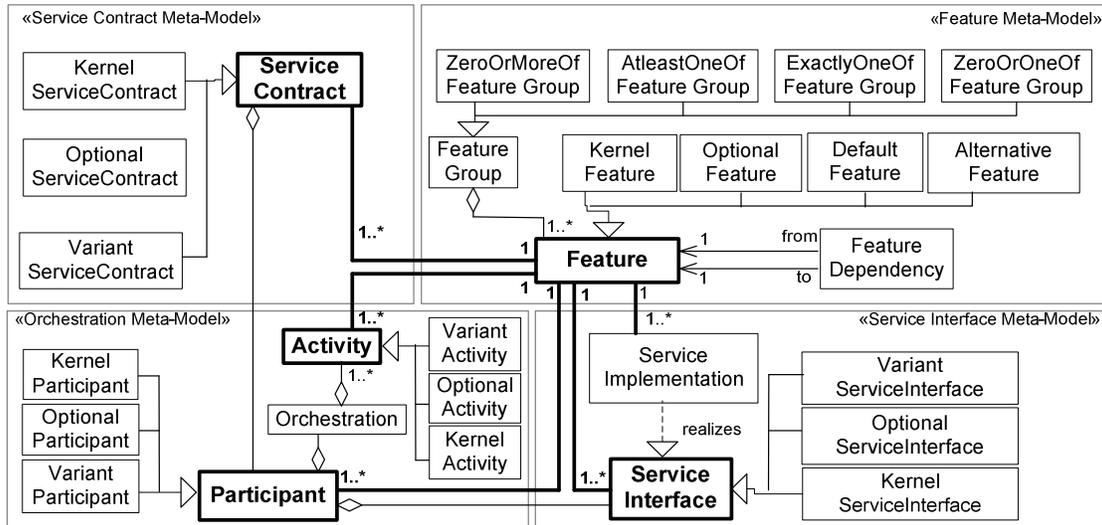


Figure 3: The Feature to Service Mapping Meta-Model

#### 4.4 Service Implementation Feature Mapping

In Figure 3, a *Feature* maps to a set of service implementations. In the running example, if the Electronic Check alternative feature is selected, the Payment ServiceInterface would have a different implementation. Thus, the Electronic Check <<alternative feature>> is mapped to <<alternative>> Electronic Check ServiceImplementation.

### 5. Related Work

There have been several approaches for modeling variability in service oriented architectures. The authors in [13] add variability analysis techniques to an existing service oriented analysis and design method (SOAD). Decision tables are used in [13] to record variability types in each phase of the SOAD process. Finally, these tables are used to map business process requirements into service components.

The authors in [14] present architectural pattern approaches to model variation points in Web Services. They point out that the standards used for Web Services have inherent support for variability.

The authors in [15] advocate a lightweight product line engineering approach that has a specific phase for service composition in the product line architecture. They introduce several variation points that can be used to customize the product line during service selection in the application engineering phase. However, the authors in [15] do not tie service selection to the features required in the product line.

The authors in [16] presented a software product line engineering approach based on Web Services. Components in the architecture were modeled using the <<web service>> stereotype to indicate the use of Web Services. UI feature selection determined the selection of Web Services. UML

activity diagrams modeled customization choices based on feature/Web Service interactions.

In [6], the authors used the concept of features to solve variability problems for SOA. The authors used Feature Oriented Programming techniques to modify the base code of services based on requested features. However, the authors' approach assumes the availability of service implementation code, which is not the norm in true SOA scenarios.

The authors in [17] proposed a method to managing variations in SOA systems and keeping services reusable and useful over a long period of time by adapting a feature-oriented product line approach. Yet, they do not consider the relationships between features and services especially with respect to the SOA variability scenarios.

### 6. Conclusion/Discussion

One of the major benefits claimed for SOA is the flexible building of IT solutions that can react to changing business requirements quickly and economically. SOA promises a vision where service providers offer their services based on their expertise and service requesters search and discover these services based on their business needs.

Normally, service providers are decoupled from service requesters, thus requesters and providers change independent of each other. As a result, variability in SOA is more challenging to deal with than in traditional software systems. In addition, the service oriented choreography of multiple enterprises happens in a decentralized manner. This federation, also called choreography, creates a very loose coupling with no central point of authority. Thus, variability of federated enterprises has to be taken into account as well.

In this paper, we introduced an approach that addresses service oriented variability aspects in a unified manner. In particular, our approach aims to handle decentralized as well as centralized variability in a platform independent way. We used software product line feature modeling techniques and SoaML to model SOA variability at a meta-modeling level. Especially, we described the SOA variability scenarios using SoaML.

Finally, we developed a meta-model that maps features to SoaML meta-classes. It should be pointed out that although we described our SPL approach to service features and variability modeling with SoaML meta-classes, our approach could also be used with other service meta-modeling approaches.

We believe that our approach to service feature and variability modeling has several benefits:

- The use of established SPL feature modeling to manage variability in service families.
- Extended SoaML with variability modeling.
- A service could participate in many service family members, thus maximizing reusability.
- Service providers and consumers can change independent of each other.

In our ongoing research, we plan to achieve the following goals:

- Add Participant variability to our aforementioned service variability types.
- Add more details to the Feature-to-Service meta-model mapping coupled with OCL consistency rules.
- Create transformation rules that transform features in feature models to service artifacts in SOA environments. By following an MDA approach, our variable service model represents the Platform Independent Model (PIM). In essence, we will have two kinds of transformation definitions. The first transformation definition would create the service member application from the platform independent SOA model. The second transformation definition would transform the derived service member application into an SOA platform specific model (PSM).
- Tool support to manifest our approach.

## 7. References

- [1] Gooma, H. 2005. Designing Software Product Lines with UML. Boston: Addison-Wesley.
- [2] UML 2.0 Superstructure document, <http://www.omg.org/cgi-bin/doc?formal/05-07-04>, October 2006.
- [3] Warmer, J., Kleppe, A., 2003. The Object Constraint Language Second Edition Getting Your Models Ready For MDA. Addison-Wesley.
- [4] David S. Frankel 2003. Model Driven Architecture Applying MDA to Enterprise Computing. OMG Press
- [5] Kang, K, S. Cohen, et al. *Feature-Oriented Domain Analysis (FODA) Feasibility Study*. Technical report CMU/SEI-90-TR-21, Software Engineering Institute, Pittsburgh A, November 1990.
- [6] S. Apel, C. Kaestner, and C. Lengauer, "Research challenges in the tension between features and services," in *SDSOA '08: Proceedings of the 2nd international workshop on Systems development in SOA environments*. ACM, 2008, pp. 53-58.
- [7] J. Gurf, J. Bosch, and M. Svahnberg, "On the Notion of Variability in Software Product Lines," Proc. of the Working IEEE/IFIP Conf. on Software Architecture (WICSA'01), 2001.
- [8] J. Coplien, D. Hoffman, and D. Weiss, "Commonality and Variability in Software Engineering," IEEE Software, Vol. 15, No. 6, Nov/Dec 1998, pp. 37-45.
- [9] I. Jacobson, M. Griss, and P. Jonsson, *Software Reuse: Architecture, Process and Organization for Business Success*, Addison Wesley Longman, 1997.
- [10] Hyun Jung La et al., "Practical Methods for Adapting Services Using Enterprise Service Bus," Proc. 7th International Conference (ICWE 2007), LNCS 4607, Springer-Verlag, 2007, pp. 53-58.
- [11] SoaML Beta 1 document, <http://www.omg.org/spec/SoaML/1.0/Beta1/>, April 2009
- [12] H. Gooma and M.E. Shin, Multiple-View Modeling and Meta-Modeling of Software Product Lines, Journal of IET Software, Volume 2, Issue 2, Pages 94-122, Published by Institution of Engineering and Technology, April 2008.
- [13] Soo Ho Chang, Soo Dong Kim, "A Service-Oriented Analysis and Design Approach to Developing Adaptable Services," pp. 204-211, IEEE International Conference on Services Computing (SCC 2007), 2007
- [14] N. Y. Topaloglu, R. Capilla, "Modeling the Variability of Web Services from a Pattern Point of View". European Conference on Web Services (ECOWS2004), LNCS Springer-Verlag, 2004, pp. 128-138.
- [15] Rafael Capilla, N. Yasemin Topaloglu, "Product Lines for Supporting the Composition and Evolution of Service Oriented Applications," pp. 53-56, Eighth International Workshop on Principles of Software Evolution (IWPSE'05), 2005
- [16] H. Gooma and M. Saleh, Feature Driven Dynamic Customization of Software Product Lines, Proc. 9th International Conference on Software Reuse, Springer Verlag LNCS 4039, Pages 58-72, Torino, Italy, June 2006
- [17] J. Lee, D. Muthig, M. Naab, M. Kim, and S. Park, "Identifying and Specifying Reusable Services of Service Centric Systems through Product Line Technology", Proc. of Service Oriented Architectures and Product Lines - What is the Connection? (SOAPL - 07), CMU/SEI-2008-SR-006, May 2008.

# TestDrive – A Cost-effective Way to Create and Maintain Test Scripts for Web Applications

Sachin Patel, Priya Gupta, Prafullakumar Surve

*Tata Consultancy Services, Innovation lab - TRDDC,  
54B Hadapsar Industrial Estate, Pune 411013, India*

{sachin.patel, priya1.g, prafullakumar.surve}@tcs.com

**Abstract** — A large number of web applications are maintained and tested across the industry. Most of these change frequently and require a regression test suite to be run before major releases. As this is a repeated task, testers wish to automate it. We conducted some surveys and found that numerous projects had attempted an automation exercise and faced problems such as lot of time spent in script maintenance and lack of skills to automate.

In this paper, we propose a model-based approach that works in conjunction with existing tools and techniques for automation. Our approach includes a recorder to record tests into a test model and generators capable of generating test scripts into popular architectures and platforms used in practice.

We show through a study that our approach and tool took 30% lesser time to maintain scripts as compared to existing methods.

**Keywords** — Testing, Test automation, Selenium, Test Script Generation, Model-based

## I. INTRODUCTION

When systems are in maintenance, there exist a set of regression test cases which testers like to execute before every major release. Very often, the deadlines for these releases are pre-decided, thus limiting the time available for the maintenance cycle. Regression testing, being one of the last activities in the maintenance cycle, tends to be compromised when there is shortage of time. Since the regression tests are test cases already identified, they are a good candidate for automation. Automating such mundane tasks also leaves more time for the testers to focus on identifying new test cases for the changes under test, and hence deliver better quality.

Test Automation has its own set of challenges. Systems continuously change; hence keeping the automation scripts up to date becomes an overhead. We observe numerous instances where automation scripts are abandoned because maintaining them consumes too much of effort. Developing the scripts also tends to be a long run effort as project teams have to find time for automation along with working on the pending maintenance requests. Further to this, the team may not possess the skills required for test automation.

We aim to provide an approach for test automation that can work well within these constraints, namely: frequent changes, shortage of time and insufficient skills.

We have developed a model-based approach for test creation and maintenance. Our test-model consists of test scenarios, test inputs, screen structures and assertions. This

model is automatically populated by a recorder that extracts required information from the application at runtime. Asserts can be added by the tester during the recording process. The model is then used to generate the test scripts. When the application under test changes, the testers can change the model and regenerate the scripts. An interface is provided to edit the test-model. This approach helps in faster changing of UI and data, as changes have to be done in only one place.

We have implemented this approach in our tool TestDrive which has been built using open source tools Selenium and robotframework. We conducted an empirical study of the first version of the tool and found maintaining test scripts created using TestDrive takes 30 percent less time. In this paper, we have described a newer version of the tool that was enhanced based on findings from the study.

## II. RELATED WORK

Most of the research in Model Based Testing (MBT) has been around modelling system requirements and generating test cases from the model. Some of the model forms that have been discussed are UML models [5], Domain Specific Languages [12] and FSMs [13]. UML State charts, class and sequence diagrams are the models most often used [6].

Model-based approaches have been mostly applied for System Testing but there have been few attempts to apply it to regression testing [6]. These approaches are better suited to generate test cases rather than executing them as they use design level models. Test models need to be more closely tied to implementation, in order to achieve automated execution. Our approach enables this by extracting the test model at runtime using a proxy recorder.

Different MBT approaches require different skills and these are needed ahead of usage. The main issue is how to minimize the human skill required and simultaneously maximize the automation level [6]? Our approach addresses this issue by automatically extracting the test-model and then allowing the tester to refine it.

A more closely related approach is GUITAR [4] where an event flow graph of the system is extracted automatically and used to generate and execute test cases. Our approach differentiates itself by being better suited for business applications where testers would like complete control over specifying flows / input data, and so on.

More discussions on different approaches to extract the GUI model can be found in [3] and [7]. The techniques help to

minimize the efforts required to make test script changes when there are changes in the GUI. Our approach aims further, at desensitizing tests from changes not only in the GUI, but also in test data and flows.

### III. BACKGROUND

Many tools and frameworks are available for test automation. For example, QTP, Rational FT, and Selenium. Testers can choose one such tool and use either of the following methods to automate:

- 1) Record and Payback is a quick method to create scripts. Scripts created by this method tend to be difficult to maintain since they are not organized for maintainability.
- 2) Programming the scripts in a modular manner seems better in order to make them easy to maintain. Testers choose one of the following architectures for their test scripts.
  - Keyword-driven frameworks aim at localizing changes by creating test libraries of commonly performed user actions (e.g. login, search transactions). These are used to create bigger and more complex tests.
  - Data-driven frameworks allow test inputs to be parameterized, that is, converted to variables and read from data tables or data files. These are useful when data changes are frequently required.
  - Hybrid frameworks are a combination of the above two and something that most projects need. This is because both, the application and test data undergo frequent change.

This method requires significant effort and skills for script development. Our approach aims to provide the tester the best of both the approaches i.e. quick creation and maintainability.

### IV. APPROACH

The central idea behind the approach is that testers should specify/model tests without worrying about how to automate them. However, specifying the tests should not take significant time/skill.

The test-model has sub-sections for containing UI structures, test scenarios, test data, and test cases. We provide a recorder which populates the model by capturing the necessary information from the application at run-time. This allows the tester to create the model with the ease and speed of the record-playback method. Moreover this is possible without having to learn the model structure.

A script generator reads the model and generates test scripts into architectures of the tester's choice. The generator structures the output scripts into a modular form for maintainability. For example, it automatically identifies all the UI actions and related inputs for each screen and generates a reusable script for that action. Thus the tester gets an action library that can be used to build tests very quickly. For example, you can have a ready function to Login, which inputs the username, password and clicks Submit. Any test case which requires to Login may just call this function.

The generator provides options to configure the generated test script architecture. The appropriate test suite architecture

depends on the dynamics of the application. In an application with many and frequently changing scenarios, a keyword-driven architecture may suit, whereas in an application with fewer scenarios but a large variety of input classes, a data-driven architecture may be better. Our approach allows the tester to make such a choice without any new script development effort.

Our approach has the following advantages 1) Tool Independence 2) Frees the tester from low level scripting details 3) Ability to generate scripts into multiple possible architectures/frameworks/tools/environments 4) Easy integration with other tools 5) No need for tool/automation expertise 6) Possibility of using the model for new test case generation.

One of the constraints that the users of this method would face is the occasional need for manual changes in scripts. There might be special cases not handled by our generator for which the tester wishes to change the scripts directly. This can be handled by implementing a generator capable of preserving changes or providing a facility to reverse engineer the model from modified scripts.

### V. IMPLEMENTATION

We have developed a tool named TestDrive to try out this approach. This section contains details about the tool implementation.

#### A. The Test Model

The Test Model is an amalgamation of two Object Management Group (OMG) standards, namely, UML 2.0 Testing Profile (U2TP) [1] and Knowledge Discovery Meta-model (KDM) [2].

The high level structure of our model resembles U2TP. We have implemented concepts from the Test Architecture, Test Behaviour and Test Data sections. We have not implemented Time Concepts in the current version. The U2TP does not provide means to specify details about the interface of the System under Test (SUT). As deemed necessary for our implementation we have added a Test UI section to model the GUI structure of the SUT.

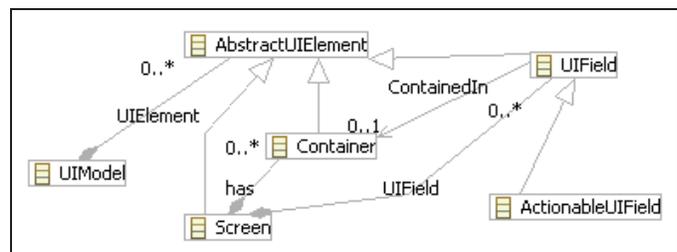


Figure 1: GUI section of the Test Model

The granular details such as the interactions model in the behaviour section, the structure of the data pool in the test data section, and UI model in the test UI section are adapted from relevant packages of the KDM.

The GUI model (Figure 1) is organized as a hierarchical structure where screens have container, and containers have UI fields or Actionable UI fields. The Test Behaviour section



No	Measure	CT-N <sup>1</sup>	TD-E <sup>2</sup>	TD-N <sup>3</sup>
1	Total script creation time	187.25	124.37	182.62
	Short Scenarios, Simple Asserts	35	25	32
	Complex Asserts or Long Scenarios	43.75	62.5	70.62
	Complex Asserts & Scenarios with Multiple Datasets	89.5	55.62	80
2	Number of support requests	6	5	14

#### Maintenance

No	Measure	CT-N	TD-E	TD-N
1	Total time needed for changes	255	167	181
	UI change, impacting many screens	-	90	70
	New field impacting 3 tests	85	35	-
	New input added to existing test	30	15	15
	New menu option	30	10	16
	New column in a list	75	-	65
	More datasets to an existing test	35	17	15
2	Number of support requests	0	0	0

#### 3) Summary of Findings

The setup process for TestDrive was quicker as compared to that of CT and also took very less space.

The script development effort was approximately same. One of the teams using TestDrive could develop scripts in 34% less time than others. The script development effort using TestDrive could have been reduced further by implementing the feature of data-driven testing in TestDrive. There were four test cases which had to be executed for multiple data sets. The recent version of TestDrive has this feature.

The script maintenance effort for both the teams using TestDrive was approximately same and at least 30% less than that taken by the CT team. The reasons for this could be attributed to 1) The model editor that does not require scripting or modeling expertise 2) localizing of changes for e.g. a field promotion code was removed from some screens and was expected to impact 5 test cases. The TestDrive users could do this simply by removing the field from the UI Model.

Participants mentioned that test maintenance was very easy with TestDrive and they could do it without any scripting knowledge. There were some suggestions for improving the tool, such as, control of execution speed, handling queries returning multiple rows, and so on.

#### 4) Threats to Validity

A threat to validity is any factor that influences the results of the experiment. Here are some of those we noticed: 1) It is possible that the test cases developed for jbilling did not resemble those written by testers in a real testing assignment. In a larger implementation, we might come across cases where our approach does not work well. 2) The difference in time taken by the two teams is not significant enough to infer that the approach is effective. The effectiveness of the approach would only be known after some sizeable implementations in real world projects.

<sup>1</sup> Novices using the Commercial Tool

<sup>2</sup> Experienced participants using TestDrive

<sup>3</sup> Novices using TestDrive

## VII. CONCLUSIONS AND FUTURE WORK

In this paper we presented a model-based approach to generate test scripts and the tool TestDrive that implements it.

In an empirical evaluation, we observed that users found TestDrive easy to use and it took lesser time to create and maintain test scripts as compared to existing methods. The time savings as compared to existing methods were to the order of 30%.

The model-based approach presents us numerous opportunities to help improve test quality along with serving the basic need of test automation. For example, when a screen changes, it should be possible to automatically identify and correct the impacted test cases. Another possibility is to assist the tester in identifying un-tested system flows by building a scenario graph. Similarly, state-transition diagrams could be generated by monitoring states for entities of the tester's interest. This will help the tester identify untested states or transitions. For example, a tester should be able to see how the stock order flow happens from order placement to settlement. Since the model represents a black box view, we should be able to devise some black-box coverage measures such as UI fields used/unused and scenario coverage. Such measures are likely to be more acceptable to testers who do not understand the application code.

We believe that the model based approach to automation of test execution holds great potential in helping testers automate and improve their test cases. Multiple project teams in our organization are now using TestDrive and we shall continue to observe the usage/results and refine our approach.

## REFERENCES

- [1] UML Testing Profile  
[http://www.omg.org/technology/documents/formal/test\\_profile.htm](http://www.omg.org/technology/documents/formal/test_profile.htm)
- [2] Knowledge Discovery Meta model  
<http://www.omg.org/technology/kdm/index.htm>
- [3] Mark Grechanik, Qing Xie, and Chen Fu. *Maintaining and Evolving GUI-Directed Test Scripts* In Proceedings of the ICSE 2009
- [4] Atif M. Memon, *An event-flow model of GUI-based applications for testing* In STVR, Volume 17, September 2007
- [5] Marlon Vieira, Johanne Leduc, Bill Hasling, Rajesh Subramanian, Juergen Kazmeier. *Automation of GUI testing using a model-driven approach*. In Proceedings of Workshop on automation of software testing, ICSE 2006
- [6] Arilo C. Dias Neto, Rajesh Subramanian, Marlon Vieira, Guilherme H. Travassos. *A survey on model-based testing approaches: a systematic review* International workshop on Empirical assessment of software engineering languages and technologies, ASE 2007
- [7] Christof J. Budnik, Fevzi Belli, Axel Hollmann, *Structural Feature Extraction for GUI Test Enhancement*, pp.255-262, ICST 2009
- [8] Cem Kaner, James Bach, Bret Pettichord. *Lessons Learned in Software Testing: A Context-Driven Approach*
- [9] Cem Kaner, *Architectures of test automation*. In STAR West, 2000.
- [10] Pettichord, B. *Seven Steps to Test Automation Success*. In STAR West, San Jose, Nov 1999
- [11] Buwalda Hans. *Automated testing with Action Words*. In STAR West, 1998
- [12] Sinha, A. Smidts, C. *HOTTest: A Model-Based Test Design Technique for Enhanced Testing of Domain-Specific Applications* In ACM TOSEM 2006, VOL 15; NUMB 3, pages 242-278
- [13] Fevzi Belli, *Finite-State Testing and Analysis of Graphical User Interfaces*, In ISSRE'01

# Ontology-Based Test Case Generation For Simulating Complex Production Automation Systems

Thomas Moser, Gregor Dürr and Stefan Biffel

Christian Doppler Laboratory for  
Software Engineering Integration for Flexible Automation Systems  
Vienna University of Technology, Austria  
{thomas.moser, gregor.duerr, stefan.biffel}@tuwien.ac.at

*Abstract*—The behavior of complex production automation systems is hard to predict, therefore simulation is used to study the likely system behavior. However, in a real-world system many parameter variants need to be tested with limited resources. Therefore, test cases need to be generated in a systematic way to find suitable scenarios efficiently. This paper investigates the effort of two approaches for providing test cases based on available testing knowledge. The traditional approach uses a static generator script based on implicit testing knowledge, which takes significant effort to add new parameters. The innovative approach uses a dynamic generic generator script based on an ontology data model of the testing knowledge. We empirically evaluate these approaches with a use case from the production automation domain. Major result is that the high-level test description of the ontology-based approach takes more initial effort for setup, but increases the usability and reduces the risk of errors during the test case generation process.

*Keywords* - test case generation, ontology, production automation simulation, explicit testing knowledge.

## I. INTRODUCTION

Production automation systems are often complex as the behavior of the overall system cannot easily be predicted from the behavior of the subsystems. Therefore, simulation is used to study the behavior of complex production automation systems. In addition to simulation accuracy the simulation system performance is an important issue, particularly if many parameters and value ranges for system behavior need to be evaluated systematically [6]. Example test parameters for assembly lines in production automation are scheduling strategy, failure handling strategy, and the number of products.

Software testing investigates the quality of the product or service under test. In order to achieve sufficient test coverage for the requirements of an application, there must be at least one test case for each requirement and relevant parameter setting, which can take considerable effort. A major goal is to generate test cases in a systematic way to efficiently find suitable scenarios for most of the requirements with limited testing resources. In this paper, the test cases define input data to simulate complex distributed assembly line systems with a simulation tool [7, 8, 13].

This paper presents an approach for extracting expert testing knowledge into an ontology and for using this explicit knowledge to efficiently generate test cases for a production automation simulator [7, 8, 13]. For this purpose, the test coverage combined with the cost to achieve this test coverage is used as performance metric. In our context, test coverage is the ratio between the number of generated test case scenarios and the number of all possible test case scenarios regarding a given set of parameters respectively their possible parameter values.

This paper investigates two approaches for providing test cases. The traditional approach implements a test as a static generator script, which is fairly simple and quick to start up, but takes significant effort to add new parameters, since all existing scripts need to be updated to accommodate the new parameter settings. In addition, the tester needs programming skills both for setting and modifying parameters.

The innovative approach models system and test know-how in an ontology-based data model and dynamically adapts a generic generator script according to the settings in the test data model. Test cases are generated and run with respect to the parameters chosen by the user. Important advantages of this approach are (a) available efficient tool support for modifying ontologies in case of changes to the parameters and/or structure of the system under test and (b) the fact that the generator script is not affected by the modification. Therefore, even testers without programming skills can add new parameters by modifying the underlying data model. Test cases are generated from the ontology and exported as XML file. The implemented generator script and the ontology are loosely coupled. Therefore, changes to the ontology do not necessarily lead to changes of the generation script. This design approach enables a flexible and high-level test description.

The evaluation part explores how the ontology-based approach can reduce the cost for test description, enables a more efficient and effective generation of test cases while aiming at an exact definition of test coverage to be achieved, and improves the changeability of the test case generation approach (e.g., the introduction of new test parameters) due to lower efforts to implement new test case parameters.

The remainder of this paper is structured as follows: Section 2 summarizes related work on software testing, production automation simulation, and on ontology-based test case generation. Section 3 identifies the research issues and summarizes the use case. Section 4 introduces the ontology-based test case generation approach, while section 5 presents the evaluation results. Section 6 discusses the evaluation results with regard to the research issues, and finally section 7 concludes the paper and identifies further work.

## II. RELATED WORK

This section presents related work on automated software testing. Furthermore, the production automation simulation systems MAST and SAW are shortly introduced. Finally, related work on ontologies and on ontology-based test case generation is summarized.

### A. Automated Software Testing

Testing of software is an important part of every software life cycle to demonstrate the capability of the software and identify defects before operational use. The National Institute of Standards and Technology (NIST) reported in a 2002 study that software bugs cost the U.S. economy around 60 billion dollar per year [10]. The same study showed that these costs could be reduced by more than a third by improving testing. In the IEEE Standard on the Software Life Cycle Process, a process is defined as a set of activities; process groups represent a higher level of abstraction [1].

A test process can be seen as the systematic execution of a software program. Test management and running the test object with specific data are the important parts of the test process. The goal of the test management is to plan, execute, and analyze the test run. A test run consists of one or more test cases [11].

### B. Production Automation Simulation

The Manufacturing Agent Simulation Tool (MAST) is an agent-based solution for some typical manufacturing tasks by using multi-agent technologies for manufacturing control [13]. Flexible distributed manufacturing systems can be modeled as intelligent, autonomous and cooperative agents [12], where each agent manages the local behavior to meet goals locally without centralized control [13]. In addition, the agents interact to achieve the system goal cooperatively.

The Simulation of Assembly Workshop (SAW) [7, 8], which is an extension of the original MAST system, investigates processes, methods, and tool support for planning, coordination, simulation, and lab tests for work shifts in an assembly workshop. Generally, a workshop aims to effectively and efficiently carry out the work orders in a work shift. SAW helps to understand the impact of tactical decisions in the production automation environment. For instance, the user can optimize his assembly line by analyzing the effect of the different strategies. The SAW project provides a simulator that can execute the test cases for the production automation domain. In addition, the logged events during the simulation allow analyzing the simulation object under test. Thus, the results of the simulation enable check-

ing the post condition of the test case. The explained life cycle of a test case is adopted for the simulation process of the SAW project during the elaboration of this work.

### C. Ontologies for test case generation

In general, ontologies are a main part of the semantic web technology and facilitate the knowledge representation of real-world concepts. Ontologies are formal models of a specific application domain, and primarily used to facilitate the exchange and partitioning of knowledge. More precisely, an ontology is a data model that represents a set of concepts within a domain and their relationships. The word ontology has its origin from the Greek words *ontos* (=being) and *logos* (=word). From a philosophical point of view an ontology refers to the subject of existence, that is the study of being as such [3]. Gruber [4] defines an ontology as an explicit specification of a conceptualization. Where a conceptualization illustrates an abstract, simplified picture of the world used for representation and designation. Each knowledge representation follows a certain degree of conceptualization, either explicitly or implicitly. Moreover, ontologies can effectively support software development processes by providing a continuous data model [2].

Ontologies could help generate basic test cases since they encode domain knowledge in a machine processable format. A simple example for this would be regarding cardinality constraints. Since those constraints define restrictions on the association of certain classes, they can be used to derive equivalency classes for testing [5]. Ontologies may not be the first candidate for such a scenario, since there are formalisms like OCL that are specialized for such tasks. However, once domain knowledge is available in an ontology format, it might be feasible to reuse that knowledge. Nguyen et al. [9] describe a framework for automated test case generation in the context of multi-agent systems. They use agent interaction ontologies that define content semantics of agent interactions to generate test inputs, guide the exploration of the input space during test case generation, and verify messages exchanged between agents with respect to the agent interaction ontology.

## III. RESEARCH ISSUES AND USE CASE

The generation of test cases based on available testing knowledge is an import factor for the simulation of complex systems, such as production automation systems. This paper investigates two approaches for providing test cases. A traditional approach based on manually derived static generator scripts, which takes significant effort to add new parameters. In addition, the users need programming skills for both setting and modifying parameters. The ontology-based approach uses a dynamic generic generator script based on an ontology as data model. Test cases are generated with respect to the chosen parameters by the user.

This work proposes an ontology as suitable data model to provide the necessary data to generate a suite of test cases with an ontology-based approach for a given set of parameters. This claim is also met by the static approach. In other words, the fact that the ontology-based approach is newer

and works as well as the old static one is not enough motivation for change. However, measurable benefits are essential to accept the new ontology-based approach and the change costs. As with every new engineering method an important question is what aspects of the new approach are comparable to or better than a best-practice traditional approach. Therefore, we derive the following research issues.

**RI-1. Feasibility of the ontology-based test case generation approach.** The ontology-based approach aims at overcoming limitations of the traditional static approach. Firstly, the new approach should provide a high-level test description to allow the target audience to generate test cases with less effort. Secondly, a validation check and consistency check of the parameter setting is essential to reduce the risk of making mistakes during the configuration phase of the test case generation process.

**RI-2. Cost-benefit potential of ontology-based test case generation approach.** The following two sub research issues define a metric how the cost-saving potential can be measured with respect to the effort for test description and the effort for setting up the solution. RI-2a assumes that both the number of parameters to choose and the number of supported data types are constant, while RI-2b addresses the cost for adding new parameters with different data types.

#### IV. ONTOLOGY-BASED TEST CASE GENERATION APPROACH

This section describes the ontology-based test case generation approach. In the first subsection, a short overview of the simulation systems underlying ontology data model is given, while the second subsection describes the test case generation suite.

##### A. Simulation System Data Model

The SAW system [7, 8] uses a 5-layer ontology as underlying data model. The five layers of the simulation system and their relationships among each other are illustrated in Figure 1. In the following, each layer is described shortly. The *business layer* prioritizes all incoming orders with respect to their due date. In the *shift layer* a capacity check of the resources needed for producing the ordered product takes place. In addition, the business orders get transformed to work orders. In the *job shop layer* the work orders get broken down to single tasks by the production strategies. Afterwards, the responsible production strategy orders the tasks with respect to their specific criteria. The load balancer in the *operation layer* is responsible for a well-balanced utilization of the available machines. Furthermore, the shortest path consisting of one or more conveyor belts is calculated to fulfill the different tasks. The *master data layer* contains all information, which remains rather constant during the simulation such as the arrangements of the conveyor belts, the arrangements of the machines, and the structure of the product trees. All other layers can refer to this information.

In order to provide the ontology-based test case generation functionality, we extended the original SAW ontology data model by adding the so-called *test case layer* to the ontology data model. The *test case layer* together with a suitable test case generator script provides high quality test cases

in an automated and systematic way. As a consequence high test coverage can be achieved, which is essential for testing the performance of simulation systems. The user can decide if she wants to create test cases manually or with the help of a generator script. In case of using a generator script the user just has to configure the parameter setting which contains the test case parameters and the corresponding set of valid values. Afterwards, the generator script generates the test cases with respect to the chosen parameter setting.

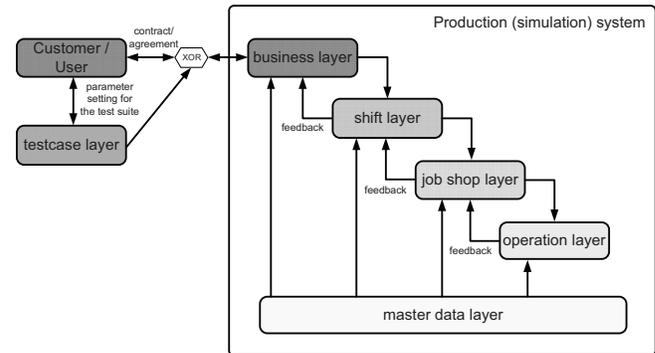


Figure 1. Layers of the SAW systems ontology data model.

##### B. Test Suite Generation

Initially, the ontology-based approach requires a GUI that can be dynamically adapted with regard to the available test case parameters. The ontology-based approach extracts the offered test case parameters, the allowed values for the parameters, and structural information from the underlying data model. Afterwards the identified information of the ontology is passed to the GUI. This data flow makes it possible to build a dynamic GUI which corresponds to the ontology at runtime. As a consequence, modifications to the ontology do not necessarily lead to manual changes neither to the GUI nor to the dynamic generic script.

On the one hand the ontology-based approach uses the underlying ontology's structure and restrictions to ensure the consistency of the test cases. On the other hand the data ranges of the offered parameters are used to ensure the validation of the test cases. Nevertheless, the user has to manage the underlying ontology. Therefore, the user needs skills for modifying the ontology with common graphical editor tools like Protégé<sup>1</sup>. Of course, for modifying an ontology users need some experience but it involves less effort than modifying a hard-coded script. Furthermore, if something goes wrong during the modification of the ontology the user will recognize it immediately in the parameter setting configuration process. On the contrary, the user might realize that something went wrong during the modification of the static script after the simulation run took place by analyzing the simulation results. This is a frustrating experience as users usually do not know what went wrong. In that case the user is captured in a trial and error loop. Surely, these circumstances negatively affect the acceptability of the static approach.

<sup>1</sup> <http://protege.stanford.edu/>

By using the ontology-based approach, the user can always choose from all test case parameters supported by the generator script to define the parameter setting. In addition, the ontology-based approach ensures that only valid and consistent parameter settings can be defined by the user. This circumstance can be achieved with the information of the ontology. Afterwards, the ontology-based script generates test cases based on the parameter setting. At the end of the path the dynamic generic script generates the XML file. A useful characteristic of the “3 Phases Process Model” shown in Figure 2 is the fact that the first phase and the second phase are totally independent of the domain. This is ensured as the domain specification is hidden in the ontology. The first phase communicates with the ontology and passes the element names, valid values for element instances, and information about the structure of the ontology to phase two. After this step the second phase uses the received information about the ontology to build a dynamic GUI at runtime. The user can configure the third phase based on the data model easily.

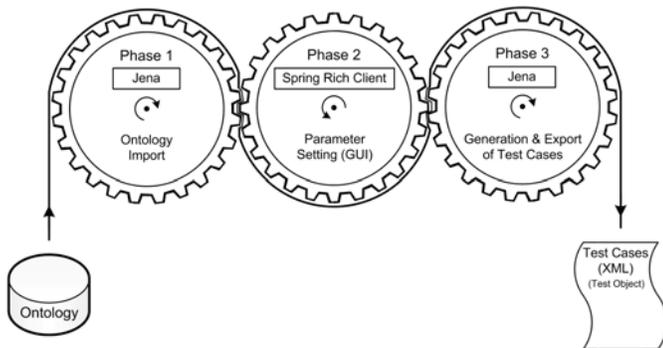


Figure 2. 3 Phases Process Model of the ontology-based approach.

## V. EVALUATION

This section focuses on the evaluation concept and explains in detail the criteria for the different objectives. The objectives are the costs for the test description, the effort for implementing test case parameters, and whether the test coverage is definable by the user.

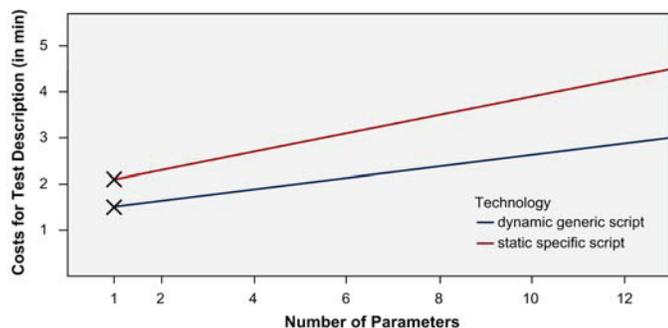


Figure 3. Duration of the test description.

The evaluation compares the traditional static approach and the novel ontology-based approach. Both approaches are test case generator scripts, which aim at providing test cases as input data for a simulation in an automated and structured way. Firstly, the evaluation concept determines the costs for

the test description as configuration duration of the parameter setting. Secondly, the effort for implementing test case parameters is determined with respect to the experience level. Lastly, the Return on Investment (ROI) is calculated assuming experience in the use of the different technologies.

Figure 3 shows the costs for the test description for both generator scripts with respect to the implemented number of parameters. Therefore, the time to configure the test case generation process is measured. For the empirical evaluation, we measured the time needed for test description using 1, 4 and 10 parameters for each approach and extrapolated the results. In order to empirically evaluate the effort needed for additional parameters, we again measured the time needed to add 1, 3 or 5 parameters and then extrapolated the results to be able give evidences for larger use case scenarios. Figure 4 outlines the effort for implementing up to 195 test case parameters. As a result, the higher effort for the first time implementation of the dynamic generic script pays off after the implementation of the 38th test case parameter. Nevertheless, the secondary criteria such as a lower risk for mistakes during the configuration phase of the generation process and ensuring valid and consistent test cases as output of the generation process make the use of the dynamic generic script preferable even for a small number of test case parameters.

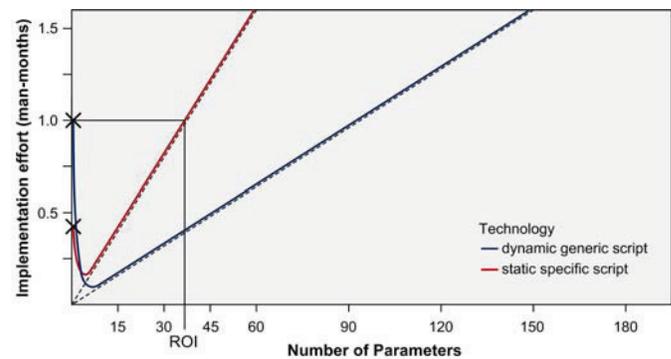


Figure 4. Effort for adding a high number of parameters.

## VI. DISCUSSION

The results of the evaluation are listed in Table 1. As can be seen, the ontology-based approach performs better on most objectives. The ontology-based approach demands a higher effort for the first time implementation. In addition, the user needs to obtain the necessary skills for the used generator script. For using the static script the user needs programming skills for both generating test cases and adding new parameters. The ontology-based script assumes skills in using an ontology editor for adding new test case parameters to the ontology. The ontology-based approach requires no specific skills for generating test cases. The percentage values for the implementation effort are based on the effort interpolation presented in the evaluation section. For the constant parameters scenario, the effort needed for implementation of the static approach is take as base effort (100%) since it is lower than the effort needed for implementing the ontology-based approach; and vice versa for the expandable number of parameters scenario.

Table 1. Results of the Evaluation.

	Ontology-based approach		Static approach	
	Constant Parameters	Expandability of parameters	Constant Parameters	Expandability of parameters
<b>Test Description</b>	high-level	high-level	low-level	low-level
<b>Implementation</b>	260%	100%	100%	250%
<b>Test Coverage</b>	is defineable	is defineable	is not defineable	is not defineable

## VII. CONCLUSION AND FURTHER WORK

High test coverage is essential for testing the performance of simulation systems not only in the production automation domain. Test case generators provide such test cases as input data for the simulation in an automatic and structured way. Many solutions for test case generators use a static approach which is difficult to expand. Mostly, these solutions offer an unsatisfying usability since only a low-level test description is supported. In this work, we discussed the test case generation process and developed an ontology-based approach based on an available static one to achieve high-level test description, lower effort for implementing additional test case parameters, and a definable test coverage.

This paper identified weaknesses of traditional approaches and proposed and evaluated a solution to address these weaknesses. After the implementation of the new approach the effectiveness of both approaches was compared. The ontology-based approach performs very well on most objectives as the result of the evaluation shows". Once the dynamic generic script is running it does not have to be maintained anymore even if the number of test case parameters varies over time.

**Future Work.** Future research will include the feedback of the simulation results into the ontology. An important fact is that the feedback is not limited to the simulation results since the results depend on the test cases and the assembly line. Another future research topic will be the more excessive usage of ontology-based reasoning to support the deduction of test cases. At the moment the ontology is used for building a dynamic GUI at runtime to ensure that the parameter setting is valid and consistent. In addition, the structure of the XML file corresponds to the structure of the ontology. However, the deduction of test cases is probably more efficient than generating test cases as combinatorial possibilities of the parameter setting especially since the ontology is a knowledge-based system and therefore enables to infer from the stored fact base.

## ACKNOWLEDGMENT

This work has been supported by the Christian Doppler Forschungsgesellschaft and the BMWFJ, Austria. This work has been partially funded by the Vienna University of Technology, in the Complex Systems Design and Engineering Lab.

## REFERENCES

- [1] B. Bruegge, and A.H. Dutoit, *Object oriented software engineering*, Prentice Hall, 2009.
- [2] C. Calero, F. Ruiz, and M. Piattini, *Ontologies for Software Engineering and Software Technology*, Springer-Verlag New York Inc, 2006.
- [3] D. Gašević, D. Djurić, V. Devedžić, and B. Selić, *Model driven architecture and ontology development*, Springer-Verlag New York, Inc. Secaucus, NJ, USA, 2006.
- [4] T.R. Gruber, "A translation approach to portable ontology specifications," *Knowledge acquisition*, vol. 5, no. 2, 1993, pp. 199-220.
- [5] H. Knublauch, D. Oberle, P. Tetlow, and E. Wallace, "A semantic web primer for object-oriented software developers," *W3C Working Group Note* <http://www.w3.org/TR/sw-ooosprimer>, 2006.
- [6] A. Lüder, J. Peschke, T. Sauter, S. Deter, and D. Diep, "Distributed intelligence for plant automation based on multi-agent systems: the PABADIS approach," *Production Planning and Control*, vol. 15, no. 2, 2004, pp. 201-212.
- [7] M. Merdan, T. Moser, D. Wahyudin, and S. Biffel, "Performance Evaluation of Workflow Scheduling Strategies Considering Transportation Times and Conveyor Failures," *International Conference on Industrial Engineering and Engineering Management (IEEM)*, 2008, pp. 389-394.
- [8] M. Merdan, T. Moser, D. Wahyudin, and S. Biffel, "Simulation of Workflow Scheduling Strategies Using the MAST Test Management System," *10th International Conference on Control, Automation, Robotics and Vision (ICARCV 2008)*, 2008, pp. 1172-1177.
- [9] C.D. Nguyen, A. Perini, and P. Tonella, "Ontology-based test generation for multiagent systems," *7th international joint conference on Autonomous agents and multiagent systems*, International Foundation for Autonomous Agents and Multiagent Systems, 2008, pp. 1315-1320.
- [10] A. Press, "Software disasters are often people problems," <http://www.msnbc.msn.com/id/6174622>, 2004.
- [11] A. Spillner, and T. Linz, *Basiswissen Softwaretest*, dpunkt-Verl., 2004.
- [12] E.H. Van Leeuwen, and D. Norrie, "Holons and holarchies," *Manufacturing Engineer*, vol. 76, no. 2, 1997, pp. 86-88.
- [13] P. Vrba, "MAST: manufacturing agent simulation tool," *IEEE Conference on Emerging Technologies and Factory Automation (ETFA '03)* 2003, pp. 282-287.

# PLeTs—Test Automation using Software Product Lines and Model Based Testing

Elder de M. Rodrigues  
Faculty of Informatics -  
PUCRS  
Porto Alegre, Brazil  
elder.macedo@pucrs.br

Leonardo D. Viccari  
Faculty of Informatics -  
PUCRS  
Porto Alegre, Brazil  
leonardo.davi@pucrs.br

Avelino F. Zorzo  
Faculty of Informatics -  
PUCRS  
Porto Alegre, Brazil  
avelino.zorzo@pucrs.br

Itana M. Gimenes  
Faculty of Informatics - UEM  
Maringa, Brazil  
itana@din.uem.br

## ABSTRACT

This paper presents the advantages in combining the Model Based Testing technique with the Software Product Line approach, in order to improve testing efficiency during the software development process. This composition is allowed by a tool able to generate software products to test a specific application, based on its model. Each application contains a set of aspects that may be tested. The tool architecture is based on plug-ins that provide flexibility and allow the expansion of test aspects. To show the results of our approach, we describe two software products developed using our tool: Performance Testing and Security Testing based on UML models.

## Keywords

Model Based Testing, Software Product Line, Plug-ins, Reuse, Performance Testing, Security Testing

## 1. INTRODUCTION

The increase of the use of computer systems on many services also increases the need for more dependable software, therefore software with unexpected behaviors or incorrect operations is not tolerable. Usually software testing is used to improve software dependability by means of removing existing software faults. Furthermore, software testing allows evaluating whether a computer system meets its requirements. Typically, each software requirement is associated, during the development process, to one or more test cases. Each test case, on the other hand, is a condition that guides a tester to determine if the system under test meets or not a particular requirement. Due to the fact that a test analyst is the responsible for the definition and generation tests cases, information about the part of the system that will be

tested is usually preserved only within test scripts, or in a test plan. However, if a model of the system is constructed, it contains the expected behavior of the system, therefore it would be possible to extract important information for the testing team from the system Model.

The use of models improves the software development of a system, because it makes it possible to obtain more precise and correct specification. Moreover, the use of models is effective in systems development whose requirements are constantly changing—to generate new tests, the testers simply modify the models and quickly regenerate the set of test cases. The technique that uses the model of the system to define and generate test cases is called Model-Based Testing (MBT) [9]. However, the test cases generation based on models is still an area to be explored.

Some studies [2] [4] show that to obtain acceptable results in industry, a purely academic method to apply MBT would not be fully effective. It is necessary to associate a simple modeling language with the power to describe the expected behavior of a system. Additionally, it is also desirable to be able to generate test cases from the models. For instance, the Unified Modeling Language (UML) is a semi formal specification that allows to produce a test plan for the early stages of the software development process.

MBT can be applied in various test domains, such as in functional testing, security testing, performance testing, and so on. However, usually it is necessary to develop different testing tools for each domain. Despite the development of various tools, and high similarity among them, is very likely that the developed artifacts (models, parsers, scripts) for one testing domain are reused in another testing domain. In an ideal situation, the artifacts produced during the process of developing a MBT tool should be reused when a new tool is developed, thus reducing the development time and effort.

Software Product Line (SPL) is a technique that allows, through the reuse of software components, to create a portfolio of similar software systems, thus reducing time to market, cost, productivity, and improving quality [3]. In this context, this paper presents an approach to automate parts of the generation and execution of test cases. Our approach follows a workflow to generate test cases based on the model of a system. This workflow can be modified to include different artifacts for any desired test domain. Furthermore, we

propose a tool that arranges the test workflow in accordance to an SPL, focusing on the reuse of components.

This paper is organized as follows. Section 2 discusses some concepts (Model Based Testing technique and Software Product Line approach) and tools. In Section 3, we present the architecture of PLeTs-Testing Product Line Tool that extracts relevant information from models and generates test data to specific domains. Each domain is described by a series of plug-ins that are incorporated to the Product Line. Finally, in Section 4, we present two example products already developed using our approach and tool. These example products provide information to evaluate performance and security tests on web based applications.

## 2. RELATED WORK

### 2.1 Model Based Testing

Modeling is an economic way of capturing knowledge about a system and to allow the reuse of this knowledge as a system grows. The use of a model to describe the behavior of a system is an advantage and a proven technique that the test and development teams have in their favor.

Usually, information about the performed tests and results are preserved in few artifacts, such as test plans, test scripts or models. However, when using a model to define the system structure and expected behavior, we have a rich set of information about the system, that may increase the test quality. In addition, we have many techniques that aim to extract relevant information from models, allowing to focus the test process in parts that may compromise the system correctness, instead of focusing on the test case generation and execution.

The application of MBT has increased the software quality as a whole. The use of models describes how a system must be developed, and the use of this information incremented by desired characteristics makes the software testing more efficient, exploring system parts that may not guarantee some aspects [9].

In [6] are presented the basic tasks in MBT: knowledge about the system being tested, choice of model, construction of model, generation of tests, execution of tests, collection of test results and use of test results.

Several studies such as [2] seek to take advantage of this technique through the development of MBT.

### 2.2 Software Product Line

A SPL is a set of software systems that share a common and managed set of features, meeting the specific needs of a mission or market in particular and are developed from a common set of basic features [3].

According to [16], SPL is rapidly emerging as an important and viable paradigm of software development, allowing scope for improvement in the level of order of magnitude in time to market, cost, productivity, quality and other areas.

SPL allows for rapid entry of a product on the market as well as making it easier for mass customization of products of a company. Companies are finding that the practice of building sets of related systems from common assets can, in fact, produce quantitative improvements in product quality and consumer satisfaction, efficiently meeting the current demand for mass customization.

According to [13], the tangible benefits, i.e., those that can be measured by any metric, are profitability, quality, perfor-

mance, the integration time (in incremental development) and productivity. Furthermore, there are also several intangible benefits (not easily measured), such as lower effort, the acceptance of the methodology and job satisfaction of the development team, and also customer satisfaction with the final product.

### 2.3 Tools

As presented in Section 2.2, a SPL promotes the reuse of artifacts through the process of software development. In order to provide the benefits of a SPL, tools have been developed. Some of them are based on the process of automatic generation of a variation of an existing product and others in the derivation of a product from the composition of artifacts already developed (e.g. models, software components). Some examples of SPL tools are Pure: variants [15], Gears [1] and FMP2RSM. Other examples can be found in [14].

Pure:variants is a commercial tool for derivation of SPL that uses models to describe the domain of the problem, the elements of implementation and the products that compose SPL. It provides an explicit representation of sets of components (parts that are joint models of the products family) and features. Regarding the functionality, Pure:variants provides direct support for three different stages in the software life-cycle: the feature type, which defines the common characteristics and variability of Product Line, the configuration of a SPL, which is defined as the instance of the line, and the construction of a product by the composition of the artifacts.

Gears is a commercial tool that allows the definition of features, variability points and concrete products by selecting variants. The tool provides as infrastructure a development environment and a set of configurations that facilitate the creation of a SPL from the definition of a model focused on the automatic derivation of products. However, it also shares a weakness with Pure: variants, since neither of both provide direct support for stages such as analysis of requirements or architectures, development of assets or implementation of a SPL.

FMP2RSM is an implementation of FMP2RSM templates of models based on features of modeling tools for the IBM Rational Software Modeler [11] and IBM Rational Software Architect [10]. The FMP2RSM integrates the features modeling plug-in (FMP), which is originally a plug-in for the Eclipse IDE and aims to edit and configure feature models with Rational Software Modeler (RSM), enabling the modeling of Products Line in UML and automatic derivation of products. The FMP has a purely academic history, which changed with the FMP2RSM, since its use makes sense only when integrated with a commercial tool, although the tool is available for free.

### 2.4 Discussion

The tools discussed in [14] cover only the stages of defining features and variations, as well as the composition of products through the selection of features. However, there is almost no explicit support for testing, implementation and maintenance of SPLs.

Moreover, the tools are commercial or depend on other commercial tools. In addition, the tools presented are not designed to software testing, and there is no development of any Testing Product Line.

In Section 4 an architecture for a novel SPL for testing

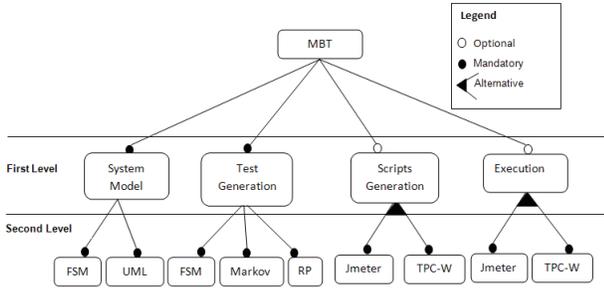


Figure 1: Model-based - Feature Model

tools, focusing mainly on the generation and implementation of MBT, but also applicable to many other areas.

### 3. PLETS ARCHITECTURE

In order to provide a tool that automatize the generation, execution and results collection of MBT, this paper presents the architecture and implementation of PLeTs. The tool is able to manage the MBT process, based on the concepts of SPL. Its goal is not only the reuse of artifacts to make it easier and faster to develop a new tool of the family, but also to improve the creation, runing and collection of test results. It was developed with the intent to be used by software engineers, developers and test engineers, assisting the process of defining and executing test cases and test scripts.

We have developed a Feature Model that represents the definition and visualization of the scope of the tool (in Figure 1). The first level contains the basic features that an MBT Product Line should have; e.g. a Formal Model. It is important to mention that this Feature Model can be extended with new features is added (in the first or second level), for the evolution of the SPL.

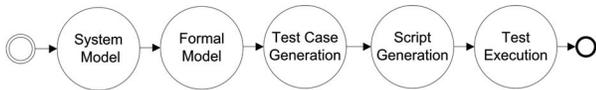


Figure 2: MBT - Basic Workflow

Our second step was to chart each basic feature (basic level) presented in the Feature Model into a step in a workflow of PLeTs. This is important because we may convert any test activity into a workflow model (see Figure2). Considering that we can model a sequence of model-based testing generation and performing activities as a workflow, we can use this workflow as the input of the tool. The features presented in the second level (variability points) are mapped to the software components during the development of the tool.

The PLeTs architecture is divided into two main parts, the core and the components, as shown in Figure 3. The core is responsible for configuring and organizing the workflow (basic features) and providing interfaces to connect components (specialize features). The components, in this case, are plugins that implement a functionality in any step of the workflow. The plugins are used/developed to extend de basic SPL functionality. Each component (Figure 3) that

is responsible for describing a step in our basic workflow (Figure 2) has a same interface to the core of our tool. For instance one can choose Markov Chains, Finite State Machine (FSM) or Petri Nets to describe the Formal Model.

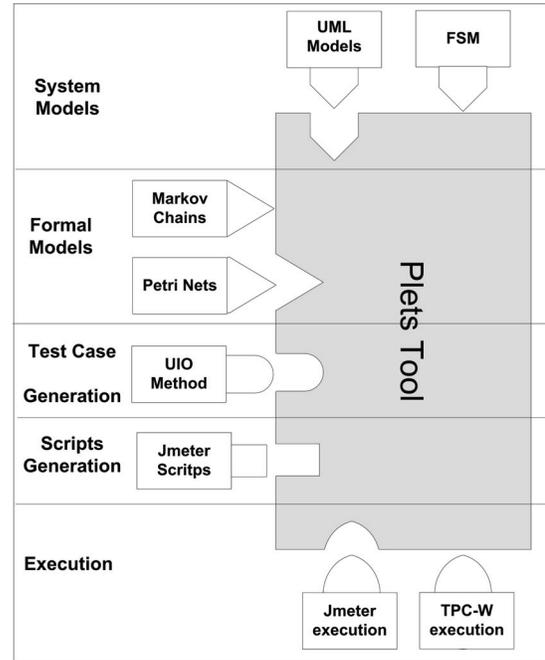


Figure 3: PLeTs Architecture

The combination of the plugins, one for each basic MBT step, will build each product in the MBT Product Line. However, when the test analyst decide to implement a new test tool, it is necessary to define the dependency among the plugins. This is important because different software testing techniques may need different system models, formal models, scripts generations and execution steps. For example, if we want to add a new product to test parallel systems, we can reuse the system model, the formal model (if it is available in the MBT Product Line), but we may need to implement one plugin to generate de test scripts for a specific tool to test the parallel system.

In the development process, some functionalities will be defined, and every functionality theoretically corresponds to an activity in the workflow, since it has also a certain amount of functionalities and a return type. This leads to another feature of the tool: it allows the user to associate activities to plugin methods, through a drag-and-drop user interface, as long as their interfaces (i.e. parameters and return types) are compatible. Here is the point where reuse starts to be used as a feature of our tool. Suppose that we built a plugin to handle model-based performance testing. This plugin will have methods for parsing the UML model and for generating test cases, for example. If we need to include another plug-in in the tool, in order to support model-based security testing, we do not need to implement the parsing methods again, if they are semantically the same as the ones implemented in the previous plug-in. This means that a set of workflow/plugin associations can include one workflow and methods from many plug-ins. Since the user can add as many plug-ins as he wants, extensibility is also trivial to

understand.

Originally, the tool was thought to handle only linear workflows, but later on, we realized that this approach was insufficient. So, a new mechanism to handle workflows as graphs was added to the tool. That means that workflows can contain branches and joins, which are represented in the workflow XML file as follows: if an activity B depends on the activity A, B will have an attribute “dependsOn” indicating that it depends on the activity A. If more than one activity depends on A, we have a branch after the execution of A. If activity C depends on both A and B activities, we have a join before activity C. This new feature will be important when the tool needs to expand and to generate complex test cases, as the test of concurrent systems.

In order to demonstrate the expansion of the tool to support other test aspects, the next section present how it is possible to connect two existing works, taking advantage of the tool capability in handling workflows, as well as the support to model-based testing plug-ins. The two test aspects already supported by the tool are Performance Testing and Security Testing based on UML models.

## 4. CASE STUDIES

In this section we present the development of two Model-Based Testing tools based on the architecture of the Testing Product Line presented in Section 3.

### 4.1 Performance Testing in Web Applications Based on UML Models

As a case study, we implemented a tool to generate test cases based on UML models of the application under test (SUT - System Under Test).

To implement the tool (testing product), we use UML diagrams from the application to extract information about the behavior of the application under testing. In this case, the needed information will be extracted from Use Case Diagrams and Activity Diagram. However, some information necessary to automate the generation of performance test cases in web applications might not be found in the UML Diagrams and, therefore must be inserted. This information, relevant for understanding the behavior of the system, is inserted through stereotypes from UML 2.0 SPT (Schedulability, Performance and Time) Profile.

In the Use Case Diagrams, two information must be inserted through the use of UML stereotypes. The first, called the “PApopulation”, is the number of users that will access the system during the test, and the second, called “PAprob” is the probability of execution of each Use Case.

The inclusion of such information is necessary in order to inform how many users must be simulated in each test, and what is the probability of a given path be followed by each Use Case. Using the tag “PAprob” is especially important when the application under test is an e-commerce application, because some links are more accessible than others, requiring the simulation of this behavior during the application test performance. Stereotypes must be placed in each activity of the Activity Diagram, aiming to simulate the time it takes for each user to perform the activity (think time). In this case a tag is inserted with a random value, corresponding to the think time for each activity.

After the modeling it is necessary to extract relevant information from the models for the generation of test cases. To extract this information and standardize the format for

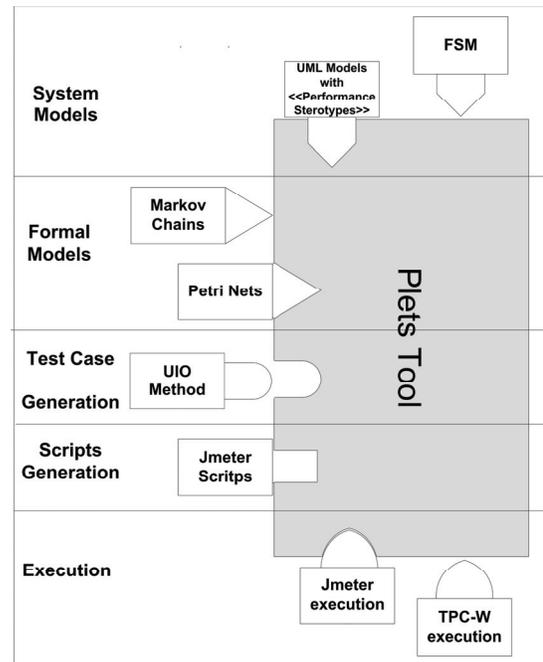


Figure 4: PLeTs Tool - Performance

entering data into the tool, we develop a XML parser (plugin that we add to the MBT Product Line), which extracts and standardizes the format of entry to the tool for generation of test cases/scripts. Using this new format, the data is inserted into the tool which generates the possible pathways and interactions of the user within the system, thus generating the test cases. To generate the tests cases it is necessary to develop a new plugin that supports performance testing generation. Motivated by this need, we developed a plugin that implements Generalized Stochastic Petri Nets - GSPN. The next step is the develop a new plugin (XML parser) to convert the tests cases into test scripts to be run by the performance testing tool in web applications, such as Jmeter [8] and TPC-W [7]. Figure 4 depicts the new testing tool, i.e. our core with the chosen plugins.

### 4.2 Security Test Based on UML Models

A tool for generating test cases for security [12], aims to enable the automated generation of test cases for security from UML models.

The work offers a set of security stereotypes that can be inserted into UML models of the application, denoting items that may become vulnerable if not properly developed. From these stereotypes it is also possible to generate test cases for security, indicating the steps to be performed by the tester to verify if the vulnerability previously reported in the model affects the software or not. The use of these stereotypes has two objectives:

- Guide the developers during the software design, highlighting parts of the model that may become vulnerable if not properly implemented;
- Provide necessary information for the automated generation of test cases.

To analyze the model and extract relevant information,

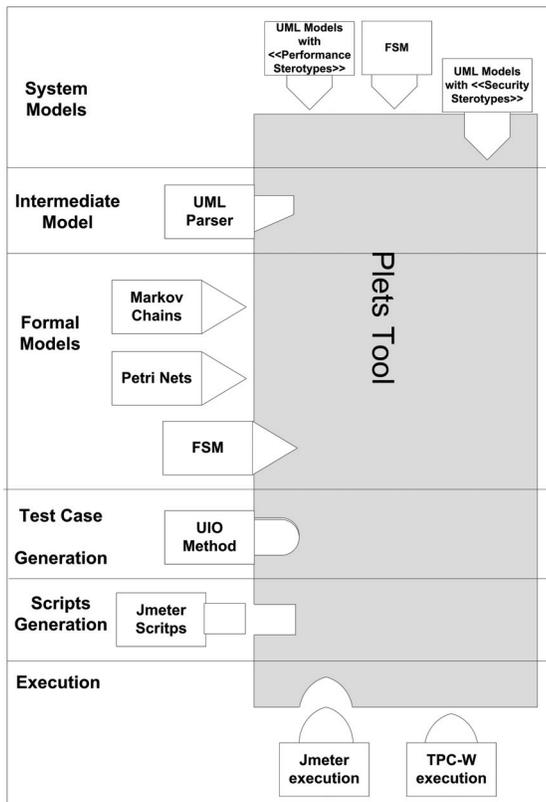


Figure 5: PLeTs Tool - Security

the algorithm Unique Input/Output (UIO) [5] was implemented, which utilizes a Finite State Machine with inputs and outputs to establish the sequence for implementing the software and generating the test cases.

Figure 5 shows the general architecture of this product, describing the steps followed from the stage of modeling of stereotypes (System Model) until the generation of test cases.

After the insertion of stereotypes in the UML model of the proposed application, a corresponding XMI file is generated, describing the structure of the system, activities, use cases and stereotypes. The XMI file is analyzed and converted (reusing the XML parser developed for the performance product) into a Finite State Machine (FSM) with inputs and outputs. The UIO method is then applied, whose outputs are sequences of steps that, after being handled, are the equivalent of test cases in natural language.

As mentioned, the definition of stereotypes was performed with the aid of some taxonomies of security (especially the proposal by [18]) and the information made available by some repositories of security, such as the database provided by the OWASP project [17]. From these, 6 stereotypes were proposed:

- <<BufferOverflow>>: indicates the maximum number of characters that a field may receive;
- <<Flooding>>: indicates the maximum number of simultaneous connections supported by the application;
- <<Encrypt>>: indicates the name of the field to be

encrypted;

- <<ByPassing>>: indicates roles that can access certain links in the system;
- <<Expiration>>: indicates the maximum time to expiry of the user's session if it does not interact with the system;
- <<SqlInjection>>: indicates the fields that may be susceptible to SQL Injection attacks.

Finally, we emphasize that the process of specification of the stereotypes defined in the work associated the flexibility provided by UML to the knowledge provided by the repositories of security. Thus, to expand the proposed set and add stereotypes able to represent other vulnerabilities, it is only necessary to analyze the security database, extract the information needed to test and propose the new case.

### 4.3 Discussion

The proposed architecture for the tool is generic enough to be used in various applications. The focus of this work is the generation and implementation of Model Based Testing using the concept of Software Product Line. The suggested plug-ins represents excellent use cases, since they share many of the same concepts.

The possibility of association among the features of one or more plug-ins is a desirable item from the standpoint of re-use and composition, and was reached by the use of the tool. Due to the simplicity of developing a new plug-in, without touching the source code of the core of the tool, the extensibility is guaranteed.

As the work described in Sections 4.1 and 4.2 are basically a sequence of well defined steps, a workflow for each work can be defined, and therefore the tool is capable of supporting both types of tests. Figures 4 and 6 illustrate two possible workflows that represent the activities of both works.

The actual implementation of plug-ins can occur in several ways:

- One plug-in supporting each workflow with each plug-in that is implemented having at least one method of implementing each one of the activities of the workflow.
- A single general plug-in, able to manage both workflows. This can be considered as a plug-in implementation of Model Based Testing, able to handle many types of model based testing.
- Various plug-in, each one specialized in a common task of many workflows. For example, a plug-in specialized in parsing of UML diagrams, another plug-in specializing in the generation of tests, etc.
- A combination of the previously mentioned items. As the tool supports the association, in a dynamic form of plug-ins to workflows, it is possible to combine more than one of the implementations described above in an execution.

One of the reasons for the statements above is the fact that both products have similar workflows, despite possible internal differences. Despite the fact that only two products were added to MBT line, we can already reuse components. For example, in the examples the same system model and the same UML parser.

## 5. CONCLUSION

Based on the fact that the software testing is a crucial part of quality assurance in software, it is safe to affirm that any methodology or software development process should have a strong concern about the planning, implementation and collection of metrics for testing. The automation of the stages of the cycle of testing has many benefits, and optimizes the time and effort of the test team.

Techniques such as MBT exploit the capture of existing knowledge in formal models of software, and is valid for re-use this knowledge in other areas of the software development process. Taking advantage of the fact that different types of tests have several steps in common in their life-cycle, this study also addresses the concept of Software Product Lines as an interesting concept to be applied to the area of Software Testing.

Thus, the work has presented the architecture of an extensible tool, based on plug-ins, with the goal of automating the testing stages of the software cycle, making use of techniques of composition and re-use of software components. The tool has proved to be useful not only for the testing area but also to any set of activities ordered in a workflow.

The future steps of this work include improvements and extensions in the architecture with a view to adding security, ease and intuition of use and, especially, more products for the Software Product Line. To achieve this goal, we work together with some Software Houses, to apply this tool on their software testing process. At this time, we are encouraging these companies to develop their own plug-ins and/or modify the plug-ins already developed by others.

Finally, we expect that in a short period of time we can release the tools and plug-ins developed in a repository of software projects. Thus, we seek to reach all the people who work with software testing and not just academic users.

## 6. ACKNOWLEDGMENTS

Study developed by the Research Group of the PDTI 001/2010, financed by Dell Computers of Brazil Ltd. with resources of Law 8.248/91. We thank INCT-SeC/CNPq and CAPES for the financial support. Prof. Avelino Zorzo is also funded by CNPq.

## 7. REFERENCES

- [1] BigLever Software Inc. "Gears". Available from <http://www.biglever.com/solution/product.html>. 14/06/2009.
- [2] Bertolino, A.; Marchetti, E.; Muccini, H. Introducing a Reasonably Complete and Coherent Approach for Model-based Testing. *Electronic Notes in Theoretical Computer Science*. Proceedings of the International Workshop on Test and Analysis of Component Based Systems, 2004.
- [3] Clements, P.; Northrop, L. and Northrop, L. M. *Software Product Lines: practices and patterns*. Addison-Wesley Longman Publishing Co., Inc., Boston, USA, 2001.
- [4] Craggs, I.; Sardis, M.; Heuillard T. Model Based Testing – Agedis case studies: Model-Based Testing in industry. In *European Conference on Model-Driven Software Engineering*, 2003.
- [5] Delamaro, M.; Maldonado, J.; Jino, M. *Introduction to Software Testing*. Elsevier, São Paulo, Brazil, 2007.
- [6] El-Far, I. K.; Whittaker, J. A. *Model-Based Software Testing*. In *Encyclopedia on Software Engineering*. Wiley, 2001.
- [7] García, Daniel F.; García, J. Tpc-w e-commerce benchmark evaluation. *IEEE Computer*, pages 42–48, 2003.
- [8] Halili, E. *Apache JMeter*. Packt Publishing, 2008.
- [9] Hartman, A. Model based test generation tools. Available from <http://www.agedis.de/documents/ModelBasedTestGenerationTools-cs.pdf>. 02/12/2008.
- [10] IBM. Rational software architect. Available from [http://www-01.ibm.com/software/awdtools/architect/swar\\_chitect](http://www-01.ibm.com/software/awdtools/architect/swar_chitect). 23/06/2009.
- [11] IBM. Rational software modeler. Available from [http://www-01.ibm.com/software/awdtools/modeler/swm\\_odeler/index.html](http://www-01.ibm.com/software/awdtools/modeler/swm_odeler/index.html). 23/06/2009.
- [12] Karine P. Peralta; Alex M. Orozco; Avelino F. Zorzo; Flávio M. Oliveira. Specifying Security Aspects in UML Models. International Conference on Model Driven Engineering Languages and Systems, 2008.
- [13] Matinlassi, M. Comparison of software product line architecture design methods: COPA, FAST, FORM, Kobra and QADA. *Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on*, pages 127–136, 2004.
- [14] Pohl, C. et al. Survey of existing implementation techniques with respect to their support for product lines. Available from <http://www.evolvica.org/wpcontent/uploads/2009/03/ample2007-31.pdf>. 02/04/2009.
- [15] pure-systems GmbH. Pure::variants. Available from [http://www.puresystems.com/Variante\\_Management\\_49.0.html](http://www.puresystems.com/Variante_Management_49.0.html). 20/06/2009.
- [16] Software Engineering Institute (SEI). Software Product Lines (SPL). Available from <http://www.sei.cmu.edu/productlines/>. 01/07/2009.
- [17] The Open Web Application Security Project. The ten most critical web application security vulnerabilities. Available from <http://www.owasp.org>. 21/01/2009.
- [18] Weber, S; Karger, Paul A; Paradkar, A. A software flaw taxonomy: aiming tools at security. *SIGSOFT Softw. Eng. Notes*, pages 1–7, 2005.

# Experience with Maintenance of a Functional GUI Test Suite using IBM Rational Functional Tester

Yuri Shewchuk  
inUnison Technology  
Calgary, Alberta, Canada  
yeshewch@ucalgary.ca

Vahid Garousi  
Software Quality Engineering Research Group  
Dept. of Electrical and Computer Engineering  
University of Calgary, Calgary, Alberta, Canada  
vgarousi@ucalgary.ca

**Abstract**— Most of the modern software systems are being maintained and evolved through numerous versions. Thus, effective co-maintenance and co-evolution of their test suites along with their source code becomes an important and challenging issue. In this context, issues such as the types and extent of required maintenance activities on test suites, change in size/complexity, fault and cost effectiveness of multi-version functional test suites are among the most important issues. To study, analyze and get insights into co-maintenance and co-evolution of functional test suites with software systems w.r.t. the above issues, we have performed a case study on a functional GUI test suite of a popular open source project (jEdit). We chose as our test tool the IBM Rational Functional Tester, one of the leading commercial functional testing tools. The case study reveals interesting practical/empirical insights into the subject, e.g., developing a functional test suite in an earlier version and maintaining it might be a cost effective approach.

**Keywords**- *Functional Testing, Test Maintenance and Evolution, Tool Evaluation.*

## I. INTRODUCTION

Functional testing is based on the view that the program is a function that maps the input domain to its output range. Unlike white-box testing that designs the test cases based on the structure of the code, the test cases in functional testing are designed based on the functionalities specified in the software requirements.

GUI testing is a type of functional testing that uses the graphical user interface of the system Under Test (SUT) to ensure it meets its specifications.

Since most of the modern software systems are being maintained and evolved through numerous versions, the co-maintenance and co-evolution of their functional test suites in parallel is an important while a challenging issue. In this context, issues such as changes in size/complexity, fault detection, and cost effectiveness of multi-version functional test suites are among the important challenges for both the academic and practitioner software testing communities. For example, a test manager might ask: is it more cost effective in the long run to develop a multi-version test suite in earlier versions and to maintain it rather than developing the test suite later in future versions?

To study, analyze and get insights into the nature of co-maintenance and co-evolution of functional test suites, and to shed some light on the above challenging issues, we have performed a case study on a functional GUI test suite

we have developed for a popular open source project (jEdit [1]).

Introduced to the open source community in 1999, jEdit [1], written in Java, is a popular open-source text editor for programmers. It is a popular and very active project, i.e., its SourceForge mailing list has had about 214 entries every month (about 7 daily) since 1999. According to [2], as of September 2008, Introduced had 157 developers and was ranked the 6th all-time most active project in SourceForge. To conduct regression functional GUI testing, we first developed a functional test suite for version 3.2.2 of jEdit and then conducted regression using that test suite on versions 4.0 and 4.1 subsequently.

For the testing tool, we selected the IBM Rational Functional Tester (IRFT) version 7.1, one of the leading commercial functional testing tools in the industry. A recent report by Forrester Inc. [3], a market research firm, ranks IRFT as the second leading functional testing tool (HP QuickTest tool being in the first rank).

Our case study intends to mimic real-life industry-scale projects as we are using a popular industry-strength testing tool, i.e., IRFT, and also a widely-used system under test (i.e., jEdit).

Formulated using the Goal-Question-Metric (GQM) approach [4], the goal of this study is: to gain insights into co-maintenance and co-evolution of functional test suites with a multi-version System Under Test (SUT) from the point of view of software testers in the context of using the IRFT tool for generating and keeping test suites up to date. With the above goal in mind, we pose the followings research questions (RQs):

- RQ 1: What is the effort level needed to create a functional test suite using IRFT and to maintain it for the next versions of the SUT?
- RQ 2: What are the types and scales of test maintenance activities when the SUT evolves across multiple versions? For example, for how many test cases, a “re-recording” of the GUI test case is needed due to changes in the SUT?
- RQ 3: What is the fault detection ability of a test suite, developed for a first version, on the next versions?
- RQ 4: How does the size of the test suites grow with growth in size (i.e., LOC) of the SUT across different versions?

The above RQs intend to provide empirical, practical, and also tool evaluation outcomes in the context of regression functional testing. For example, since IRFT identifies itself a “regression testing tool” (according to its

website [5]), RQs 1, 2 and 3 intend to assess the effectiveness of regression testing using the IRFT with respect of the effort needed to develop and maintain functional test suites and also the fault detection effectiveness of those test suites.

In the view of growing sizes (e.g., LOC) of software systems and their automated test suites, RQ 4 intends to provide insights on the scale of increase in the GUI test suite code size versus the size of its SUT code base.

The rest of this article is structured as follows. The related work is discussed in Section II. The design of the case study is discussed in Section III. Case study results are presented and analyzed in Section IV. Threats to the validity of our study are discussed in Section V. Finally, future works are discussed in Section VI.

## II. RELATED WORK

A few existing works have previously studied the maintenance and evolution of test suites for multi-version software [6-11].

The work in [6] by Kaiser et al. is one of the early studies on the subject which presents a novel approach to integration test management for large-scale multi-version projects. A new change management framework, called *Infuse*, is introduced which checks out new version of the modules from a version control system, extracts their dependency information and then builds automated integration tests. The approach is experimented on a small-scale SUT. But the issues of tool evaluation and fault effectiveness of integration test suites across versions is not addressed.

The work in [7] is a case study on regression test suite maintenance for an evolving system with three updated versions, created using three different change strategies: change propagation, refactoring and role splitting. Test suites for automated unit and functional tests were used for regression testing the extended applications. The study found that, with one change strategy, more changes were made in the tests code, and with another strategy no changes were needed for the unit tests to work. The SUT used was a graphical tool called Drawlet (~17.8 KLOC).

Reiss presents in [8] a prototype tool for a mechanism to ensure that the different software dimensions (e.g., specifications, design, and test cases) evolve concurrently. The author argues that as the source code of a system evolves, developers gradually forget to update the specifications, the design, and the test suites. Interesting comments about test artifacts in system evolution are made by the author, e.g.: “*Test cases lose relevance when developers fail to add new test cases as the software evolves.*”, and “*Testing may be the single dimension that receives proper attention, but the test suite usually bears little relationship to other dimensions.*” The technique was applied to two SUTs: (1) a 2,500 LOC game-playing Java program, and (2) the prototype tool itself (~17 KLOC).

Elbaum et al. studied in [9] the impact of software evolution on code coverage. The results of their studies suggest that even relatively small modifications can greatly affect code coverage information, and that the

degree of change impact on coverage may be difficult to predict. As SUTs, this study used two systems: (1) the GNU Bash shell program (~90 KLOC in its last 2.04 version), and (2) an anonymous C program developed for the European Space Agency (6.2 KLOC). The study did not consider functional test suites.

Memon et al. [10] studied the fault-detection effectiveness of GUI test cases for rapidly evolving software. The study focused on one of the rapid-feedback-based testing techniques, i.e., *smoke* testing. A smoke test generally consists of a collection of preliminary test cases that are applied to a newly-created or repaired software. Smoke testing is done by developers before the build is released or by testers before accepting a build for further testing [12]. The results of the empirical study showed that: (1) the entire smoke testing process is feasible in terms of execution time, storage space, and manual effort, (2) smoke tests cannot cover certain parts of the application code, and (3) having comprehensive test oracles may make up for not having long smoke test cases. The SUT of choice was an open source office suite called *TerpOffice* (~90 KLOC). Three main questions raised and empirically answered by [10] were:

- What is the fault detection ability of the GUI tests?
- What is the impact of using different test oracles?
- What is the impact of test suite size on test results?

The study did not discuss in detail the test suite’s maintenance and evolution across different versions of the SUT.

The work by Harrold et al. [11] addresses the problem of regression test selection and minimization for Java software: selecting only a portion of the test suite developed for an earlier version of a SUT for use in the testing of a later version while guarantying that the faults revealed by this subset will be the same as those revealed by running the entire test suite. Four SUTs were used: SIENA (Scalable Internet Event Notification Architecture), jEdit, JMeter and RegExp. Regression test selection was performed in code level and thus generates (selects) white-box unit tests. The study on jEdit was conducted on versions 3.0-pre4 and 3.0-pre5. Based on the changes between the above two releases, the authors created 11 sub-versions of the software (we are not sure why this was done). A structural (unit) test suite (with 189 test cases) was developed to exercise various features of jEdit. The proposed regression test selection technique was used to select, from the test suite of a previous sub-version, varying percentages (between 3% and 100%) of test cases to test the next sub-version of jEdit. Although interesting results are reported in this work, its focus was on unit test suites, while the focus of our work is on functional test suites. Furthermore, we report in this article a few interesting insights based on the correlation of test coverage and code size across different versions which no other work has studied.

From the above discussion of the related work, we can find out that: (1) very few existing works on the subject have used at the same time a popular industry-strength functional testing tool and a widely-used SUT such as the

one we use in this work, and (2) issues such as effort and types and scale of required test maintenance activities in co-maintenance of functional test suites have not been the focus in the previous works and not many insights into these important topics have been reported. On the same topic, in a keynote speech [13] on the empirical research in software testing at the International Symposium on Empirical Software Engineering and Measurement (ESEM) 2007, Briand mentioned that: "... , in most cases [studies], the releases [systems under test] are not real software releases with real changes, and therefore, besides a handful of studies, it is unclear what kind of external validity can be expected from these results." Our case study aims to be a preliminary experience report in providing insights towards the above issues.

In term of the related work on the use of IBM Rational Functional Tester (IRFT), although only a handful number of works (e.g., [3, 14, 15]) evaluate IRFT or use it for testing experiments, in our search of the literature we found no study specifically examining or evaluating IRFT for multi-version SUTs. One of the important features which good functional testing tools (such as IRFT) should have is that porting a test suite from an earlier version of a SUT to a newer version should be relatively simple and not very costly. In order to evaluate this statement, our case study intends to evaluate the amount of effort it takes to port an IRFT test suite from an older version of the SUT to a newer version.

### III. DESIGN OF THE STUDY

#### A. System Under Test

jEdit [1] is a text editor specifically intended for developers. The features of this tool includes like syntax highlighting, code completion, code block contraction and expansion, which can improve a developer's productivity. A snapshot of jEdit version 4.3 running on Linux is shown in Figure 1. As an indicator of jEdit's popularity, according to [16], jEdit was downloaded on average 1,719 times a day in the period of August 2007-August 2008. jEdit has evolved since April 2000 and it is currently (as of February 2010) in version 4.3.1. To keep our project resources manageable, we conducted our study on versions 3.22, 4.0 and 4.1 only (as representative versions).

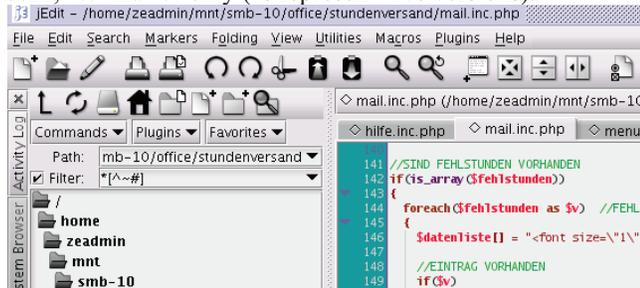


Figure 1 jEdit version 4.3 running on Linux.

Since the context of our study is functional testing, we needed to have a reasonably precise requirements document to derive the functional test suites from. Since jEdit does not seem to have a (publicly-available) formal

requirements document, we developed for this system a reverse-engineered requirements document (available online [17]) using the notion of use-cases based on its feature list, users' guide and also the business logic of its domain (a text/code editor for software developers). We developed 57 use cases in total for version 3.2.2 which included: 25 text editing, 10 file navigation, 11 syntax highlighting, 8 programming assistance, and 3 multiple file functionality use cases.

Furthermore, unlike some other open source tools (e.g., JFreeChart and FitNesse), there were no functional test suites bundled with jEdit. So we developed one based on our inferred requirements document for version 3.2.2. Since some of the use cases had alternative paths, a functional test suite of 89 test cases would be required to fully cover the 57 use cases. To mimic time constraints in real projects and also in the outset of our project timeline, we devised 71 functional test cases, thus achieving 79.8% requirements branch coverage for version 3.2.2. The 71 test cases were selected out of 89 possibilities by conducting a simple risk analysis and test prioritization based on system's use cases. According to discussions with our partners in the software testing industry, we are well aware that there are inadequate automated functional test suites in most real-world projects. Thus, our scenario seems to be realistic with respect to the level of test automation.

Having developed the functional test suite for version 3.2.2, regression functional testing was performed on versions 4.0 and 4.1 to see if the defect fixes, code/GUI changes and the new added features (functionality) broke some of the previous test cases. In addition, new functional test cases were also added to cover the new GUI features in two later versions. This resulted in adding 3 and 5 new test cases to the set of 71 initial test suite, respectively, in each later version (4.0 and 4.1). Note that we conducted "retest-all" regression testing approach since the entire test suite was automated and there was no human effort or cost involved in re-running all the test cases on next versions.

The following is an example functional test case code we developed in IRFT to test one of the use cases ("Copy text"). All the other test suite code for the three versions of jEdit are available online [17] and can be used to replicate our study or to perform other similar studies.

```

1. public class TC_TE_10 {
2.     public void testMain(Object[] args){
3.         startApp("jedit-322");
4.         view().inputKeys("aaaaa{ENTER}");
5.         view().inputKeys("aaaaa{ENTER}");
6.         view().inputKeys("aaa{ENTER}");
7.         view().inputKeys("aa{ENTER}");
8.         view().inputKeys("a{ExtUp}{ExtUp}{ExtUp}
           {ExtRight}+{ExtUp}");
9.         view().inputKeys("^c");
10.        String[] arg = new String[1];
11.        arg[0] = "aaa\rnaa";
12.        callScript("TestClipboardWithWordpad", arg);
13.        jMenuBar().drag(atPath("File->Close All"));
14.        discardSelected().click();
15.        view(ANY,MAY_EXIT).close();
16.    }
17. }

```

Figure 2 An example functional test case code in IRFT.

## B. Testing Tool

There are various functional testing tools in the market which are either commercial [3] (e.g., HP QuickTest Professional, and Compuware TestPartner) or open-source [18] (e.g., JFunc: JUnit Functional Testing Extension, and Jameleon).

In order to evaluate and *test* those tools in practice, empirical studies should be conducted in order to try and assess different features of each tool. Thanks to the IBM Academic Initiative program, we have been provided with a license of the IRFT. We used this tool in the generation of automated test cases (scripts).

To gather code metrics about the jEdit code base (e.g., LOC), the *Metrics* Eclipse plug-in [19] was used.

## IV. RESULTS

### A. RQ 1. Effort to Create/Maintain the Test Suite

Figure 3 shows the change in the SUT code size (LOC), effective LOC under test, and the amount of time required to initially create or to maintain the developed test suite. Similar to the effective LOC metric used in the literature, “effective LOC under test” is a measure reported by test tools and denotes the effective number of code lines which are really tested (excluding function declarations, blank lines, comments, etc.).

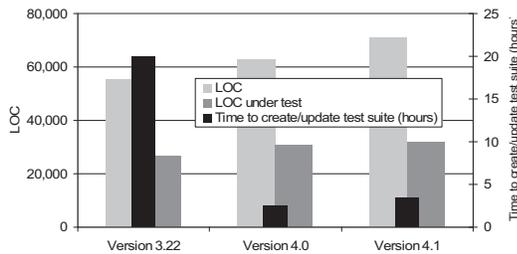


Figure 3 Test suite effort vs. SUT size

From Figure 3, we can see that the total size (LOC) of the system is increasing at a relatively constant pace between releases. This may indicate that there was a steady amount of work being performed by the jEdit development team to add new code to improve or implement new functionality.

The tester subject (i.e., the first author) was a 4<sup>th</sup> year Software Engineering student who was taking an advanced course in testing and also had a few months of industrial testing experience before this project.

There seems to be a large difference in the amount of time required to update the test suite for each version. Once the test suite was developed for the earlier version (about 20 hours), it took minor effort (about 2-3 hours) to maintain it for the next two versions.

In addition, new functional test cases were also added to cover the new GUI features in two later versions. This resulted in adding 3 and 5 new test cases to the set of 71 initial test suite, respectively, in each later version. About half of the time in versions 4.0 and 4.1 were spent on

adding the new test cases, and half of the time on the maintaining old test cases.

### B. RQ 2. Types and Scale of Test Maintenance Activities

After analyzing the overall effort in creating and maintaining the test suite, we wanted to study the types and scale of test maintenance activities. For this purpose, we executed the GUI test suite on versions 4.0 and 4.1 and counted the number of test cases that passed, failed, and also the number and types of changes that needed to be made to the failed test cases due to code /GUI change in the SUT. Figure 4 depicts that information.

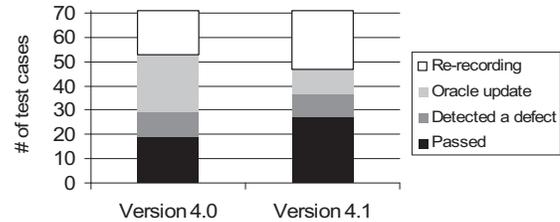


Figure 4 Types of required test maintenance activities.

Out of total 71 test cases in the test suite, 19 and 27 passed in the two later versions, and the remaining 52 and 44 failed, respectively. Out of those failed, 10 test cases in each of the versions 4.0 and 4.1 detected a defect, as per our root-cause analysis. However, the remaining, 42 and 34 failed not because of a defect in the SUT, but were *broken*, i.e., problem with the test cases themselves (as per our root cause analysis), e.g., the test oracle had to be revised due to moving from a version to the next.

For the 42 and 34 test cases that failed due to problems caused by the SUT changes, two types of test maintenance activities were needed to be performed: (1) updating test oracles (called verification points in IRFT), and (2) re-recording the entire or a portion of a test case, for example a previous text edit box was removed in the new version. In versions 4.0 and 4.1, 24 and 10 test cases needed oracle update, respectively, while 18 and 24 test cases needed re-recording. According to our analysis of the jEdit GUI, the number of changes in the GUI was more in version 4.1 compared to version 4.0, and this resulted in more test case re-recording.

The above numbers seem to indicate that although an initial test suite is useful as a test harness for the next versions of a SUT, however a tester should expect to maintain (update) a fair proportion of test cases in the next versions. However, the time spent to maintain the test cases was not an issue in our case study: 2.5 and 3.5 hours in versions 4.0 and 4.1, compared to the initial large time budget of about 20 hours for the creation of the test suite.

### C. RQ 3. Defect Detection in next Versions

Table 1 shows the defects detected by executing the developed test suite (an “x” means that the test suite was able to detect the defect). Coincidentally, exactly 10 failures were detected in each version.

In each of versions 4.0 and 4.1, one failure had apparently been fixed with another failure being introduced, e.g., failure #6 was fixed in version 4.0 and failure #11 was introduced in version 4.0 which did not exist in version 3.22. Since our testing approach was functional, we have not yet aimed at finding the inter-dependency among the code modules to see if the introduced failures in each version were somewhat related to the ones being fixed. But it should be interesting to analyze those cases based on source code changes. A high-level functional comparison of the nature of such failures does not seem to indicate any visible inter-dependency, e.g., failure #11 observed in version 4.0 for the first time does not relate to failure #6 fixed in that version.

The test suite was able to show the presence of 10 different failures in each version of jEdit. This is significant as it gives a test case effectiveness of 14% (10 out of 71) failures per test case. Considering that this test suite is fairly small (71 test cases), this is somewhat a significant value. The detailed failure report (e.g., which test case revealed which failure) is provided online [17].

TABLE I. FUNCTIONAL DEFECTS IN JEDIT DEFECTED BY OUR TEST SUITE.

Defect	v 3.2.2	v 4.0	v 4.1
1. Extra end line character at the end of a file being saved.	x	x	x
2. Multiple selections not implemented.	x	x	x
3. The end of line threshold does not work for inserting text before the end of a line or pasting text.	x	x	x
4. Paragraph count not implemented.	x	x	x
5. Character count is incorrect, it counts end line markers as characters	x	x	x
6. When undoing the first change to a saved file, the file saved status doesn't display the file as once again equivalent to what is in the saved file.	x	Fixed	
7. Line number is off by one when a marker (bookmark) is created.	x	x	Fixed
8. When text is highlighted, navigation between markers (bookmarks) doesn't work.	x	x	x
9. When expanding a hidden code block, any contained code blocks should return to their previous visibility setting.	x	x	x
10. File syntax analysis to determine file type is not implemented.	x	x	x
11. Searching for text multiple times requires selecting find (or replace) from the menu each time. Instead, the dialog should remain visible and the next button clicked.		x	x
12. Moving up after pressing down to go to the end of the next line moves to the wrong position in the previous line.			x

#### D. RQ 4. Size of Test Suite Code vs. SUT Code

Figure 5 shows the overall size and complexity of the SUT code vs. its test suite code.

There are two LOC measures reported for the test suite: (1) Total test LOC which includes the automatically generated test helper classes by IRFT, and (2) Test script LOC (only the actual test code that the tester can see in the IRFT). The figure shows that even though the size of the project was obviously increasing, the test suite size did not change relative to that. This might be explained as the changes/improvements in the SUT's GUI were not in the

same scale as the internal SUT source code. Thus, the functional test suite did not need to grow in the same ratio as the SUT code base did.

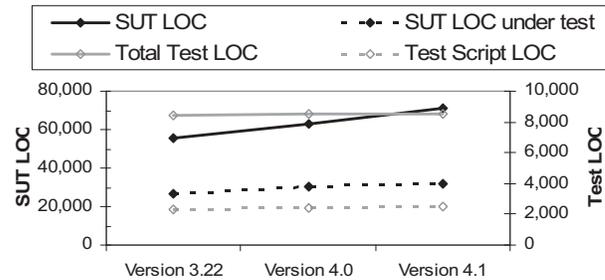


Figure 5 Size comparisons between SUT and test suite.

#### E. Discussions and Lessons Learned

By the results to RQs 1, 2 and 3, we can assess the effectiveness of regression testing using the IRFT with respect of the effort needed to develop and maintain functional test suites and also the fault detection effectiveness of those test suites.

In the view of growing sizes (e.g., LOC) of software systems and their automated test suites, RQ 4 provided insights on the scale of increase in the SUT size versus that of its GUI test suite.

Some of our practical challenges have been learning how to use the testing tool effectively, difficulties with Java version incompatibilities, coming up with a way to infer requirements from an already developed application, and decisions regarding use case granularity and definition for the requirements.

Another more specific challenge which we faced early in the test case development process was determining a process to test the text which was contained within the jEdit window. The jEdit application has a customized text painter (rendering engine), which the IRFT was unable to properly fetch and thus test. This took some time to figure out, but eventually it was determined that Microsoft Wordpad could be used programmatically to test that particular textbox. Once that testing process was developed and made reusable, many following test cases used that testing method without any problem.

One improvement to the test suite which could be made (and should be made if any future versions are going to be tested) is to make the test cases more component-based. This means developing small reusable test components in a more modular and maintainable way. While the development of the original test suite was underway, it was thought that a change to the way that files were opened would not occur. Thus, the process for opening a file was included in almost each individual test case. This could have been pulled out in to a separate process, and that decision to leave it in the individual test cases was regretted later. In jEdit 4.1, a change was made to the file opening business logic and dialog box. This change caused a lot of hassle in modifying a lot of test cases. If the file opening had been moved to a single

component, that component could have easily be changed and made more robust without the need to go over almost all of the test cases to revise the file open procedure.

## V. THREATS TO VALIDITY

In terms of *construct validity*, note that we used only a few metrics (e.g., type and of number of maintenance activities, LOC and defects found) to analyze maintenance and evolution of functional test suites across different versions. For a more comprehensive understanding of the subject, more maintenance and evolution metrics should be studied.

The main threat to the *internal validity* of this study is that the functional test suite should be applied to more than only three versions of the SUT.

Regarding *external validity* of the study, we only studied one open source project (jEdit). Similar case studies and/or experiments could be performed on different systems to determine if different systems have varying or similar results in terms of maintenance and evolution nature of functional test suites. Also, only one tester (i.e., the first author) with intermediate skill-level was involved in the actual testing phase. Also, we only evaluated one functional testing tool (i.e., IRFT) and other similar tools also need to be evaluated in similar studies.

## VI. FUTURE WORK

As a future work, we plan to conduct more formal experimental studies and use statistical tests to assess different hypothesis on the subject. Also, we plan to repeat our study on other SUTs and using other testing tools.

## ACKNOWLEDGMENT

This work was supported by the Discovery Grant no. 341511-07 from the Natural Sciences and Engineering Research Council of Canada (NSERC).

## REFERENCES

- [1] jEdit Community, <http://www.jedit.org>, Last accessed: July 2008.
- [2] jEdit SourceForge page, <http://sourceforge.net/projects/jedit/>, Last accessed: Feb. 2010.
- [3] C. Schwaber and M. Gilpin, "Evaluating Automated Functional Testing Tools," *Forrester Research*, 2005.
- [4] V. Basili, G. Caldeira, and H. D. Rombach, "The Goal Question Metric Approach," in *Encyclopedia of Software Engineering*, Wiley, 1994.
- [5] IBM, "IBM Rational Functional Tester," <http://www.ibm.com/software/awdtools/tester/function>, Last accessed: Feb. 2010.
- [6] G. E. Kaiser, D. E. Perry, and W. M. Schell, "Infuse: Fusing Integration Test Management with Change Management," *Proc. of Int. Computer Software and Applications Conference*, pp. 552-558, 1989.
- [7] M. Skoglund, "A Case Study on Regression Test Suite Maintenance in System Evolution," *Proc. of Int. Conf. on Software Maintenance*, pp. 438-442, 2004.
- [8] S. P. Reiss, "Constraining Software Evolution," *Proc. of Int. Conf. on Software Maintenance*, pp. 162-171, 2002.
- [9] S. Elbaum, D. Gable, and G. Rothermel, "The Impact of Software Evolution on Code Coverage Information," *Proc. of Int. Conf. on Software Maintenance*, pp. 170-179, 2001.
- [10] A. M. Memon and Q. Xie, "Studying the Fault-Detection Effectiveness of GUI Test Cases for Rapidly Evolving Software," *IEEE Trans. on Softw. Eng.*, vol. 31, no. 10, pp. 884-896, 2005.
- [11] M. J. Harrold, J. A. Jones, T. Li, D. Liang, A. Orso, M. Pennings, S. Sinha, S. A. Spoon, and A. Gujarathi, "Regression Test Selection for Java Software," *Proc. of Int. Conference on Object - Oriented Programming Systems Languages and Applications*, pp. 312-326, 2001.
- [12] S. Mcconnell, "Daily Build and Smoke Test," *IEEE Software*, vol. 13, no. 4, pp. 143-144, 1996.
- [13] L. Briand, "A Critical Analysis of Empirical Research in Software Testing," *Keynote address, Int. Symp. on Empirical Software Engineering and Measurement*, 2007.
- [14] A. W. Brown, S. Iyengar, and S. Johnston, "A Rational Approach to Model-driven Development," *IBM Systems Journal, Model-Driven Software Development*, vol. 45, no. 3, 2006.
- [15] N. Y. Zhao and M. W. Shum, "Technical Solution to Automate Smoke Test Using Rational Functional Tester and Virtualization Technology," *Proc. of Int. Conf. on Computer Software and Applications Conference*, pp. 367-367, 2006.
- [16] jEdit SourceForge page, <http://sourceforge.net/projects/jedit/>, Last accessed: August 2008.
- [17] V. Garousi and Y. Shewchuk, "Support and experimental data for the jEdit Functional Testing project," [http://www.softqual.ucalgary.ca/projects/2008/jedit\\_functional\\_testing/](http://www.softqual.ucalgary.ca/projects/2008/jedit_functional_testing/).
- [18] Open Source Testing forum, "Open Source Functional Testing Tools," <http://www.opensourcetesting.org/functional.php>, Last accessed: Feb. 2010.
- [19] "Metrics tool (version 1.3.6)," <http://metrics.sourceforge.net>, Last accessed: July 2008.

# A Multiagent System for Automated Detection and Diagnosis of Active Tuberculosis on Chest Radiograph and CT Thorax

Abdel-Halim Hafez Elamy<sup>1,3</sup>, Behrouz H. Far<sup>2</sup>, Richard Long<sup>1,3</sup>

<sup>1</sup>Department of Public Health Sciences, School of Public Health, University of Alberta, Canada

<sup>2</sup>Department of Electrical and Computer Engineering, University of Calgary, Canada

<sup>3</sup>Department of Medicine, Faculty of Medicine and Dentistry, University of Alberta, Canada

<sup>1</sup>elamy@ualberta.ca, <sup>2</sup>far@ucalgary.ca, <sup>3</sup>richard.long@ualberta.ca

## ABSTRACT

Tuberculosis (TB) is a chronic and communicable disease that mostly affects the lungs, but can also affect other organs. TB is a leading cause of morbidity and mortality in adults worldwide, killing almost 2 million people every year. In the USA and Canada, TB is often ignored or misdiagnosed in emergency departments (EDs). This is due to the fact that it is relatively rare in North America. However, it remains prevalent in minority groups, including immigrants from TB endemic countries. The early detection of TB is a key factor to improve its treatment and interrupt its spread. In this paper, we introduce an integrated computer-aided detection (CAD) system for detecting infectious cases of TB on plain chest radiographs by means of an intelligent multiagent system (MAS). This system would be of great value to protect population health.

**Keywords**— Tuberculosis, Computer-aided detection (CAD), multiagent system (MAS), diagnostic imaging, feature detection.

## 1. INTRODUCTION

In the USA and Canada, TB is often not considered in the differential diagnosis (a process of weighing the probability of one disease versus that of other diseases that are possibly accounting for a patient's illness) of the presenting clinical symptoms at EDs. The likelihood of disease resulting from such places is extremely high because the newly infected person will have a weakened immune system due to his/her illness. Also, the health care providers can develop the disease and inadvertently pass it to their patients.

Computer Aided Detection (CAD) is an emerging technology that aims to enhance the quality and productivity of radiologists' tasks by improving the accuracy and consistency of radiologic interpretation, as well as reducing the image reading time. CAD systems can also outweigh human limitations in identifying many diseases in earlier stages. In consequence, devising an automated CAD system that detects and diagnoses TB from plain chest radiographs and triggers warning devices to announce for the presence of active TB would be of great value to public health.

### 1.1 TB Definition and Transmission

TB is a common and potentially life threatening contagious disease caused by *Mycobacterium tuberculosis* (MTB). Most commonly, TB affects the lungs but it can also affect the central nervous system, the lymphatic system, the circulatory system, the genitourinary system, bones, joints, and even the skin [1]. Symptoms of active TB include a persistent cough that lasts several weeks, weight loss, anorexia, fever, night sweats, and hemoptysis. A person with active TB in his or her lungs or throat can transmit the bacteria

to others. At the beginning, TB infection is "latent," meaning that the person has the TB-causing bacteria but it is "dormant" or inactive. However, latent TB can progress to "active" TB. Active TB means that the TB bacteria are multiplying and spreading in the body. Because MTB bacteria spread through the air, when a person with active pulmonary TB coughs, sneezes, spits or even talks, tiny droplets that contain the germs are released and can be inhaled by anyone in the area. The germs inhaled through the nose and mouth reach the windpipe and the dividing air tubes that lead to the lungs. The germs can also spread from the initial location in the lungs to other parts of the body through the bloodstream. When a contagious person is identified isolation precautions should be applied to avoid spreading the disease. A person only needs to breathe in a small number of these bacteria to become infected. Infecting bacteria that come from a person who has drug-resistant TB may result in drug-resistant TB in the infected person.

### 1.2 Diagnosis and treatment of TB

The plain chest radiograph is a widely available, inexpensive, and safe screening tool for active TB; however, it sometimes lacks sensitivity and specificity as compared to CT thorax. If pulmonary TB is suspected then the definitive diagnostic test is sputum microscopy and culture.

Sputum is the phlegm that we cough out of the lungs or throat. If a person has symptoms of pulmonary TB, several samples of the sputum, usually three, will be taken and sent to the laboratory. They will then be looked at under a microscope to check if TB bacteria can be seen. If the acid-fast bacteria – AFB can be seen under the microscope, this is called sputum smear positive TB and the person expectorating such sputum is more likely to pass on the infection to others. If the bacteria cannot be seen under the microscope, but cultures are positive for MTB, this is called sputum smear negative TB, which means that the person is less likely to pass on the infection to others. After diagnosing TB, physicians must: 1) decide whether the patient is infectious and requires respiratory isolation; and 2) determine what drugs will be most effective against the particular strain of the TB bacteria. To avoid drug resistance, a combination of antibiotics is usually prescribed.

### 1.3 Radiology of TB

Pulmonary TB is classically divided in 2 main stages: (i) primary; and (ii) postprimary or reactivation. Primary TB is the most common form of pulmonary TB in infants and children. It follows recent exposure and infection. During this stage, the location of seeding in the lung is variable, but often, the middle and lower lung zones are first involved. Hilar and/or mediastinal adenopathy along with a focal area of consolidation typically well defined, homogeneous, and segmental are the usual radiographic abnormalities. Primary TB is not

considered infectious and is difficult to diagnose with confidence from chest radiographic findings alone. Postprimary TB is almost always a disease of adolescence and adulthood. A few cases occur in children younger than 10 years [2]. This type of TB usually results from either reactivation of a remote infection, or in a minority of cases, from reinfection of a previously sensitized host. It may be possible to suspect postprimary TB from chest radiographs. Primary and postprimary TB encompass different pathological processes that result in some differences in their respective radiographic presentation [3]. Fortunately, there are some distinguishing features for postprimary TB; for instance, cavitation. Cavitation is looked at as the hallmark of postprimary disease that usually indicates the presence of active TB [4]. It occurs as the result of caseous necrosis and is usually associated with high concentration of mycobacteria. Radiographic features of TB cavities include their upper lung zone location, intermediate wall thickness (cavity walls of cavities are thicker than those of cysts), rounded configuration, and absence of an air-fluid level. Figure 1 shows a plain chest radiograph and CT evidence of cavitation.

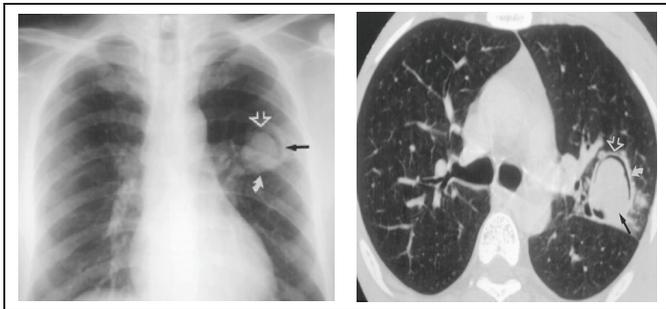


Figure 1. Frontal x-ray and axial computer tomography (CT) radiographs show a cavitory lesion. The presence of cavities on chest x-ray indicates that the patient is more likely to be infectious [5].

## 2. RELATED WORK

Considerable work has been done and presented in the literature to address the topic of computer-aided diagnosis. Various image processing methods have been introduced and adopted. A few CAD systems have been developed and commercialized. Nevertheless, none of these systems covers the application of CAD in automating the detection and diagnosis of TB in an integrated approach.

*Kakeda et al.* [6] developed a Temporal Subtraction Technique with which a previous chest radiograph is subtracted from a current one in order that interval changes are extracted and enhanced. This technique is useful for the interpretation of chest radiographs to alert radiologists to the location of newly developed abnormalities.

*Ramana et al.* [7] present an artificial neural network system to detect pulmonary TB from Mass Miniature Radiographs (MMR). This system adopts two modules: (1) an image processing module receives the digital image of a chest radiograph to resize it, enhance its quality, reduce its noise, and extracts its TB features; and (2) a neural network module that receives the extracted features and converts them into a gray-scale format that will be given to neural network for training, by applying a back-propagation algorithm. A knowledge base system is used to interpret the processed results.

*Bae et al.* [8] developed an automated CAD system, which is based on a lung nodule detection algorithm to detect nodules of 3 different types— isolated, juxtapleural, and juxtavascular— in terms of their locations or adjacent anatomic structures. This study showed that CAD is useful for detecting pulmonary nodules, including small ones with high sensitivity and a low false-positive detection rate.

*Yoshida et al.* [9] proposed a method for the discrimination of nodules from false positives in digital chest radiographs by means of a wavelet snake technique.

*Van Ginneken et al.* [10] presented a method for detecting abnormalities in frontal chest radiographs and aggregating them into an overall score. In this method, abnormal signs of a diffuse textural nature (e.g., in mass chest screening of TB) are detected.

*Mundy et al.* [11] patented a system for detecting lung diseases, such as bronchitis, emphysema and lung cancer from x-ray and CT images. They attempted to automate lung cancer and emphysema detection in CT scans by adopting a variety of nodule detection and classification techniques, along with lung parenchyma metrics.

*Phuong et al.* [12] described a diagnosis system for TB, called TUBERDIAG. This system uses of a set of IF-THEN rules for the diagnosis of TB where each rule assigns its weight on a scale of (0, 1). The inference module of the system produces the final diagnosis as a degree of confirmation/exclusion of diagnosis. TUBERDIAG adopts of *Knowledge Acquisition* component to update symptoms and expert's rules such that a user can add, delete or modify symptoms or rules in the knowledge base, a *Question and Answer* component to communicate with user to select the appropriate answers for TB symptoms, Diagnosis, and Explanation.

*Giger et al.* [13] proposed the Difference-Image technique, a method in which the original image is filtered twice: Once with a spherical kernel to obtain a nodule-enhanced image, and once with a median filter to obtain a nodule suppressed image. The candidate nodules are then obtained by subtracting the two filtered images.

Artificial neural networks (ANNs) have been recently applied to a variety of pattern recognition and data classifications in medical imaging such as chest radiography, chest CT, and mammography [14]. ANNs are well suited to problems with a high degree of complexity for which there is no algorithmic solution or the solution is too complex for traditional techniques to determine. ANNs use training data to "learn" the solution to the given problem by example. Thus, it is not necessary for the data to be complete or to have a clear trend. ANNs can still converge to a solution under these conditions. Unlike humans, ANNs are not affected by fatigue, emotional state, and working conditions [15].

## 3. RESEARCH PROBLEM AND CHALLENGES

### 3.1 Motivation

The current approach of CAD research is somewhat disjointed. *Summers* [16] argues that researchers tend to identify a clinically useful problem, and then a large portion of the research community devotes their effort and time to solve it. For example, lung cancer detection; a large portion of research has been devoted to proposing CAD systems for diagnosing lung cancer, especially by detecting and assessing lung nodules. At the same time, other important lung diseases; such as TB that have both individual and public health implications are neglected. This approach of research can actually contribute to creating two major drawbacks: (1) neglecting a wide range of clinically important problems amenable to CAD; and (2) developing CAD products with a limited scope of applicability. For these reasons, some researchers have registered concern about the accuracy, practicability, and acceptance of CAD systems [17, 18].

This conclusion prompts us to consider a broader view of the radiologists' needs and the clinicians' expectations in order to augment traditional methods by an advanced CAD system. Achieving this target requires adding some kind of intelligence to the prospective system to make it reliable and capable to perform all the necessary tasks that match our scope of research objectives in controlling the disease by generating warning signals that guide on

suspicious cases promptly. An integrated and intelligent approach would provide a more balanced and productive solution [22].

### 3.2 Challenges

#### 3.2.1 Recovering the Lack of Accuracy in Manual Interpretation

Due to the breadth of possible overlapping abnormalities that are identifiable in radiographic images, radiographic diagnostic decision making is difficult, and in most cases is a subjective task. Double reading is a proved medical practice for reducing diagnosis error, and consequently improving the detection accuracy. However, small lung features (e.g., small nodules) are often missed in the interpretation. *Kakeda et al.* [6] claim that radiologists may fail to detect lung nodules in up to 30% of cases with actual positive diagnoses. Nodules that are difficult to detect are reported to be small [19], of low contrast, and overlapping with normal anatomic structures [20]. Erroneous interpretation of radiographs results from human subjectiveness, fatigue, and limited accuracy.

#### 3.2.2 Discrete Radiographic Patterns

There are various difficulties associated with the detection of chest radiographic abnormalities that suggest the presence of pulmonary TB. The most apparent difficulty is the disconnected patterns of abnormality. We could, for example, have signs of nodules or cavities with a wide range of sizes- from few millimeters up to several centimeters- with large variation in density, and hence visibility on radiographs. Some abnormalities can be slightly denser than the surrounding lung tissues, while the densest ones are calcified. Moreover, some features can be obscured by other organs, such as ribs, mediastinum, and structures below the diaphragm. Another difficulty is the possibility of overlapping the detection and diagnosis of TB with other lung diseases. We suggest 2 independent means for overcoming these difficulties. First: by focusing on distinguishing abnormalities that lead to the detection of smear positive TB. Second: by providing intelligent mechanisms that can overcome the presence of similarity to accept undefined patterns.

## 4. MAS TB-CAD SYSTEM

Our approach is based on utilizing an innovative methodology that adopts object and task orientation software engineering (SE) logic through a modular approach for optimal interaction and parallel implementation in an effective development life cycle. The system architecture is illustrated in Figure 2. In this system, the Feature Detection module will receive its input as a set of (1) two scanned CXR images; one anterior, and one lateral or (2) a set of CT-Scan images. The system will process these images to extract important radiographic features of infectious TB (i.e., cavity, adenopathy, acinar shadows, volume loss, etc.) by means of image segmentation and intelligent annotation. The output of this module will be delivered in XML format and passed to the TB Diagnosis module to process it by means of artificial intelligence techniques to deliver an automated diagnosis that answers the question “Is it infectious TB or not”. Moreover, this module will take advantage of two other inputs: (1) patient-related information (age, Aboriginal status if Canadian/American-born, country-of-birth if foreign-born, symptoms and their duration, contacts to TB transmitters, and risk factors for reactivation); and (2) autonomously learnt knowledge that will be created by examining patterns and diagnosis stored in the TB annotated image repository.

Two submodules will be involved in the diagnosis process: (1) the Bayesian Belief Network (BBN) sub-module will be used for coarse diagnosis and probabilistic reasoning; and (2) the Case-based Reasoning (CBR) submodule will be used for confirmative diagnosis. CBR is the process of solving new problems based on the solutions of similar past problems using analogy. It has been argued that CBR is

not only a powerful method for automated reasoning, but also a pervasive behavior in everyday human problem solving; or, more radically, that all reasoning is based on past cases personally experienced. CBR starts with a set of cases or training examples; it forms generalizations of these examples by identifying commonalities between a retrieved case and the target problem.

CBR is effective when the problem has records of previously solved problems exist and specialists describe their domain by giving examples. CBR is often used where experts find it hard to articulate their thought processes when solving problems. This technique allows the case-base to be developed incrementally, while maintenance of the case library is relatively easy and can be conducted by domain experts. BBN is used to model domain containing uncertainty. Conditional probability table (CPT) of a node contains probabilities of the node being in a specific state given the states of its parents. Probabilities can be assessed using a combination of theoretical insight, empiric studies independent of the constructed system, training and various more or less subjective estimates. BBN provides advantages such as be able to read the uncertainty of the conclusion from the network, be able to get the next-most probable diagnosis and - probably the most important - know under which assumptions about the domain the suggested diagnosis is the most probable.

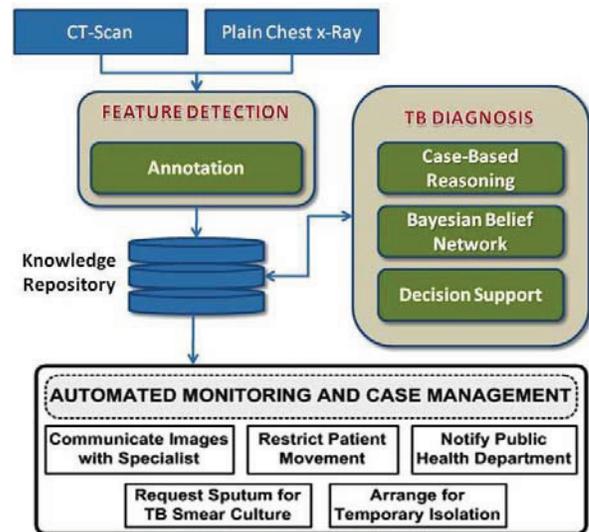


Figure 2. Conceptual architecture of the project modules

Finally, the Automated Monitoring and Case Management module takes the output of the TB Diagnosis module as an input. If there is a positive sign of infectious TB incident, a set of automated monitoring and management actions will take place by triggering multimedia devices to generate special warnings to health care officials (see Figure 2). Agent platform is used to implement the system. *Elamy and Far* [21] define an agent as a software entity that possesses special intelligent properties, such as autonomy, reasoning, mobility, sociability, learning ability, cooperation, and negotiation. Such intelligent properties allow agents to initiate their actions without the need of direct intervention or guidance of humans or other entities, and to interactively cooperate and communicate with each other and with their environment to accomplish special tasks that cannot be performed by conventional software. Figure 3 depicts a high level architecture for the MAS TB-CAD system. The system is composed of 4 layers each dedicated to a particular set of jobs within the CAD system. The units shown in ovals are agents each having a set of dedicated roles (tasks) and appropriate expertise to handle the roles assigned to them.

The source layer is an embodiment of various data sources currently available, including the annotated x-ray image database, or anticipated to be added later, such as the patient history database. A challenge in MAS TB-CAD is that the data may have been collected at various times in various formats and new data resources may popup as more needs are identified or due to advancement of technology. The data guide agents can handle the data diversity in a seamless way. The agents in the acquisition and integration layer can learn from or mine the data in a goal-oriented way. The goal orientation is a central concept. The goal usually comes from the higher levels such as the user or decision or access layers. A goal specifies a meaningful target to be analyzed either individually (e.g., detailed examination of a certain point in an x-ray for a given patient case) or as a group (e.g., group discussion regarding a patient case). The data integration agent has the role of providing the knowledge acquisition agent with the appropriate data. The local database and the knowledge base are task specific resources. The agents in the decision layer implement the decision making algorithms as well as the visualization of data. The personal assistant agent in the access layer is a thin application residing on the radiologist or technician's computing environment. Several users can interact with the system via wired or wireless connections. The architecture is open and new agent types with appropriate expertise can be added when needed.

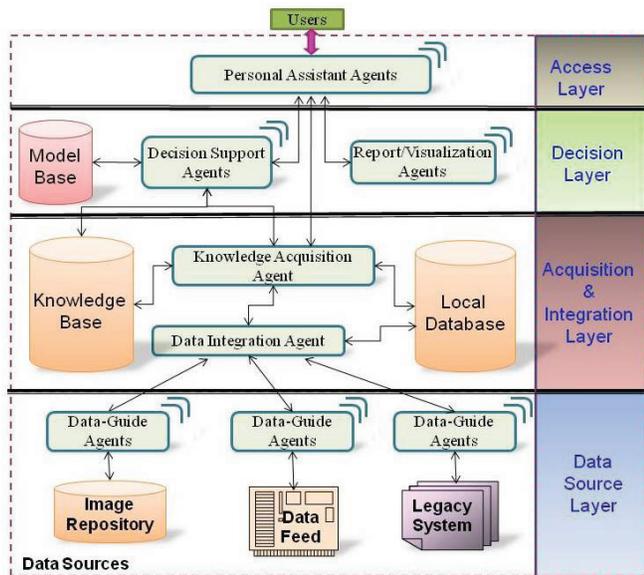


Figure 3. MAS TB-CAD architecture

## 5. CONCLUSIONS

In this paper, we exhibited the most recent and significant advances in the area of automating the detection of the radiographic features of pulmonary diseases, such as lung cancer and tuberculosis (TB). Our study concluded that the performance of the existing CAD systems varies significantly. This is probably due to the variability among research groups and datasets. In addition, most groups did not have enough number of instances to assess their approaches. We ended up by proposing an integrated CAD system that can be implemented to detect infectious TB from plain chest radiographs. We have already developed some components of this system. The system is built by means of intelligent MAS. Some monitoring devices can be linked and triggered automatically to warn for potential incidence of smear positive TB. In order to have a significant impact on the population's health, a CAD system should pass the scope of research towards the scope of commercialization.

The proposed system would be of great value in protecting residents from the spread of serious TB contagion.

## REFERENCES

- [1] M. Guillemin, M. Usdin, J. Arkinstall, Tuberculosis Diagnosis and Drug Sensitivity Testing: An overview of the current diagnostic pipeline, Campaign for Access to Essential Medicines, Paris, 2006.
- [2] J. Shewchuk and M. Reed, Pediatric Postprimary Pulmonary Tuberculosis, *Pediatr Radiol*, vol. 32, 2002, pp. 648–651.
- [3] S. Ellis, The Spectrum of Tuberculosis and Non-Tuberculous Mycobacterial Infection, *Eur Radiol*, vol. 14, 2004, pp. E34–E42.
- [4] C. Daley, M. Gotway, R. Jasmer, Radiographic Manifestations of Tuberculosis, 2nd ed., Francis J., CNTC, USA, 2002.
- [5] M. Harisinghani, T. McLoud, J-A. Shepard, J. Ko, M. Shroff, P. Mueller, Tuberculosis from Head to Toe, RadioGraphics, vol. 20, 2000.
- [6] S. Kakeda, K. Nakamura, K. Kamada, H. Watanabe, H. Nakata, S. Katsuragawa, K. Doi, Improved Detection of Lung Nodules by Using a Temporal Subtraction Technique, *Radiology*, 2002.
- [7] K. Ramana and S. Basha, Neural Image Recognition System with Application to Tuberculosis Detection, Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04), vol. 2, 2004, pp. 694-699.
- [8] K. Bae, J-S. Kim, Y-H. Na, K. Kim, J-H. Kim, Pulmonary Nodules: Automated Detection on CT Images with Morphologic Matching Algorithm—Preliminary Results, *Radiology*, vol. 236, 2005.
- [9] H. Yoshida, S. Katsuragawa, Y. Amit, K. Doi, Method, Apparatus, and Storage Medium for Detection of Nodules in Biological Tissue Using Wavelet Snakes to Characterize Features in Radiographic Images, US Patent 6,078,680, 2000.
- [10] B. van Ginneken, S. Katsuragawa, B. ter Haar Romeny, M. Viergever, Automatic Detection of Abnormalities in Chest Radiographs Using Local Texture Analysis, *Medical Imaging*, vol. 21, no. 2, 2002.
- [11] J. Mundy, C. McCulloch, R. Avila, S. Hastings, R. Kaucic, W. Lorensen, M. Turek, Method and System for Lung Disease Detection, US Patent 09, 683,111, 2006.
- [12] N. Phuong, D. Hung, D. Tuan; N. Viet-Co, B. Duong, P. Thinh, N. Hoa, and N. Linh, Designing an Experimental Expert System for Lung Tuberculosis Diagnostics Using Fuzzy Set Theory, *IEEE*, 1998.
- [13] M. Giger, K. Doi, H. MacMahon, Image Feature Analysis and Computer-Aided Diagnosis in Digital Radiography: Automated Detection of Nodules in Peripheral Lung Fields, *Medical Physics*, vol. 15, no. 2, 1988.
- [14] W. Penny and D. Frost, Neural Networks in Clinical Medicine, *Medical Decision Making*, vol. 16, no. 386, 1996.
- [15] Fukushima, K. Ashizawa, T. Yamaguchi, N. Matsuyama, H. Hayashi, I. Kida, et al., Application of an Artificial Neural Network to High-Resolution CT: Usefulness in Differential Diagnosis of Diffuse Lung Disease, *AJR*, vol. 183, 2004.
- [16] R. Summers, Road Maps for Advancement of Radiologic Computer-aided Detection in the 21st Century, vol. 229, 2003, pp. 11-13.
- [17] J. Ridderikhoff and B. van-Herk, Who is afraid of the system? Doctors' attitude towards diagnostic systems, *International Journal of Medical Informatics*, vol. 53, 1999, pp. 91–100.
- [18] G. Sutton, How Accurate is Computer-aided Diagnosis, *Lancet*, 1989.
- [19] Kelsey Brogdon, R. Moseley, Factors Affecting Perception of Pulmonary Lesions, *Radiol Clin North Am*, vol. 21, 1983.
- [20] G. Revesz, H. Kundel, M. Graber, The Influence of Structured Noise on the Detection of Radiologic Abnormalities, *Invest Radiol*, 1974.
- [21] A. Elamy and B. Far, "On the Evaluation of Agent-Oriented Software Engineering Methodologies: A Statistical Approach," Lecture Notes in Artificial Intelligence (LNAI) series, Agent Oriented Information Systems IV, M. Kolp, A. Garcia, A. Ghoze, P. Bresciani, B. Henderson-Sellers, and F. Mouratidis (eds.), vol. 4898, ISBN 978-3-540-77989-6, Springer-Verlag, Heidelberg, Germany, 2008.
- [22] A. Elamy, M. Mandal, B.H. Far, A. Basu, I. Cheng, R. Long, An Intelligent CAD System for Automated Detection of Pulmonary Tuberculosis on Chest Radiograph and CT Thorax: A Road Map, proceedings of the IEEE - CCECE 2010, Canada, 2010.

# Impact Analysis Model for Brasília Area Control Center using Multi-Agent System with Reinforcement Learning

Antonio Carlos de Arruda Junior  
Alessandro Ferreira Leite  
Cícero Roberto Ferreira de Almeida  
Alba Cristina Magalhaes Alves de Melo  
Li Weigang

Laboratory of Group of Computational Model in Air Transport - TransLab  
Department of Computer Science, University of Brasília - UnB, Brazil  
{jnarrd, alessandro.leite, cicero.almeida}@gmail.com {alves, weigang}@unb.br

## Abstract

*This paper describes a methodology based on multi-agent theory for Air Traffic Flow Management (ATFM) problem. We modeling a function that take account the impact of traffic managers actions, such as, fairness in the distribution of delays, and financial cost. This model is integrated with a Decision Support System Applied to Tactical Air Traffic Flow Management (SISCONFLUX). The architecture was implemented based on multi-agent system using reinforcement learning. The objective is to ensure the safety and prevent or reduce congestions in diverse sectors within the airspace compute the financial cost and impact of the delay generated on the aircraft. Experimental results using real data from Brasília's Flight Information Region (FIR-BS) show that the delay time is 25% less than the results computed only with Graph Theory and fairness factor and financial cost can be used together with congestion data, without affecting safety and traffic flow.*

## 1. Introduction

Air Traffic Flow Management (ATFM) is considered an extremely complex task, highly specialized and heavily based on the experience of the traffic manager [4], where the tasks should address critical issues such as efficiency (fluency and reduced delays), fairness (working with different airlines), adaptability (dealing with weather conditions), trust and security (managing airports).

The main objective of the Traffic Flow Management is to ensure flight regularity, effectiveness and safety with an appropriate balance between demand and the available aero-

nautical and airport structure, according to meteorological conditions and aircraft operational restrictions [3].

The Brazilian airspace is divided into five Flight Information Regions (FIR). This paper analyzes the Brasília FIR which is coordinated by the First Integrated Center of Aerial Defense and Airspace Control - CINDACTA I which control about fifty per cent of the entire Brazilian air traffic flow.

To coordinate the safe and efficient flow of these aircraft, the centers of air traffic control (ATC) meet with many challenging situations that are specific and adverse that may lead to a state of saturation [6].

With the objective of preventing congestion in sectors air traffic managers apply restrictive actions such as, ground holding, waiting in flight, reducing velocity, alternative routes, indication in curve for delays in flight and landing and waiting in intermediate aerodrome.

However, every action generates a cost applied to air scenario as a whole, where potential delays or even cancellation of flights affecting the origin and destination airports.

To assist in the optimization of existing processes, researchers worldwide have studied and suggested various solutions, which employ techniques ranging from dynamic programming to multi-agent systems.

In [1] the authors built and tested a prototype of multi-agent system for managing congestion in a simulated air environment. The agents are defined as fixed points (radar) in the air scenario and take actions based on restrictive measures MIT (Miles in Trail). The results show an improvement in the system-wide performance by up to 20% over human traffic manager. However the fairness about the restrictive actions is not checked by agent suggestions.

In [2] the authors investigated techniques to achieve fairness between airlines schedules distributing the impact of

restrictive actions between aircraft of the system. The system was modeled through the dynamic programming and defines a new concept called Pairwise Reversals, that uses information about the maximum permissible delay for a flight. Their results show a small increase of 10% in the total delay, with the advantage of increasing the fairness and satisfaction among airlines.

The research proposed in this paper uses specific points of above works, with different and innovative contributions that will be presented in the next sections.

The remainder of this paper is organized in the following manner: Section 2 presents the modeling of the agents and the environment. In Section 3, the architecture of the system is described. Section 4 shows the experimental results of the prototype. Finally, in section 5 the final consideration of the research and the relevant references are presented.

## 2 Agent Modeling

The model uses agents that obtain experience from the environment through online information (radar) about aircraft that are impacting the air scenario.

### 2.1. Q-Table Construction

The structure of reinforcement learning used is based on Q-Learning algorithm, proposed by Watkins [5]. In this algorithm, the agent uses a matrix (Q-Table) to store its experiences with the environment.

The Q-Table was modeled as  $s \times a$ , where  $s$  represents the states that characterize the environment and  $a$  represents the actions suggests by agent, as show Figure 1.

Q	$A_1$	$A_2$	$\dots$	$A_m$
$S_1$	$Q(s_1, a_1)$	$Q(s_1, a_2)$	$\dots$	$Q(s_1, a_m)$
$S_2$	$Q(s_2, a_1)$	$Q(s_2, a_2)$	$\dots$	$Q(s_2, a_m)$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$S_n$	$Q(s_n, a_1)$	$Q(s_n, a_2)$	$\dots$	$Q(s_n, a_m)$

Figure 1. Q-Table: agent's rewards

When condition of saturation or congestion is detected the agent suggest delay time as restrictive actions to be applied.

### 2.2 Reward Structure

The reward structure was modeled as a function composed by three terms: congestion, delay time, and financial cost and fairness as show in Equation 1.

$$G(z) = \beta C(z) + \delta T(z) + \gamma I(z) \quad (1)$$

In Equation 1,  $C(z)$  and  $T(z)$  represents congestion and delay time, as defined in [1],  $I(z)$  the financial cost and fairness of the restrictive action and  $\beta$ ,  $\delta$ , and  $\gamma$  are the weights of each term.

### 2.2.1 Evaluation Function: Costs and Fairness

For airline, the notion of fairness is to have a schedule that preserves the order of flight arrivals at an airport according to the published schedules in the online Airline Guide (OAG) [2].

The agent use Equation 2 to analyze the impact of a restrictive action.

$$I(z) = \sum_{s \in z} J_{Ad}(s) + F_{Ath}(s) \quad (2)$$

In Equation 2,  $J_{Ad}(s)$  is the delay imposed by a restrictive action given by equation 3, and  $F_{Ath}$  is the financial cost in apply the restrictive action, given by Equation 4.

$$J_{Ad}(s) = \sum_{a \in s} \left| 100 - \left( \frac{d_{Agh}(a) * 100}{d_{Td}/size(Term, t_{ini}, t_{end})} \right) \right| \quad (3)$$

Where,  $d_{Agh}$  is the delay of the aircraft  $a$  on the ground, in minutes,  $d_{Td}$  is total delay in a terminal, and  $size$  is a function that returns the number of aircraft that have been delayed over a period of time, between  $t_{ini}$  and  $t_{end}$  in a terminal.

$$F_{Ath}(s) = \sum_{a \in s} \theta(d_{Ath}(a) - D(a))^{1+\theta(d_{Ath}(a)-C(a))} \quad (4)$$

In Equation 4,  $C$  is a variable defined in terms of the maximum time that an aircraft may delay, before being canceled,  $D$  is a variable defined in terms of the maximum delay that an aircraft may suffer no harm other flight as defined in [2],  $d_{Ath}$  the total delay of the aircraft and  $\theta$  is a function that return zero, if the aircraft is in time.

## 3 Architecture

The architecture presents the development of a modular system, proposed by [6].

The input data of system is composed by radar, repetitive flight plans (RPL) and eventual flight plans (FPL), managed by System of Data Treatment and Visualization (STVD) and Visualization System and Data Processing of the Aerial Navigation Management Center (SYNCROMAX) [4].

The Module of Monitoring and Scenario Forecast (MAPC) get this data and put them in a format recognized by SISCONFLUX, modeling current air scenario and the future position of the aircraft on the sectors.

When future congestions are detected by the system, this information is getting by Module of Flow Balancing (MBF) and actions based on ground holding problem (GHP) are calculated and analyzed by the agents. These actions are suggested to traffic managers, that choose an action to resolving the problem. So, the Module of Evaluation and Decision Support (MAAD) analyze the result scenario, and update the agents experience [6].

The proposed architecture can be found in figure 2, where CGNA is Center of National Aviation Management.

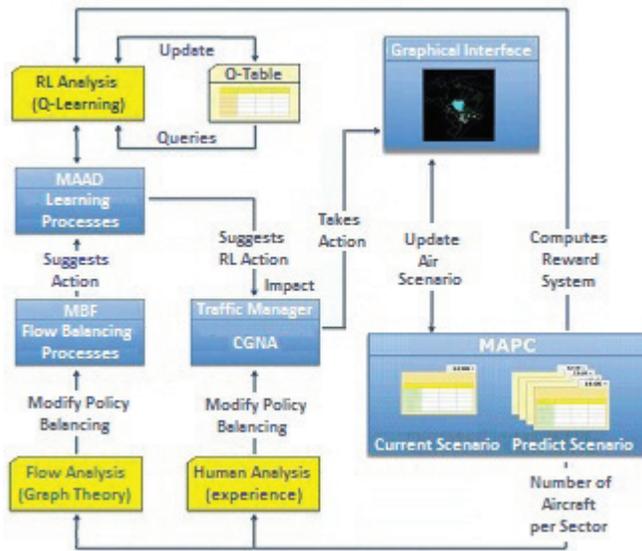


Figure 2. Architecture of MAAD

The concept of this architecture represents the human traffic manager that make decisions suggested by agents. Thus, the system is characterized as a semi-automatic control of flow actions.

## 4 Experimental Results

To analyze the behavior of the evaluation function proposed in section II, experiments were performed with data obtained directly from CGNA and CINDACTA I.

The implementation of the prototype was done in a controlled and supervised environment using a fixed number of cycles. The results of simulation shown the comparison of two evaluation functions used, with a learning convergence nearly 300 cycles about.

### 4.1 Case 1: High Movement Situation

The analysis is studied by comparison between the results produced by the evaluation function that checks only security and the flow of aircraft, called *Congestion function*,

and the function that checks addition to the above aspects, the impact of the actions, called *Impact function*.

The figure 3 shows the result of simulation with congestion occurring in São Paulo's Terminal, where they were observed for the two functions compared.

Terminal	Restrictive Action without Impact Terms	Restrictive Action with Impact Terms	Result Analysis	Air Movements
TMA-SP	CY(20,25) SP(15,20)	CY(20,20) SP(15,20)	delay reduced	1339

Figure 3. Sao Paulo's Terminal: best restrictive actions for the evaluation function

The Congestion function in the second column indicates that to reduce the congestion in the São Paulo's Terminal (TMA-SP), a delay between 20 and 25 minutes should be applied in Cuiabá's Terminal (CY 20, 25) and one between 15 and 20 minutes in São Paulo's Terminal (SP 15, 20).

When using the Impact function, shown in the third column, the delays were reduced in Cuiabá's Terminal (CY 20, 20). This action reduces the delay and cancellations possibility in a terminal, and also the value of the index of financial impact and fairness. This occurs because the level of congestion was relatively low, with up to 13 aircraft in the sector.

### 4.2 Case 2: Average Movement Situation

In the Figure 4 is shown the simulation with congestion occurring in Vitoria's Terminal (TMA-VT), where it was observed a maximum of 12 aircraft.

Terminal	Restrictive Action without Impact Terms	Restrictive Action with Impact Terms	Result Analysis	Air Movements
TMA-VT	CY(5,10)	CY(5,10)	equal	863

Figure 4. Vitoria's Terminal: best restrictive actions for the evaluation function

The Congestion function manage the congestion in Vitoria's Terminal suggesting a delay between 5 and 10 minutes to the aircraft that will take off in the Cuiaba's Terminal (CY 05, 10) only. The Impact function also suggested the same restrictive action.

For this case, it indicates that when the restrictive actions are low delays and the application terminals are few, fewer aircraft will be impacted.

### 4.3 Case 3: Low Movement Situation

The situation in air scenario of Amazonica’s Terminal (TMA-MN) is that the delays were high in all cases. This happened because the terminal analysed had a small amount of flights, but the most of the flights occurred at same times. This making the situation more critical congestion, reaching 15 aircraft at any given time.

As shown in Figure 5, according to the restrictive actions suggested by the Congestion function, the traffic manager must apply between 15 and 20 minutes in Cuiaba’s Terminal (CY 15, 20), between 5 and 10 in São Paulo’s Terminal (SP 5, 10) and between 15 and 20 in Amazonica’s Terminal (MN 15, 20), to solve the problems of congestion.

Terminal	Restrictive Action without Impact Terms	Restrictive Action with Impact Terms	Result Analysis	Air Movements
TMA-MN	CY(15,20) SP(5,10) AM(15,20)	CY(30,30) SP(5,10) AM(20,20)	delay increased	467

**Figure 5. Amazonica’s Terminal: best restrictive actions for the evaluation function**

When using the Impact function, the suggested delays was 30 minutes in Cuiaba’s Terminal (CY 30, 30), between 5 and 10 in São Paulo’s Terminal (SP 5, 10) and 20 in Amazonica’s Terminal (MN 20, 20).

In this case study was identified the highest delays and the most affected terminals by them. This can be justified by the greater amount of aircraft affected, leads to a significant increase in the amount delay imposed on the environment, generating high rates of delay, where the level of fairness can be harmful and also the financial costs increase.

## 5 Final Consideration

This paper has presented a proposal to reduce congestion using a multi-agent system that uses innovative evaluate functions offering a impact analysis of air traffic flow.

The tests show that the delay time suggest by agents is up to 25% less than the time computed with function of Graph Theory.

With this approach, traffic managers can acquire knowledge that would help make better decisions on ATFM and the traditional analysis can be replaced by a new vision of impact that restrictive actions cause.

The contributions of this work are: a impact sub-module that uses Artificial Intelligence applied to the area of Decision Support Systems; a new architecture focused on the

Brazilian air scenario; restrictive actions based on ground holding problem (GHP); a new reward function that takes into account points as financial costs and fairness contributing more visibility on the impact generated by the restrictive actions applied.

As future works its recommend the studies with new functions and new restrictive actions. And it is also with the perspective to apply the developed model in CINDACTA I for the real operation.

## References

- [1] A. Agogino and K. Tumer. Learning indirect actions in complex domains: Action suggestions for air traffic control. *Advances in Complex Systems (ACS)*, 12(4):493–512, 2009.
- [2] D. Bertsimas and S. Gupta. Fairness in air traffic flow management. *INFORMS Meeting*, October 2009.
- [3] Rolim, T. H. L., de Almeida Portela and Roberto de Almeida Alves, T. O controle do espaço aéreo. Technical report, DECEA, Departamento de Controle do Espaço Aéreo. AS-COM/DECEA, 2004.
- [4] A. Timoszczuk, W. Pizzo, G. Staniscia, and E. Siewerdt. *The SYNCROMAX Solution for Air Traffic Flow Management in Brazil, Computational Models, Software Engineering, and Advanced Technologies in Air Transportation: Next Generation Applications*. Eds: Li Weigang and Alexandre de Barros and Italo Romani de Oliveira, IGI Global, USA, 2009.
- [5] C. Watkins. *Learning From Delayed Rewards*. PhD thesis, Cambridge, England, 1989.
- [6] L. Weigang, B. B. de Souza, A. M. F. Crespo, and D. P. Alves. Decision support system in tactical air traffic flow management for air traffic flow controllers. *Journal of Air Transport Management*, 14(6):329–336, 2008.

# Mobile Agents for Active Media

Ichiro Satoh

National Institute of Informatics

2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan

E-mail: ichiro@nii.ac.jp

## Abstract

*This paper presents an agent-based framework for building and operating context-aware multimedia content on digital signage in public/private spaces. The framework enables active and multimedia content to be composed from mobile agents, which can travel from computer to computer and provide multimedia content for advertising or user-assistant services for users. The framework automatically deploys their agents at computers near to their current positions to provide advertising or annotation on objects or for users. To demonstrate the utility of the framework, we show a user-assistant for shopping with digital signage.*

## 1 Introduction

Modern out-of-home (OOH) advertising is focused on marketing to consumers according to their context, e.g., when and where they are 'on the go' in public places, in-transit, or waiting. For example, in public museums, most visitors want annotations on the exhibits in front of them, because they lack sufficient knowledge about the exhibits. Their knowledge and experiences are varied so that they tend to be become puzzled (or bored) if the annotations provided to them are beyond (or beneath) their knowledge or interest. Content displayed on digital signage should be selected and adapted to according to the users' context.

This paper presents a framework for building and managing context-aware advertising in public places, in-transit, or waiting. The framework introduces the notion of counterpart objects for people or physical entities. These are digital representations of people or physical entities and can define and provide context-aware for advertising products or user-assistant services. An application should not directly interact with people or physical objects, but with their virtual counterpart objects. On behalf physical objects, their counterpart objects need to interact with each other. The framework provides each user or physical entity

with at least one counterpart object, where each counterpart object is implemented as a mobile agent. It can also spatially bind a user to its agent using location-sensing systems. For example, when a product is exhibited in a store, its agent is deployed at computer to its position to display advertising messages to sell it. When a user buys it and carries it to his/her home, the agent is deployed at a computer near to it to display how to use it.

- Content, which are played on digital signage and public terminals, can be anything, including text, images, animations, video, audio, and interactivity.
- To enhance customer experience, the framework can monitor contextual information in the real world by means of sensing systems and select and adapt content according to changes in the real world and user preferences.
- The framework supports stationary computers rather than mobile computers because customers in stores or visitors in museums should pay attention to their purpose, e.g., shopping or watching exhibition.
- It needs to provide massive users with context-aware services at massive ubiquitous and mobile computing devices in their personalized forms.
- It provides context-aware content with the ability to monitor and response to their users' behaviors, because advertising or user assistant services provided as context-aware services should adapt to the users' profiles.

The framework should provide a variety of multimedia content, including interactive content, for consumers, from digital signage or terminal in public spaces.

### 1.1 Related work

There have been several commercial projects for providing context-aware content on digital signage, but they

were constructed based on ad-hoc manners. On the other hand, several researchers have explored context-aware services independently of the literature of digital signage. Cambridge University's Sentient Computing project [2] provides a platform for location-aware applications using infrared-based or ultrasonic-based locating systems in a building. Microsoft's EasyLiving project [1] enabled services running on different computers to be combined dynamically according to contextual changes in the real world. We discuss differences between the framework presented in this paper and our previous frameworks. We constructed a location model for ubiquitous computing environments. The model represents spatial relationships between physical entities (and places) as containment relationships between their programmable counterpart objects and deploys counterpart objects at computers according to the positions of their target objects or places [3]. This was a general-purpose location-model for context-aware services, but was not an infrastructure for deploying and operating such services. We presented an outline of mobile agent-based services in public museums in our early versions of this paper [4, 5], whereas this paper addresses agent-based advertising on digital signage for shopping.

## 2 Design and Implementation

This section describes the design and Implementation of the framework.

### 2.1 Basic Approach

This framework builds and manages context-aware multimedia content on stationary computers, including digital signage and public terminals. We discuss design principle behind the framework.

- Each mobile agent is a programmable entity with data store. Therefore, each mobile agent-based services can define programs to play its visual/audio content and interact with users inside it. Therefore, the framework itself and underlying systems are independent of application-specific tasks.
- After arriving at its destination, a mobile agent can continue working without losing the results of working, e.g., the content of instance variables in the agent's program, at the source computers. Therefore, users can continue to enjoy or interact with content from computers close to their current positions, even when the users move from location to location.
- The framework deploys and execute mobile agents bound to physical entities or people at computers near

the position of the user instead of any remote servers. As a result, mobile agent-based content can directly interact with the user, where RPC-based approaches, which other existing approaches are often based on, must have network latency between computers and remote servers.

- Mobile and ubiquitous computers often have only limited resources, such as restricted levels of CPU power and the amounts of memory. Mobile agents can help to conserve these limited resources, since each agent needs to be present at the computer only while the computer needs the content provided by that agent.
- To support wide public spaces, the framework needs to be managed in a non-centralized manner. Mobile agents can be managed without any centralized servers.

### 2.2 Agent Platform

The framework consists of three parts: (1) mobile agents, (2) agent runtime systems, and (3) location information servers, called LISs (Fig. 1). The first offers application-specific content, which are attached to physical entities and places, as collections of mobile agents. The second is running on public terminals or digital signage and responsible for executing mobile agents and migrating to other runtime systems, which may run on different computers, through peer-to-peer based protocols. The third provides a layer of indirection between the underlying locating sensing systems and mobile agents. Each LIS manages more than one sensor and provides the agents with up-to-date information on the state of the real world, such as the locations of people, places, and things, and the destinations that the agents should migrate themselves to.

### 2.3 Location Information Server

Each LIS monitors multiple sensors that detect the presence of tags and maintains up-to-date information on the identities of tags that are within the zone of coverage of its sensors. This is achieved by polling sensors or receiving the events issued by the sensors themselves. An LIS does not require any knowledge of other LISs. To conceal the differences among the underlying locating systems, each LIS maps low-level positional information from each of the locating systems into information in a symbolic model of location. An LISs represents an entity's location at a few feet spaces, called *spot*, which distinguishes one or more portions of a room or building.

In the remanding paper, we assume to use RFID technology, where users and physical products or exhibits are

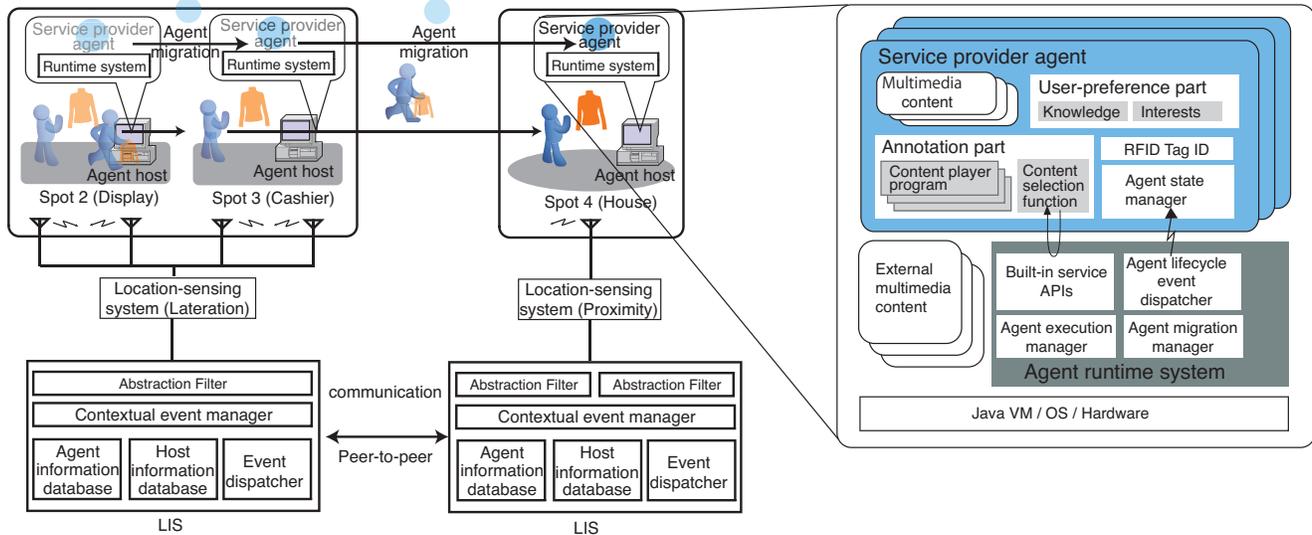


Figure 1. Architecture

attached with RFID tags. When an LIS detects a new tag in a spot, it multicasts a query that contains the identity of the new tag and its own network address to all the agent runtime systems in its current sub-network to discover agents or computers attached to the tag. When there are multiple candidate destinations, each of the agents that is tied to a tag can select one destination on the basis of the profiles of the destinations. When the absence of a tag is detected in a spot, each LIS multicasts a message with the identifier of the tag and the identifier of the spot to all runtime systems in its current sub-network.

## 2.4 Agent runtime system

Each runtime system migrate agents to other runtime systems running on different computers through a TCP channel using mobile-agent technology (Fig. 1). It is designed for only the framework and built on the Java virtual machine (Java VM) version 1.5 or later versions, which conceals differences between the platform architectures of the source and destination computers. It governs all the agents inside it and maintains the life-cycle state of each agent. When the life-cycle state of an agent changes, e.g., when it is created, terminates, or migrates to another runtime system, its current runtime system issues specific events to the agent.

When an agent is transferred over the network, not only the code of the agent but also its state is transformed into a bitstream by using Java's object serialization package and then the bit stream is transferred to the destination. Since the package does not support the capturing of stack frames of threads, when an agent is deployed at another computer,

its runtime system propagates certain events to to instruct it to stop its active threads. Arriving agents may explicitly have to acquire various resources, e.g., video and sound, or release previously acquired resources.

## 2.5 Service-provider mobile agent

Each agent is attached to at most one user or physical entity. If an agent is tied to a user, it maintains his/her preference information and programs that provide annotation inside it. If an agent is tied to a entity, it maintains information about the entity and programs to monitor or control the entity. To easily build agents, we provides built-in agents, which are dynamically assembled from *content part* and *user-preference part*.

### 2.5.1 Content part:

This part is responsible for selecting and playing annotations according to the current spot and route in addition to the information stored in the user-preference part and play the content in the personalized form of its user. It is defined as a set of content-selection function and programs for playing the selected content.

The function maps more than one argument, e.g., the current spot, the user's selected route, and the number of times he/she has visited the spot into a URL referring to the annotative content. The content can be stored in the agent, the current runtime system, or external http servers. That is, each agent can carry a set of its content, play the selected content at its destinations, directly play the content stored

at its destinations, or download and play the content stored in Web servers on the Internet. The current implementation can divide this part into three sub-parts: opening, annotation, and closing, which are played in turn.

Annotation content is various, e.g., text, image, video, and sound. The annotation part defines programs for playing this content. The current implementation supports (rich) text data, html, image data, e.g. JPEG and GIF, video data, e.g., animation GIF and MPEG, and sound data, e.g., WAV and MP3. The format for content is specified in an MIME-based attribute description. Since the annotation part is defined as Java-based general-purpose programs, we can easily define interactions between visitors and agents.

### 2.5.2 User-preference part:

This part is responsible for maintaining information about a visitor. In fact, it is almost impossible to accurately infer what a visitor knows or is interested in from data that are measured by sensing systems. Instead, the current implementation assumes that administrators will explicitly ask visitors about their knowledge and interests and manually input the information into this part. Nevertheless, it is still possible to make an educated guess with some probability as to what a visitor may be interested in, if we know which spots he/she has visited, how many he/she has visited, and how long he/she was visited. Each agent has a mechanism to automatically record the identifiers, the number of visits to, and the length of stays at spots by visitors.

### 2.5.3 Security and privacy issues

The framework only maintains per-user profile information within those agents that are bound to the user. It promotes the movement of such agents to appropriate hosts near the user in response to the user's movements. Thus, the agents do not leak profile information on their users to other parties and they can interact with their mobile users in personalized form that has been adapted to respective, individual users. The runtime system can encrypt agents to be encrypted before migrating them over a network and then decrypt them after they arrive at their destination. Moreover, since each mobile agent is just a programmable entity, it can explicitly encrypt its particular fields and migrate itself with these fields and its own cryptographic procedure. The Java virtual machine can explicitly restrict agents to only access specified resources to protect hosts from malicious agents. Although the current implementation cannot protect agents from malicious hosts, the runtime system supports some authentication mechanisms for agent migration so that each agent host can only send agents to and only receive from the trusted hosts.

## 3 Early Experience

We have several experiences of the framework in real spaces. This section describes two applications.

### 3.1 Performance evaluation

Although the current implementation was not built for performance, we measured the cost of migrating a null agent (a 5-KB agent, zip-compressed) and an annotation agent (1.2-MB agent, zip-compressed) from a source host to a destination host that was recommended by the LISs. The latency of discovering and instructing an agent attached to a tag after the CDS had detected the presence of the tag was 420 ms and the respective cost of migrating the null and annotation agent between two hosts over a TCP connection was 38 ms and 480 ms. This evaluation was operated with three computers (Intel Core 2 Duo 2 GHz with Windows XP Professional and JDK 1.5) connected via a Fast Ethernet. This cost is reasonable for migrating agents between computers to follow that visitors moving between exhibits.

The current implementation supports two commercial tracking systems. The first is the Spider active RFID tag system, which is a typical example of proximity-based tracking. It provides active RF-tags for users. Each tag has a unique identifier that periodically emits an RF-beacon (every second) that conveys an identifier within a range of 1-20 meters. The second system is the Aeroscout positioning system, which consists of four or more readers located in a room. These readers can measure differences in the arrival timings of WiFi-based RF-pulses emitted from tags and estimate the positions of the tags from multiple measurements of the distance between the readers and tags; these measurement units correspond to about two meters.

### 3.2 Digital signage for product life cycle management

We experimented and evaluated mobile agent-based active media for appliances, e.g., electric lights. It is unique among other existing active media because it does not support advertising for its target appliance but also assist users to control and dispose the appliance. We attached an RFID tag to an electric light and provide a mobile agent as an active media for the light. The media is attached to its target item and is deployed at computers close to the current position of the item. The current implementation assumes that an agent for managing active media for its target appliance is created when the appliance is shipped from its factory.

Since the agent defines programs to display three kinds of active media content inside it, it selects them according to their spaces. It supports the lifecycle of the item from shipment, showcase, assembly, using, and disposal.

- **In warehouse:** While the light is in a warehouse, its agent is deployed at a computer in the warehouse. It notifies its specification, e.g., its product number, serial number, the date of its manufacture, the size and weight of it, to a server of warehouse.
- **In store:** While the light is at a showcase in a store, its agent is deployed at a computer close to its target object to display advertising media to sale it customers who visit the store. Figure 2 a) and b) are two images maintained in the agent and displays the price, product number, and manufacture name on its current computer.
- **In house:** When the light is bought and carried to the house of its owner, its agent migrates to a computer in the house and illustrate how to assemble it. Figure 2 c) is the active media for assembly guide. The agent also illustrate how to use it as shown in Figure 2 d). When it is disposed, the agent shows its active media for disposal guide. Figure 2 e) shows the image that illustrates how to dispose the appliance.



Figure 2. Digital signage for supporting appliance

In house-setting, we can define agents that control appliances, which may not have any network interfaces. In both of the approaches we describe here, the lights are controlled by switching their power sources on or off through a commercial protocol, called X10.

The first can autonomously turn room lights whenever a user with a tagged user is sufficiently close to them. The

agent attached to the light can also work as our X10-based server's client and is running on the stationary runtime system in the room. When a tagged user approaches a light, an LIS in the room detects the presence of his/her tag in the cell that contains the light. The LIS then moves the agent that is bound to his/her tag to the runtime system on which the light's agent is running. The user's agent then requests that the lights' agent turns the light on through the inter-agent communication.

The second allows us to use a PDA to remotely control nearby lights. In this system, place-bound controller agents, which can communicate with X10-base servers to switch lights on or off, are attached to places with room lights. Each user has a tagged PDA, which supports the runtime system with WindowsCE and a wireless LAN interface. When a user with a PDA visits the cell that contains a light, the framework moves a controller agent to the runtime system of the visiting PDA. The agent, now running on the PDA, displays a graphical user interface to control the light. When the user leaves that location, the agent automatically closes its user interface and returns to its home runtime system.

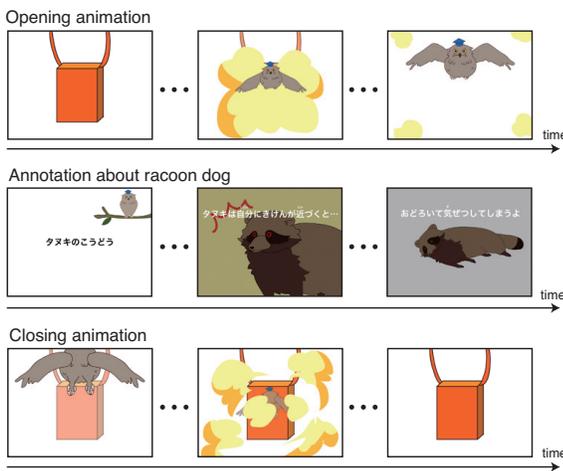
### 3.3 User-assistant agent for visitors in museums

The second is an experiment at the Museum of Nature and Human Activities in Hyogo, Japan, using the proposed system. The experiment was carried out at four spots in front of specimens of stuffed animals, i.e., a bear, deer, racoon dog, and wild boar. Each spot could provide five different pieces of animation-based annotative content about the animals, e.g., its ethology, footprints, feeding, habitats, and features, and had a display and Spider's active RFID reader with a coverage range that almost corresponded to the space, as shown in Fig. 3. When a visitor first participated in the experiment, an operator input the point of interest and the route for the new visitor and created his/her virtual agent.



Figure 3. Spot in museum.

The experiment was designed to enable visitors to imagine that their agents, which were virtual owls, were within their pendant. Each visitor has a colored pendant including RFID tags. A newly created agent is provided with its own color corresponding to the color of the pendant attached to the agent, because visitors could distinguish between their agents and others' agents through their pendants' colors. This may seem to be naive but it effectively enables visitors to readily identify their agents. For example, when a visitor entered a spot with the specimen of a racoon dog, his/her agent migrated from his/her pendant to the display located in the spot. As shown in Fig. 4, an agent tied to the pendant plays the opening animation to inform that its target is a visitor with a pendant. It next plays the annotation and then the closing animation.



**Figure 4. Opening animation, annotation animation, and closing animation for pendant**

We simultaneously provided two kinds of routes for visitors to evaluate the utility of our user-navigation support. Both routes navigated visitors to destination spots along the way (Fig. 3). They enabled all visitors to walk around an exhibition both consisting of four spots two or three times. That is, a visitor might visit the same spots two or three times depending on the navigation providing by his/her agent. In addition, the first route enabled visitors to explicitly select subjects they preferred by moving to one of the neighboring spots corresponding to the subjects selected in specified spots at specified times. The second route provided visitors with several quizzes to review what they had learnt about the animals by selecting neighboring spots corresponding to their answers in specified spots at specified times. Both the experiments offered visitors animation-based annotative content about the animal in front of them so they could learn about it while observing the correspond-

ing specimen.

## 4 Conclusion

We designed and implemented a context-aware infrastructure for building and managing mobile agent-based content displayed on public terminals or digital signage, where mobile agents are autonomous programs that can travel from computer to computer under their own control as virtual counterpart objects for people or physical entities. It provides users and physical entities with mobile agent-based content to support and annotate them. Using location-tracking systems, it can migrate content to stationary or mobile computers near the locations of the users and physical entities to which the agents are attached. That is, it allows a mobile user to access its personalized services in an active computing environment and provides user/location-dependent active media to a user's portable computer or stationary computer. It is managed on a decentralized manner. In addition, to support large-scale context-aware systems, the system is managed in a non-centralized manner. Using the system, we constructed and operated two applications of location/user-aware multimedia in a museum and stores as case studies in our development of ambient computing services in public spaces.

## References

- [1] B.L. Brumitt, B. Meyers, J. Krumm, A. Kern, S. Shafer: EasyLiving: Technologies for Intelligent Environments, Proceedings of International Symposium on Handheld and Ubiquitous Computing, pp.12-27, 2000.
- [2] A. Harter, A. Hopper, P. Steggeles, A. Ward, and P. Webster: The Anatomy of a Context-Aware Application, Proceedings of Conference on Mobile Computing and Networking (MOBICOM'99), pp. 59-68, ACM Press, August 1999.
- [3] I. Satoh: A Location Model for Pervasive Computing Environments, Proceedings of IEEE 3rd International Conference on Pervasive Computing and Communications (PerCom'05), pp.215-224, IEEE Computer Society, March 2005.
- [4] I. Satoh: Context-aware Agents to Guide Visitors in Museums, in Proceedings of 8th International Conference on Intelligent Virtual Agents (IVA'08), Lecture Notes in Artificial Intelligence (LNAI), vol.5208, pp.441-455, September 2008.
- [5] I. Satoh: A Context-aware Service Framework for Large-Scale Ambient Computing Environments, in Proceedings of ACM International Conference on Pervasive Services (ICPS'09), pp.199-208, ACM Press, July 2009.

# An End-user Domain-specific Model to Drive Dynamic User Agents Adaptations

Ingrid Nunes, Simone D.J. Barbosa, and Carlos J.P. de Lucena

Pontifical Catholic University of Rio de Janeiro (PUC-Rio) – Rio de Janeiro – Brazil

E-mail: {ionunes, simone, lucena}@inf.puc-rio.br

## Abstract

*Modeling automated user tasks based on agent-oriented approaches is a promising but challenging task. Personalized user agents have been investigated as a potential way of addressing this issue. Most of recent research work has focused on learning, eliciting and reasoning about user preferences and profiles. In this paper, our goal is to deal with the engineering of such systems, barely discussed in the literature. In this context, we present a high-level domain-specific model whose aim is to give users the power to customize and dynamically adapt their user agents. In addition, we propose a general architecture for developing user-customizable agent-based systems.*

## 1. Introduction

Many modern computer systems are providing assistance to several of our usual tasks, by incorporating features with a proactive and autonomous behavior. Typical examples include product recommendations based on our purchase history and generation of playlists based on songs we listen. These systems are increasingly becoming part of our everyday life. A generalized and ambitious idea underlying such systems is the personalized user agents [9], which are personal assistants acting on the users' behalf. Even though significant research effort has been invested on developing user agents, we are far from their massive adoption.

Schiaffino & Amandi [12] presented an empirical study that gives a solid basis for explaining this scenario. They claim the "human-computer interaction people have criticized agent-based methodologies that seem to produce systems not easily accepted by the user: one of the main reasons is the autonomy of the agents that can cause a loss of control by the user." Their study showed that different users need different kinds of user agents. In addition, a large group of users is willing to adopt user agents only if they know exactly what the agent is going to do.

Our research addresses this group of users. In this paper we demonstrate an approach to empower users with a high-level domain-specific language that allows them to

dynamically program and personalize their agents, as opposed to inference models that might reach the wrong conclusions about user preferences and cause agents to take inappropriate actions. Our approach distinguishes user *configurations* from *preferences*, which we collectively refer to as *customizations*. Configurations are direct and determinant interventions that users perform in a system, such as adding/removing services or enabling optional features. They can be related to environment restrictions, e.g. a device configuration. Preferences represent information about the users' values that influence their decision making, and thus can be used as resources in agent reasoning processes. They typically indicate how user rates certain options better than others in certain contexts. The present work evolved from our approach for building customized service-oriented user agents [11], which only dealt with user configurations and it did not address dynamic adaptations, i.e. our previous user agents did not evolve at runtime.

Our goal in this paper is twofold. We present a *Domain-specific Model (DSM) to model user preferences*, which provides the necessary vocabulary to build an end-user preferences language. Existing representation models of user preferences force users to express their preferences in a particular way. Consequently, these works create the need for elicitation techniques to interpret answers to questions and indirectly build the user model. The language that our DSM creates allows users to express different kinds of preference statements, creating a vocabulary that is very close to natural language. The proposed DSM is a metamodel that may be instantiated to build different applications.

We also show how this language is used in broader context, which is an *architecture to build user-customizable applications*, composed of user agents that are dynamically adapted based on a user model that follows our metamodel. We have taken into account software engineering issues identified as current practices to develop applications based on user models. In this sense, we also contribute with an analysis of existing mechanisms to implement user customizations, which may result in low-quality software architectures. Good (modular, stable, ...) architectures are essential to produce higher quality software which is easier

to maintain. Otherwise, software architectures may degenerate over time, making their maintenance a hard task, by increasing costs with refactorings.

This paper is organized as follows. In Section 2, we describe our user-driven software architecture. Section 3 presents our DSM, followed by Section 4, which evaluates our metamodel, showing its generality when used across different domains. Section 5 presents related work. Finally, Section 6 concludes this paper.

## 2. A User-driven Software Architecture

Our research on developing personalized user agents is driven by a reference architecture that allows to adapt agents based on an end-user’s DSM. In this section, we first present usual software engineering practices adopted to develop user-model based systems. They motivate the structure of our reference architecture, which is also detailed.

### 2.1. Software Engineering Practices to Develop User Agents

An essential characteristic of user agents is that they store information specific to each user. This is typically implemented either using: (i) a user model, which stores user information in a single location and is checked whenever a user-dependent action is performed; and (ii) control variables, which are inserted in the code to reflect user customizations and used to make some decisions that indicate to an agent the right course of actions it should take. Both solutions are essentially the same, with the difference that the first solution concentrates all the user-specific data. Even though these solutions produce the desired behavior, they have drawbacks from a software engineering perspective.

Concentrating all user customizations in a single component creates a high coupling between this component and other system components. In addition, changes in this unique component may imply a lot of little changes applied to a lot of different classes. This characterizes the Shotgun Surgery bad code smell [6]. Moreover, in both solutions, a control variable will be used – in (i), it is retrieved from the

user model – which is a program variable used to regulate the flow of control of the program. These control variables, i.e. user customizations, may be used in several system locations and are usually used in chained `if` or `switch` statements scattered throughout the system. If a new clause is added to the switch, all statements must be changed. This is another bad code smell, the Switch Statement [6], and the object-oriented notion of polymorphism gives you an elegant way to deal with this problem.

Another software engineering issue related to user agents is that user customizations may be seen as a *concern* in a system that is spread all over the code. However, at the same time, each customization is associated with different services (also concerns) provided to users. Therefore, when developing such system one has to choose the dimension in which the software architecture will be modularized: in terms of services (Figure 1(a)) or modularizing user settings in a single model (Figure 1(b)). It can be seen that it is not possible in either approach to modularize concerns in single modules. In addition, without modularizing user customizations, as in Figure 1(a), they are buried inside the code, thus making it difficult to understand them as a whole.

Based on these arguments, we claim that there is a need for better software architectures to build personalized user agents, taking into account good software engineering practices. However, dealing with variable traits that emerge from user customization points is not a trivial task. These customization points are spread all over the system architecture and play different roles in agent architectures [5, 10]. If all this information is contained in a single user model, we have the problems discussed above and this model would aggregate information related to different concerns of the system (low cohesion among user model elements).

### 2.2. Detailing our Software Architecture

Our solution to the previously described issues is to provide a *virtual separation of concerns* [7]. The main idea is to structure the user agent architecture in terms of services by modularizing its variability as much as possible into agent abstractions. We provide a virtual modularized view of user customizations, as Figure 1(c) illustrates. Customizations are not design abstractions, but they are implemented by typical agent abstractions (goals, plans, etc.), i.e. they play their specific roles in the agent architecture. The virtual user model is a complementary view that provides a global view of user customizations. This model uses a high-level end-user language, and users are able to configure their agents by means of this model. This section details our proposed architecture, depicted in Figure 2, and describes the mechanism that makes the virtual user model (henceforth referred to as user model) work with agent architectures.

The *User Agents* module consists of agents that provide different services for users, e.g. scheduling and trip plan-

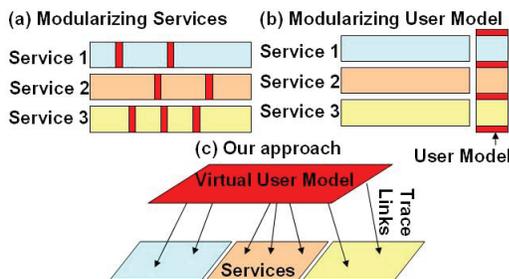


Figure 1. Modularization Approaches.

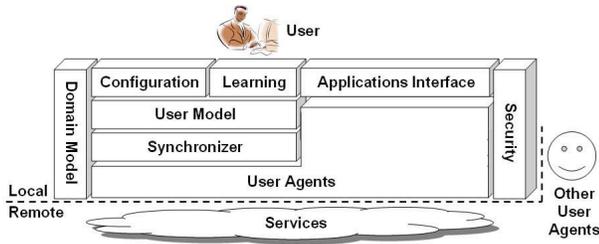


Figure 2. Proposed Architecture.

ning. Their architecture supports variability related to different users, as well as provide mechanisms to reason about preferences. These agents use services provided by a distributed environment (the *Services* cloud), and their knowledge is based on the *Domain Model*, composed of entities shared by user agents and services, application-specific, etc. The *Security* module addresses security and privacy issues, because user agents may share information with other user agents. This module aggregates policies that restrict this communication, assuring that confidential information is kept safety secured. Users access services provided by user agents through the *Applications Interface* module.

The *User Model* contains user configurations and preferences expressed in a high-level language. They are present in the user agents architecture but as design-level abstractions. Clearly, there is a connection from the *User Model* and *User Agents*. This connection is stored in the form of trace links, indicating how and where a customization is implemented in a user agent(s). Adaptations are performed at runtime and are accomplished based on the trace links between the *User Model* and the *User Agents* architecture. The *Synchronizer* is the module in charge of adapting *User Agents* based on changes in the *User Model*. It is able to understand these trace links, and knows which transformation must be performed in the *User Agents* based on changes in the *User Model*. Therefore, the *User Model* drives adaptations in the *User Agents*. By means of the *Configuration* module, users can directly manipulate the *User Model*, which gives them the power to control and dynamically modify user agents, using a high-level language. In addition, changes in the *User Model* may be performed or suggested by the *Learning* module, which monitors user actions to infer possible changes in the *User Model*. This module has a degree of autonomy parameter, so it may automatically change the *User Model*, or just suggest changes to it, to be approved by the end users.

### 3. A Metamodel for Building Application-specific User Models

In this section, we present and detail our proposed metamodel, whose aim is to allow to build application-

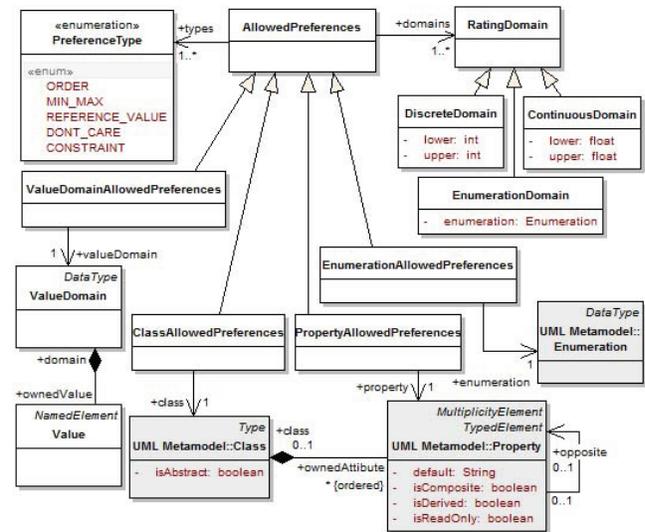


Figure 3. A Metamodel for Modeling User Preferences (Part I).

specific user models, using domain-specific abstractions. The metamodel provides concepts to represent user configurations and preferences. Our metamodel, which is an extension of the UML metamodel<sup>1</sup>, is depicted in Figures 3 and 4. Elements of the UML metamodel, e.g. *Class* and *Property*, are either distinguished with a gray color in diagrams or are referred in properties.

Before instantiating the metamodel to model user customizations at runtime, it is necessary to build the Domain Model (Section 2) at development time, for defining domain abstractions that are referred to in the User Model. The Domain Model consists of: (i) an Ontology model; (ii) a Variability model; and (iii) a Preferences Definition model. The Ontology model represents the set of concepts within the domain and the relationships between those concepts. The Variability model, in turn, allows modeling variable traits within the domain, which are later used for defining user configurations. The goal of the Variability model is to describe variation points and variants in the system, which can be either optional or alternative. In addition, restrictions may be defined in order to represent relationships between variations. The Variability model is used to define the configuration of the system in the User Model. This part of our metamodel was explored in our previous work [11] and is out of the scope of this paper. Therefore, we give this brief introduction to the Variability model, but we refer the reader to [11] for further details.

The part of our metamodel that is used in the Preferences Definition model is presented in Figure 3. The purpose of

<sup>1</sup><http://www.omg.org/spec/UML/>

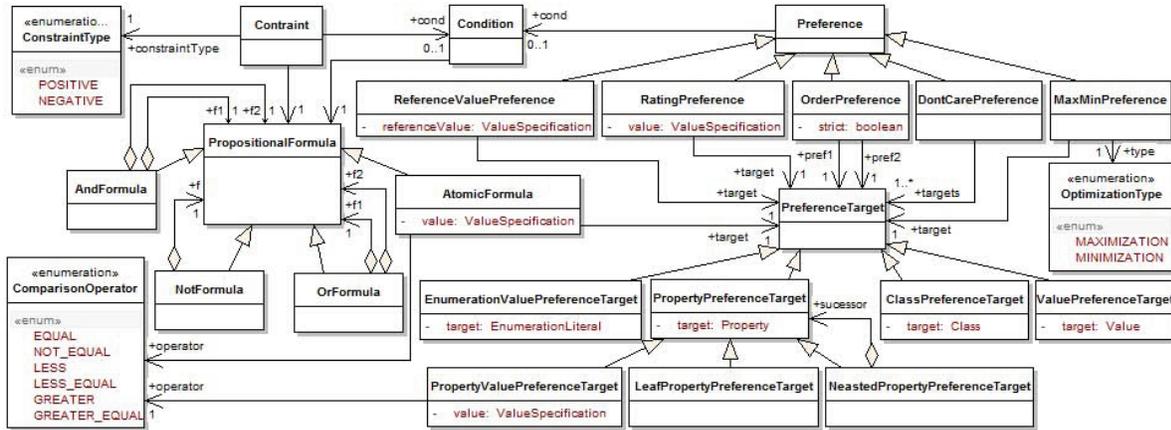


Figure 4. A Metamodel for Modeling User Preferences (Part II).

this model is to define *how* users can express their preferences and *about which elements* of the Domain Model. Even though it is desirable that users be able to express preferences in different ways, it is necessary to have agents that can deal with them. For instance, if application agents can deal only with quantitative preference statements, user preferences expressed in a qualitative way will have no effect on the system behavior.

Users can express different types of preference: (i) Order (ORDER) – expresses an order relation between two elements, allowing users to express “*I prefer trains to airplanes.*” A set of instances of the Order preference comprises a partial order; (ii) Reference Value (REFERENCE\_VALUE) – enables users to indicate one or more preferred values for an element. It can be interpreted as the user preference is a value on the order of the provided value; (iii) Minimize/Maximize (MIN\_MAX) – indicates that the user preference is to minimize or maximize a certain element; (iv) Don’t Care (DONT\_CARE) – allows indicating a set of elements the user does not care about, e.g. “*I don’t care if I travel with company A or B;*” (v) Rating – allows users rating an element. By defining a RatingDomain for an element, users can rate this element with a value that belongs to the specified domain. This domain can be numeric (either continuous or discrete), with specified upper and lower bounds. In addition, an enumeration can be specified, e.g. LOVE, LIKE, INDIFFERENT, DISLIKE and HATE. Moreover, different domains can be specified for the same element. Using Rating preferences, it is possible to assign utility values to elements, or to express preference statements; and (vi) Constraint (CONSTRAINT) – a particular preference type that establishes a hard constraint over decisions, as opposed to the other preference types, used to specify soft constraints. Constraints allows users to express strong statements, e.g. “*I don’t travel with company D.*”

Different kinds of preferences may be used by agents in

different ways, according to the approaches they are using to reason about preferences. If an agent uses utility functions and the user defines that the storage capacity of a computer must be maximized and provides a reference value  $\alpha$ , the agent may choose a utility function like  $f(x) = \sqrt{x}$ .

For defining the allowed preference types, developers must create instances of AllowedPreferences, and make the corresponding associations with types and domains. The specializations of AllowedPreferences characterize different element types that can be used in preference statements. There are four different possibilities: classes (*I prefer notebook to desktop*), properties (*The notebook weight is an essential characteristic for me*) and their values (*I don’t like notebooks whose color is pink*), enumeration literals (*I prefer red to blue*) and values (*Cost is more relevant than quality*). Value is a first-class abstraction that we use to model high-level user preferences. We adopted this term from [3]. A scenario that illustrates the use of values is in the travel domain. A user may have comfort (a value) as a preference when choosing a transportation, instead of specifying fine-grained preferences, such as *trains are preferred to airplanes, but traveling in an airplane first-class is better than by train*, and so on. In this case, the user agent is a domain expert that knows what comfort means.

Based on these definitions and on our metamodel (Figure 4), it is possible to build a User Model to model preferences and configurations. It is composed of two parts: (i) Configuration model; and (ii) Preferences model. As discussed above, in the Configuration model, users choose optional and alternative variation points from the Variability model, defining their configurations [11]. On the other hand, in the Preferences model, users define preferences and constraints. These are more closely related to a cognitive model of the user. User preferences (or soft constraints) determine what the user prefers, and indirectly how the system *should* behave. If the preferred behavior is not possible, the system

may move to other acceptable alternatives. Constraints, in turn, are restrictions (hard constraints) over elements. As opposed to preferences, they directly define mandatory or forbidden choices that *must* be respected by the system.

Figure 4 shows the `Constraint` element and five different specializations of `Preference` that represent the different preference types previously introduced. Constraints are expressed in propositional logic formulae, however using only  $\neg$ ,  $\wedge$  and  $\vee$  logical operators. Atomic formulae refer to the same types of elements of preferences and can use comparison operators ( $=$ ,  $\neq$ ,  $>$ ,  $\geq$ ,  $<$ ,  $\leq$ ) between properties and their values. The `PreferenceTarget` and its subtypes are used to specify the element that is the target of the preference statement or formula. In addition, it allows to specify nested properties, such us `Flight.arrivalAirport.location.country`.

If we have directly associated preferences to classes, properties, enumerations and values, either we would have to make specializations of each preference type to each element type or to change the UML metamodel to make a common superclass of classes, properties, enumerations and values. Given that we did not want to modify the UML metamodel, but only to extend it, and the first solution would generate four specializations for each preference type, we used the `PreferenceTarget` as an indirection for elements that are referred in preferences and constraints.

Besides defining preferences and constraints, users can specify conditions, also expressed in propositional logic formulae, to define contexts in which preferences and constraints hold. Furthermore, in order to guarantee that users produce valid instances of the metamodel, we have defined additional constraints over instantiated models, e.g. in a nested property, the child of a property whose class is X must also be a property of Class X.

#### 4. Evaluating our User Metamodel across Different Application Domains

Our metamodel was built using preference statements collected from different individuals and from papers related to user preferences. The idea was to contemplate the different kinds of preference statements in order to maximize the users' expressiveness. The metamodel uses abstractions from the user preferences domain, therefore the language is built as an end-user language. This section presents two Preferences models to show that our metamodel is generic enough to model different kinds of preferences statements in different domains – flight and computer domains. Given that these are two well-known domains, we assume that the reader is familiar with them, and due to space restrictions, we present only the Preferences models. In addition, we assume that the Preferences Definition model defines that all preference types over all elements are allowed.

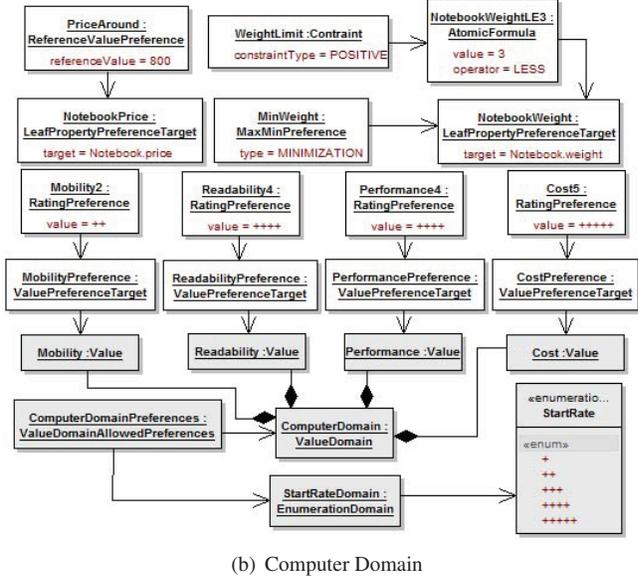
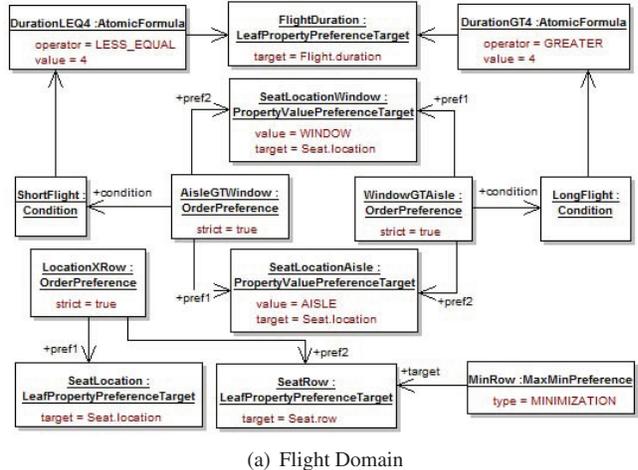


Figure 5. User Preferences model.

The first Preference model, which is from the flight domain, indicates where a user prefers to seat inside an airplane. This model consists of three order preferences, two of them with conditions, and one minimization preference. Next, we present the four modeled preference statements in natural language, and Figure 5(a) shows how they are modeled with our metamodel abstractions.

- P1.** *If the flight is short, i.e. its duration does not exceed 4 hours, I prefer a seat by the aisle to a seat by the window.*
- P2.** *If the flight is long, i.e. its duration is higher then 4 hours, I prefer a seat by the window to a seat by the aisle.*
- P3.** *I always prefer to sit at the first rows of the airplane.*
- P4.** *Sitting in the first rows of the airplane is more important to me than the seat location.*

The computer domain Preferences model presented in Figure 5(b) has some elements in gray color. They are

not part of the Preferences model, but from the Domain model, but we included them in Figure 5(b) to present some application-specific concepts used in this model. First, four values are defined in the Computer Domain (mobility, readability, performance and cost). These values can be rated with “+”, ranging from one to five. These are the natural language preference statements modeled in Figure 5(b):

- P1. *Cost is the most important value (+++++).*
- P2. *I rate performance with ++++.*
- P3. *I rate readability with ++++.*
- P4. *I rate mobility with ++.*
- P5. *I'm expecting to pay around \$800 for my laptop.*
- P6. *I want a computer with less than 3Kg.*
- P7. *The lighter the computer is, the better.*

It is important to notice that Rating and Order preferences provide different information. By saying that cost is +++++ and performance is ++++, a user is informing that cost is more important than performance (order), but performance is also important, and should be taken into account.

## 5. Related Work

Several approaches have been proposed to deal with user preferences. To build our metamodel, we have made extensive research on which kinds of preferences other proposals represent and additional concepts they define. Typically, preferences are classified as quantitative or qualitative (e.g. “I love summer” versus “I like winter more than summer”). Both approaches can be represented through our metamodel. Quantitative preferences are modeled in the framework proposed in [1], by means of a preference function that maps records to a score from 0 to 1. On the other hand, CP-Nets [4] models qualitative preferences. CP-Nets also allow modeling conditionality, which is considered in our work as well. The concept of normality is defined in [8], so that users can express preferences considering normal states of the world, but these preferences may change when the world changes. The normality abstraction can be modeled using conditions in our metamodel.

Ayres & Furtado proposed the OWLPref [2], a declarative and domain-independent preference representation in OWL. OWLPref does not precisely define the preferences model, e.g. lacking the definition of associations, it shows only a hierarchical structure of preferences. A preference metamodel is also proposed in [13]. However, its expressiveness is very limited. It only allows to define desired values (or intervals) of object properties.

## 6. Conclusion

In this paper, we proposed a domain-specific metamodel that provides abstractions from the user domain, including constraints and preferences. Different abstractions used by

end users in natural language statements are directly represented. Our metamodel provides a domain-specific language that empowers users to express their preferences to program their agents. Besides constraints, five different preferences types can be represented. In addition, we adopt values as a first-class abstraction to model high-level preferences. Instances of our metamodel are to be used in combination with our proposed architecture, which uses them as a virtual user representation. Services are provided by user agents structured with traditional agent-based architectures. The User Model provides a modularized view of different user-related concepts spread into agent architectures. A Synchronizer module ensures that changes in the User Model demands appropriate adaptations in user agents.

We are currently working on a language based on our metamodel using syntactic sugar. In addition, we are investigating how to verify the User Model to identify inconsistencies across preferences.

## References

- [1] R. Agrawal and E. L. Wimmers. A framework for expressing and combining preferences. In *2000 ACM SIGMOD*, pages 297–306, 2000.
- [2] L. Ayres and V. Furtado. Owlpref: Uma representação declarativa de preferências para web semântica. In *XXVII Congresso da SBC*, pages 1411–1419, Brazil, 2007.
- [3] T. Bench-Capon. Persuasion in practical argument using value-based argumentation frameworks. *Journal of Logic and Computation*, 13(3):429–448, 2003.
- [4] C. Boutilier, R. I. Brafman, H. H. Hoos, and D. Poole. Cp-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research*, 21:135–191, 2004.
- [5] J. Doyle. Prospects for preferences. *Computational Intelligence*, 20:111–136, 2004.
- [6] M. Fowler, K. Beck, J. Brant, W. Opdyke, and D. Roberts. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Professional, 1 edition, July 1999.
- [7] C. Kästner and S. Apel. Virtual separation of concerns - a second chance for preprocessors. *Journal of Object Technology*, 8(6):59–78, 2009.
- [8] J. Lang and L. van der Torre. From belief change to preference change. In *ECAI 2008*, pages 351–355, The Netherlands, 2008. IOS Press.
- [9] P. Maes. Agents that reduce work and information overload. *Commun. ACM*, 37(7):30–40, 1994.
- [10] I. Nunes, S. Barbosa, and C. Lucena. Modeling user preferences into agent architectures: a survey. Technical Report 25/09, PUC-Rio, Brazil, September 2009.
- [11] I. Nunes, C. J. Lucena, D. Cowan, and P. Alencar. Building service-oriented user agents using a software product line approach. In *ICSR '09*, pages 236–245, 2009.
- [12] S. Schiaffino and A. Amandi. User - interface agent interaction: personalization issues. *Int. J. Hum.-Comput. Stud.*, 60(1):129–148, 2004.
- [13] D. Tapucu, O. Can, O. Bursa, and M. O. Unalir. Metamodeling approach to preference management in the semantic web. In *M-PREF 2008*, pages 116–123, USA, 2008.

# Towards Generation of Domain Ontology from LMF Standardized Dictionaries

Feten Baccar Ben Amar, Bilel Gargouri

MIRACL Laboratory  
FSEGS, B.P. 1088, 3018  
Sfax, Tunisia

[feten.baccar@mes.rnu.tn](mailto:feten.baccar@mes.rnu.tn) [bilel.gargouri@fsegs.rnu.tn](mailto:bilel.gargouri@fsegs.rnu.tn)

Abdelmajid Ben Hamadou

MIRACL Laboratory  
ISIMS, B.P. 242, 3021 Sakiyet-Ezzit  
Sfax, Tunisia

[abdelmajid.benhamadou@isimsf.rnu.tn](mailto:abdelmajid.benhamadou@isimsf.rnu.tn)

**Abstract**—The present paper proposes an approach of domain ontology generation from LMF standardized dictionaries (ISO-24613). The originality of this work lies in the use of a unique and finely-structured source containing multi-domain lexical knowledge of morphological, syntactic and semantic levels, lending itself to ontological interpretations. Our method for the domain ontology generation consists, firstly, in building the core of the target ontology, taking advantage of the LMF standardized dictionary structure. Secondly, it consists in enriching such core starting from textual sources with guided semantic fields available in the definitions and the examples of lexical entries. The reliability of the proposed approach is shown through the illustrations carried out on the Arabic language. This choice is explained not only, by the great deficiency of work on Arabic ontology building, but also by the availability within our research team of an LMF standardized Arabic dictionary.

**Keywords**—LMF standardized dictionary; generation of domain ontology; Arabic language

## I. INTRODUCTION

In recent years, ontologies and formal representations of knowledge have received much interest, opening fascinating possibilities in several domains. Indeed, ontologies are extremely powerful knowledge representation tools for modeling and managing various applications ranging over Natural Language Processing (NLP), information retrieval, semantic Web, etc. In this context, research on ontology building has become increasingly widespread in computer science community.

The first ontology building was an entirely manual and time consuming task requiring an enormous effort for knowledge domain acquisition and especially raising maintenance problems. Aiming at overcoming these drawbacks, many methodologies have been devised in order to perform an automatic ontology development. Yet, ontologies remain complex and difficult objects to be built regardless of the grounded support: texts [1], dictionaries [9], thesauri [10], etc.

Although most of the ontology building methodologies focus on the conceptual information rather than the lexical information, the latter has been progressively sought in the ontology-based applications. Actually, providing ontologies with linguistic data constitutes, at present, a crucial topic of research. For instance, LingInfo [5], LexOnto [7], Linguistic

Information Repository (LIR) [14], Linguistic Watermark Suite (LWS) [15] and LexInfo [6] have been designed and developed over the past years. Most of these models build on the standard of the lexical resources representation recently defined by the LMF (Lexical Markup Framework) standard [12]. In fact, LMF meta-model provides a common and shared representation of lexical objects that allows for the encoding of rich linguistic information, including morphological, syntactic, and semantic aspects [8]. However, the association of lexical information with the ontological entities is a heavy and time consuming activity considering the plurality and the heterogeneity of the resort sources [13].

An LMF standardized dictionary incorporates diversified knowledge lending itself to ontological interpretations. Moreover, the huge lexical information proves to be very useful to generate domain ontologies without shifting to other resources. Indeed, finely-structured and multi-domain knowledge in an LMF standardized dictionary makes possible the constitution of the ontology core. Furthermore the big quantity of texts within guided semantic fields available in the definitions, explanations and examples are very interesting to accomplish the core enrichment and above all to provide the ontology elements with linguistic grounding or structure.

The main interest of this paper is to propose an approach for the automatic generation of domain ontologies from LMF standardized dictionaries. It is through the illustrations carried out on the Arabic language that our approach proved to be reliable. Two main reasons are behind this choice. The first one is that work on ontology construction for Arabic are very deficient and the second is that an Arabic dictionary conform to the LMF standard is at hand within our research team [3].

The present paper is outlined as follows. The next section is devoted to a review of the state-of-the-art in domain ontology building as well as to the models for associating lexical information with the ontological elements. Section 3 exposes the basics of our approach for domain ontology generation from LMF standardized dictionaries and gives details on the corresponding stages. Section 4 presents an illustration of our approach, realized with an LMF standardized Arabic dictionary.

## II. RELATED WORK

Since conceptualization is the most difficult phase in the ontology building process, many proposals have been suggested to facilitate ontological engineering and knowledge domain acquisition. Indeed, some efforts, based on MRDs (Machines Readable Dictionaries), have been put forth to extract conceptual entities. The MRDs are electronic versions of dictionaries which provide rich linguistic knowledge and constitute a structural support likely to obtain conceptual entities [11]. Therefore, the knowledge acquired from the MRDs has been useful for generating hierarchies of concepts, thesauri and ontologies [9]. Nevertheless, their fairly structured organization intended only for an editorial use, does not promote their direct use.

Moreover, the recourse to electronic dictionaries is not only restricted to the ontology building issue but also useful for the extension of the already created ontologies. In fact, lexical information is nowadays required by many applications such as ontology-based Information Retrieval and Extraction, Question/Answer and Semantic Web. But because the relied sources are lacking linguistic information, the quasi-totality of the domain ontology building methodologies tends to overlook it and concentrate on the conceptual information. It is also because the ontology representation formalisms, as RDF<sup>1</sup> and OWL<sup>2</sup> recommended by the W3C consortium, offer merely the most basic primitives for ontology modeling and they are not sufficient to allow for a richer representation.

After the introduction of LMF standard [12], representing a newly emerging standard for the creation and use of computational lexicons [8], a number of models have been devised in order to attach lexical information to ontology elements. Among the proposed models, we quote LIR developed within the framework of the NeOn<sup>3</sup> project and LexInfo which emerged as an alignment of the LingInfo and LexOnto models [6]. While LexInfo focuses on relations between the concepts of the ontology, LIR concentrates mainly on ontology classes and includes an elaborate machinery to represent spelling variants, short forms, abbreviations, transliterations, acronyms, etc [14]. Both models suggest frameworks aiming at the manual creation of a lexico-ontological resource starting from the domain ontology. This resource would then be enriched using relevant domain texts or external lexical resources (e.g. WordNet or Wikipedia) to seek further information (e.g. definitions, translations). However, as their implementations are presently manual and involving partial software assistance, these endeavours whose requested models (ontology model, lexical model and pivot model) are various, turn out to be complex and inconvenient [13].

<sup>1</sup> <http://www.w3.org/RDF/>

<sup>2</sup> <http://www.w3.org/2004/OWL/>

<sup>3</sup> <http://www.neon-project.org/web-content/>

## III. THE PROPOSED APPROACH

### A. The Domain Ontology Generation Approach: Fundamentals

Domain ontology offers a formal representation of concepts and relationships between the concepts for a given domain. Regarding the concept, it is a cognitive unit of meaning which may be an abstract idea or a mental symbol. As for the relations existing in ontology, they are conceptual relationships as taxonomy (i.e. relation of hierarchical organization) or semantic relationships as hyperonymy/hyponymy, holonymy/meronymy, kind-of and synonymy/antonymy. Besides the concepts and relations, ontology can include instances (or individuals) of concepts that form their extensions.

An LMF standardized dictionary could well be considered as a very suitable knowledge resource to generate domain ontologies. In this work, we prove that it contains all the elements required in ontology structures. Indeed, lexical information includes many tokens on the concepts, the relationships between the concepts and even the instances. Based on a subtle and powerful LMF meta-model, these tokens are expressed either directly, through the fields related to the lexical entries or indirectly, through the huge number of the texts in semantic knowledge. Figure 1 shows the matching between the LMF standardized dictionary elements and the domain ontology entities.

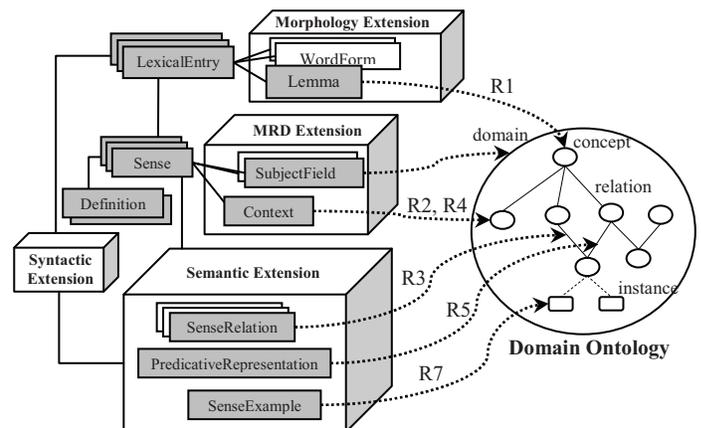


Figure 1. The matching between the LMF standardized dictionary elements and the domain ontology entities, (R1 → R7 are detailed in the next paragraph).

In order to identify the concepts of a particular domain, we consider the domain information given in the dictionary. Since a concept corresponds to a meaning of a word, we can directly deduce the concepts of the domain ontology from the instances of the *SubjectField* class attached to the class *Sense* instances.

With regard to the concepts properties, the LMF model defines many types of semantic relationships (e.g. synonymy, antonymy, etc.) between the senses of two or several lexical entries by means of the *SenseRelation* class. Consequently, a relation that connects two or several senses in the dictionary gives birth to an ontological relation linking the corresponding concepts. Moreover, according to the LMF model, it is possible

to describe the predicative structures as well as their semantic arguments, presenting links with the syntactic behaviour (s) of the word. These predicates correspond to semantic relations in the ontology. So, the generated ontology will contain additional semantic relations to those mentioned above (for more details, see the next section, R4).

The instances of concepts are directly deduced from the instances of the *SenseExample* class and those of the *LexicalEntry* class having the grammatical category « /properNoun/ ». Besides, much information obtained from LMF standardized dictionary proves to be relevant to explicit conceptual entities. Being more and more recommended in the domain ontologies, the lexical information (e.g. grammatical category, genre, number, etc.) would be obtained from the finely-structured fields of the LMF standardized dictionary attached to the lexical entries.

### B. The Domain Ontology Generation Method

The proposed approach for generating domain ontologies consists mainly of three stages (see Figure 2). Firstly, we intend to extract a fragment corresponding to the chosen domain from the whole dictionary. Secondly, we aim at digging out the basic elements of a given domain from the previously-created dictionary fragment in order to build a core of the target ontology. Finally, this core will be greatly enriched by exploiting the semantics conveyed by the textual fields of the dictionary. In the following subsections, we will detail the three stages of our approach.

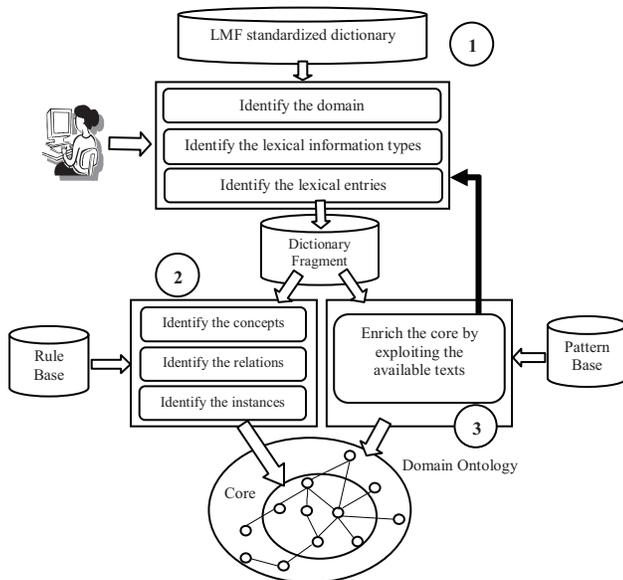


Figure 2. The approach overview for domain ontology generation.

1) *Extract a dictionary fragment*: In this initial stage, we intend to virtually extract a dictionary fragment according to the given domain via simple queries. Such a fragment, created for reasons of optimization, has twofold important role. First, it constitutes the privileged initial source for generating the target ontology. Second, when handling the generated

ontology, conceptual nodes always keep reference to lexical information contained in the dictionary fragment whose extraction is based on the following elementary sub-stages.

a) *Identify the domain*: In order to collect relevant terms of the target ontology, the designer has to specify the domain available in the dictionary, such as sports, agriculture, science, medicine.

b) *Identify the lexical information types*: The designer has also to specify the types of lexical information (e.g. grammatical category, genre, number, inflected form, etc), which are required in the ontology. Such information is relative to the terms associated with the target ontology elements and its acquisition is directly done from the structured fields of the standardized dictionary. For instance, the inflected forms are obtained from the instances *WordForm* class linked to the *LexicalEntry* instance. As for the grammatical category, it is extracted from the « /partOfSpeech/ » attribute attached to the *LexicalEntry* class.

c) *Identify the lexical entries*: The identification of the lexical entries of this fragment is based on the available information in the *SubjectField* classes. To these lexical entries are added the semantically linked ones by means of the *SenseRelation* class of the dictionary, which connects two or several lexical entries.

2) *Build the ontology core*: The second stage consists in building the ontology core. The latter is built by identifying the concepts, relationships between concepts and individuals, which are directly recognized from the dictionary fragment by means of a set of rules.

a) *Identify the concepts*: As mentioned above, a concept corresponds to an instance of the *Sense* class. The name of the concept is defined using the following rules:

**(R1)** If the lexical entry has only one sense, then the concept label is the same as that of the lemma (the written form contained in the instance of the *Lemma* class).

**(R2)** If the lexical entry has many senses, then we shall have to deal with several concepts whose labels correspond to the instances of the *Context* and *Lemma* classes.

b) *Identify the relations*: An ontological relation can be semantic or conceptual. Their corresponding rules are presented respectively:

**(R3)** If the lexical entry has a sense related to that of another lexical entry through the *SenseRelation* class, then a semantic relation of the « synonym » type is created and links the pertaining ontology concepts of each sense.

**(R4)** If the lexical entry has one or many instances of *Context* class corresponding to the ontology concepts, then a taxonomic relationship (i.e. « is-a ») connects these sub-concepts to the generic concept already deduced from the *Lemma* class.

It should be noted that new types of semantic relations will be directly specified from the predicative structures, <predicate, subject, object>, with respect to the syntactic aspects given in the dictionary:

**(R5)** If the sense of a lexical entry has an instance of the *PredicativeRepresentation* class, then the semantic relation,  $\langle predicate, domain, range \rangle$ , is created in order to link the concepts involved in the ontology.

c) *Identify the Instances:* The deduction rules of the instance identification are as follows:

**(R6)** If the «partOfSpeech» attribute attached to the *LexicalEntry* class contains the value «/properNoun/», then an instance of concept is created with the same label as that of the lexical entry.

**(R7)** If the sense of a lexical entry is linked to an instance of the *SenseExample* class, then an instance of concept is directly deduced from this class.

3) *Enrich the core by exploiting the available texts:* The objective of this final stage is to enrich the ontological core which has already been built by new elements. To this end, we intend to exploit the associated texts with the various lexical entries identified in the dictionary fragment. These texts are available in the definitions, examples and citations. The idea of getting benefits of texts in natural language to build ontologies has already aroused the interest of several researchers. In this context, most of the contributions for automatic ontology construction from text employ various NLP tools and data mining techniques to carry out the analysis of text by adopting statistical and/or linguistic approaches ([1], [4]).

In the present work, we consider particular texts with guided semantic fields containing tokens of new concepts or new relationships between concepts which were not identified in the previous stages. This can be explained, inter alia, by the fact that certain entries of the dictionary have senses holding no tokens of their usage domains (i.e. SubjectField).

Owing to the regularity of the texts within reach, we apply syntactic pattern matching to extract relationships in which the core entities and the selected lexical entries participate using lexico-syntactic pattern. The latter describes a regular expression, composed of words, grammatical or semantic categories, and symbols for identifying the textual elements corresponding to this pattern [2]. The identification of such patterns requires text pre-processing, by applying various NLP tools (e.g. tokenizer, stemmer, syntactic analyzer, etc.). Therefore, a relation is recognized when a sequence of words in the text matches a pattern. In the following section we provide some examples of the lexico-syntactic pattern relative to the Arabic language.

#### IV. ILLUSTRATING THE APPROACH

All along the present study, many experiments have been carried out on the Arabic language to assess the applicability and the feasibility of the proposed method for domain ontology generation. Among these experiments, we report in this section an illustrative case pertaining to sports domain. In the following subsections, we will, first, present the LMF standardized Arabic dictionary. Then, we will show the results

of applying the three stages of our approach to the chosen example.

#### A. General Features of LMF Standardized Arabic Dictionary

The LMF standardized Arabic dictionary was developed within the framework of the project of the electronic interactive Arabic dictionary<sup>4</sup> managed by the ALECSO<sup>5</sup> and KACTS<sup>6</sup>. Our research team has developed the standard model of this dictionary [3] and worked out an experimental version<sup>7</sup> for research, currently composed of 38423 lexical entries and 28786 semantic relationships. In addition, thanks to LMF meta-model, our dictionary would certainly be an extendable resource that could be incremented with entries and lexical properties, extracted from other sources (e.g. Arabic lexicons, text corpora, etc).

#### B. The Illustrative Case: Sports Domain Ontology

The proposed approach is illustrated through the case of sports domain ontology, especially that relative to the jump «قفز» movement. The application of the three stages for the sports ontology generation, as well as the resulting ontology are presented in the paragraphs that follow.

1) *Extract a dictionary fragment of the sports domain:* This fragment is a restriction of the whole dictionary, gathering the lexical entries having senses related to sports «رياضة» domain, their synonyms, and the corresponding lexical information of the inflection type. The resulting fragment comprises the following lexical entries: *horse* «حصان», *gymnastics* «جمباز», *quintuple* «خماسي» and *jump* «قفز». Figures 3-a, 3-b, 3-c and 3-d show the portions of these lexical entries.

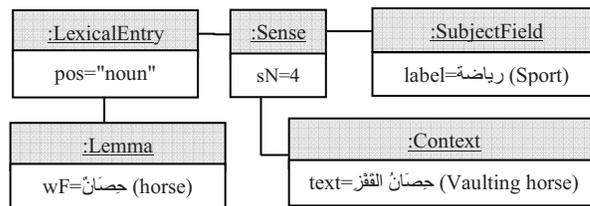


Figure 3-a. Portion of the lexical entry *horse* «حصان».

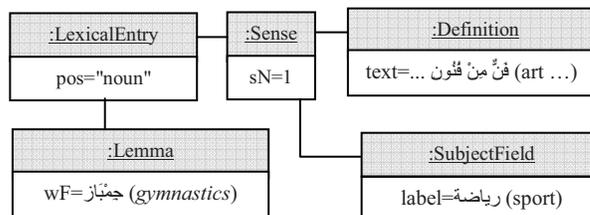


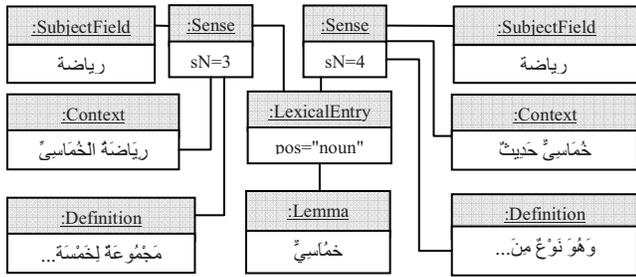
Figure 3-b. Portion of the lexical entry *gymnastics* «جمباز».

<sup>4</sup> [www.almuajam.org](http://www.almuajam.org)

<sup>5</sup> The Arab League Educational, Cultural and Scientific Organization.

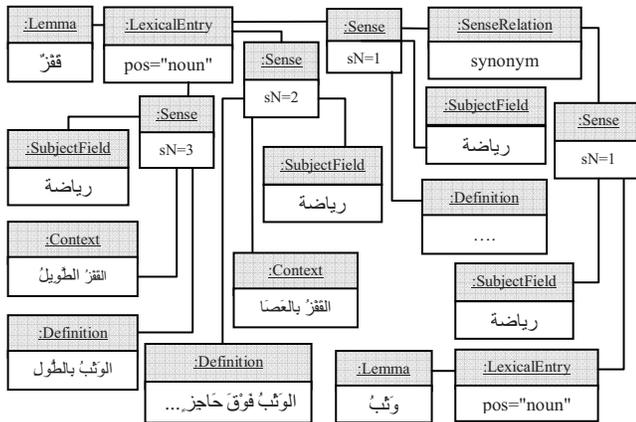
<sup>6</sup> King Abdul Aziz City for Science and Technology, Saudi Arabia.

<sup>7</sup> [www.vocwords.com](http://www.vocwords.com)



**Translations :**  
 خُمَاسِي : quintuple  
 رِيَاضَةُ الخُمَاسِي : pentathlon sports  
 خُمَاسِي حَدِيثٌ : modern pentathlon  
 مَجْمُوعَةٌ لِخَمْسَةِ... : a set of five ...  
 وَهُوَ نَوْعٌ مِنْ... : a kind of ...

Figure 3-c. Portion of the lexical entry *quintuple* « خُمَاسِي ».



**Translations :**  
 قَفْرٌ : jump  
 القفزُ بالعصا : pole vaulting  
 الوثبُ بالطول : long jump  
 الوثبُ فوقَ حاجزٍ : high jump  
 القفزُ الطويلُ : long jump  
 وثبٌ : vault

Figure 3-d. Portion of the lexical entry *jump* « قَفْرٌ ».

2) *Build the sports ontology core:* The ontology core is built from the previously extracted dictionary fragment:

a) *Identify the concepts:* During this sub-stage, we directly deduce the following concepts: sports « رياضة », horse « حصان », vaulting horse « حصان القفز », gymnastics « جمباز », jump « قفز », long jump « القفز الطويل », pole vaulting « القفز بالعصا », vault « وثب », high jump « الوثب فوق حاجز », long jump « الوثب بالطول », quintuple « خُمَاسِي », modern pentathlon « خُمَاسِي حَدِيثٌ », pentathlon sports « رِيَاضَةُ الخُمَاسِي ».

b) *Identify the Relations:* We deduce synonymy relations between the concepts vault « وثبٌ » and jump « قَفْرٌ », as well as the taxonomic relations between horse « حصانٌ » and vaulting horse « حصانُ القفز »; modern pentathlon « خُمَاسِي حَدِيثٌ »,

pentathlon sports « رِيَاضَةُ الخُمَاسِي » and quintuple « خُمَاسِي »; high jump « الوثبُ فوقَ حاجزٍ », long jump « الوثبُ بالطول », and vault « وثبٌ » and long jump « القفزُ الطويلُ », pole vaulting « القفزُ بالعصا » and jump « قفزٌ ».

Figures 4-a, 4-b, 4-c and 4-d show the ontology core elements which are directly deduced from each lexical entry.



Figure 4-a. Concept related to the lexical entry *gymnastics* « جَمبَازٌ ».



Figure 4-b. Core portion related to the lexical entry *horse* « حصانٌ ».

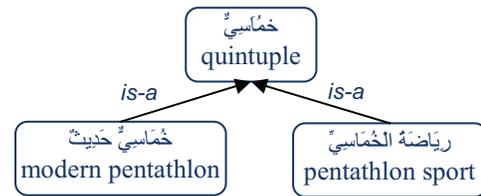


Figure 4-c. Core portion related to the lexical entry *quintuple* « خُمَاسِي ».

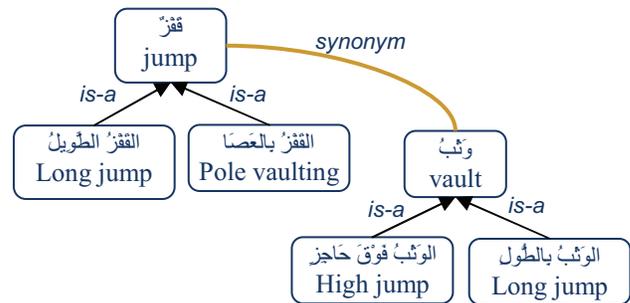


Figure 4-d. Core portion related to the lexical entry *jump* « قَفْرٌ ».

c) *Enrich the sports ontology core:* Figure 5 illustrates the results of ontology core enrichment. The extraction of the « kind-of » relation type, for example, is based on the following pattern,

< (concept1) + نوع من « kind-of » + (concept2) >.

Moreover, the relation of holonomy/meronymy is deduced along with many patterns such as:

< (concept1) + مَجْمُوعَةٌ لـ « member-holonym » + (concept2, concept3, conceptn) >.

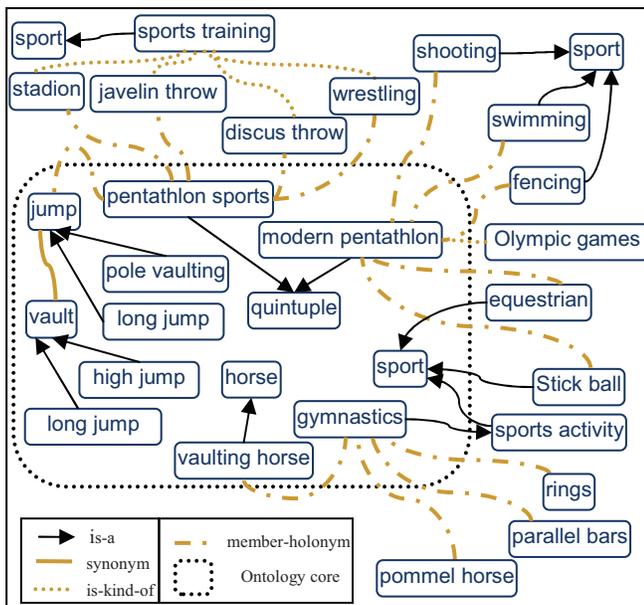


Figure 5. The generated sports domain ontology relative to the jump « قفّر » movement. The entities of the ontology core are gathered in the dotted line frame. The ontological entities are shown in English.

As it can be seen in Figure 5, the concepts and the relationships between concepts are shown to be pertinent. Therefore, the results obtained from the several experiments, undertaking the example of sports domain could well demonstrate the reliability of our method for domain ontology generation.

## V. CONCLUSION AND FUTURE WORKS

In this paper, we have proposed a new approach of domain ontology generation from LMF standardized dictionaries (ISO-24613). The originality of this approach lies in the use of a unique, finely-structured source, rich in lexical and conceptual knowledge. Our method consists of three stages. It starts with extracting a dictionary fragment, then proceeds by building an ontology core and finally ends with enriching the constructed ontology core. Moreover, the proposed approach is proven to be reliable through the illustrations carried out on the Arabic language whose choice is explained by two main motives. The first one is the great deficiency of work on Arabic building ontologies and the second is the availability within our research team of an LMF standardized Arabic dictionary as well as NLP tools.

Besides the qualitative evaluation based on the number of relationship types which can be easily detected from LMF meta-model, a first quantitative evaluation has shown that the core ontology generation can significantly reduce the high cost of building domain ontologies. This quantitative analysis has been performed by human experts (i.e. domain specialists) who carry out the manual census of ontological entities generated from a significant LMF dictionary fragment. Actually, they estimate that the core ontology representing about 30% of the total ontology elements could be

systematically generated from dictionary entities. Further, the complete evaluation will be accomplished on the whole dictionary by comparing the automatically generated domain ontologies with the available ones (e.g. Arabic WordNet).

Currently, we are setting to formalize all the deduction rules of the ontological information from the LMF standardized dictionary. In fact, we focus on the linguistic patterns, in particular, those of the Arabic language corresponding to the hierarchical relation «is-a». As future perspectives to our work, we aim at the automation of the core enrichment stage.

## REFERENCES

- [1] N. Aussenac-Gilles, S. Despres and S. Szulman, "The TERMINAE Method and Platform for Ontology Engineering from Texts". Bridging the Gap between Text and Knowledge. P. Buitelaar, P. Cimiano (Eds.), IOS Press, 2008, pp. 199–223.
- [2] A. Auger and C. Barriere, "Pattern based approaches to semantic relation extraction: a state-of-the-art". Terminology, John Benjamins, 2008, 14–1, 1–19.
- [3] F. Baccar, A. Khemakhem, B. Gargouri, K. Haddar and A. Ben Hamadou, "LMF standardized model for the editorial electronic dictionaries of Arabic". In the 10<sup>th</sup> International Conference on Enterprise Information Systems ICEIS-2008, June 12-13, Barcelona, Spain, 2008, pp. 64-73.
- [4] P. Buitelaar, P. Cimiano and B. Magnini, "Ontology Learning From Text: Methods, Evaluation and Applications", IOS Press, 2005.
- [5] P. Buitelaar, T. Declerck, A. Frank, S. Racioppa, M. Kiesel, M. Sintek, R. Engel, M. Romanelli, D. Sonntag, B. Loos, V. Micelli, R. Porzel and P. Cimiano, "LingInfo: Design and applications of a model for the integration of linguistic information in ontologies". In Proceedings of the OntoLex'06 Workshop, Genoa, Italy, 2006, pp. 28-34.
- [6] P. Buitelaar, P. Cimiano, P. Haase and M. Sintek, "Towards Linguistically Grounded Ontologies". In the 6<sup>th</sup> Conference (ESWC2009), Heraklion, Greece, 2009, pp. 111-125.
- [7] P. Cimiano, P. Haase, M. Herold, M. Mantel and P. Buitelaar, "LexOnto: A model for ontology lexicons for ontology based NLP". In proceedings of the OntoLex'07, workshop, South Korea, 2007.
- [8] G. Francopoulo and M. George, "Language Resource Management - Lexical Markup Framework (LMF)". Technical report, ISO/TC 37/SC 4 N453 (N330 Rev.16), 2008.
- [9] H. gonçalo Oliveira, P. Gomes, D. Santos and N. Seco, "PAPEL: a dictionary-based lexical ontology for Portuguese". In 8<sup>th</sup> Intl. Conference (PROPOR). Vol. 5190, Springer Verlag, 2008, pp. 31-40.
- [10] N. Hernandez and J. Mothe, "D'un thesaurus vers une ontologie de domaine pour l'exploration d'un corpus", In Actes de la conférence Veille Stratégique Scientifique & Technologique, 2006.
- [11] G. Hirst, "Ontology and the Lexicon". In: Staab, Steffen and Studer, Rudi (Eds.), Handbook on Ontologies, Berlin: Springer, 2004, pp. 209-229.
- [12] ISO 24613: Lexical Markup Framework (LMF) revision 16. ISO FDIS 24613:2008.
- [13] Y. Ma, L. Audibert and A. Nazarenko, "Ontologies étendues pour l'annotation sémantique". In Actes d'IC 2009, Tunisie, 2009, pp. 205-216.
- [14] E. Montiel-Ponsoda, W. Peters, G. Aguado de Cea, M. Espinoza, A. Gómez Pérez, and M. Sini, "Multilingual and localization support for ontologies". Technical report, D2.4.2 Neon Project Deliverable, 2008.
- [15] Pazienza, M. T. and Stellato, A. 2006. Exploiting Linguistic Resources for building linguistically motivated ontologies in the Semantic Web. Second Workshop on Interfacing Ontologies and Lexical Resources for Semantic Web Technologies (OntoLex2006).

# An Ontology-based Configurator for Customized Product Information based upon the Slow Intelligence Systems Approach

Emilio Zegarra<sup>1</sup>, Francesco Colace<sup>2</sup>, Massimo De Santo<sup>2</sup>, Shi-Kuo Chang<sup>1</sup>

<sup>1</sup>Department of Computer Science, University of Pittsburgh Pittsburgh, PA 15260 USA

<sup>2</sup>Department of Information and Electrical Engineering, University of Salerno Fisciano (SA) Italy  
{ezegarra, chang}@cs.pitt.edu, {fcolace, desanto}@unisa.it

## Abstract

*The ability to create customized product configurations fulfilling user's needs is a major aim sought by many companies, especially the design and implementation of web-based services that allows users to create and customize in a simple and intuitive way a desired product. To define a common vocabulary that opens possibilities such as the ability to express customer requirements correctly and to exchange knowledge, a popular approach uses ontological formalisms. This paper presents an ontology-based configurator system that maximally satisfies user's needs starting from the desired requirements, the available components, the context information and previous similar configurations. The process by which the system finds the candidate configuration follows an approach known as Slow Intelligence (SI). This paper presents the proposed approach of SI ontological transformations and its application to product configuration.*

## 1 Introduction

In the past, companies relied on standard configurations that focused on lowering production costs while increasing profits. However, as new manufactures join the market, existing ones must find ways to attract new or retain existing customers. One method that is gaining popularity is product personalization. With personalization, users are able to obtain a product that meets their needs as opposed to most closely meeting their needs. This allows manufacturers to gain repeat customers resulting in higher profits. However, the ability to generate such a customized product is still challenging for most companies. Customers' request for personalized products prompts companies to consider mass customization, which brings a change in how product design is organized affecting the cost-efficiency of mass production. Many researchers believe that product configuration is an effective answer to this problem [1]. In this scenario, a customer is able to specify any kind of product configuration controlled from the point of view of its technical feasibility and cost. This approach is effective not only for products but also in the world of services: in fact, customer often requests a complex service starting from a portfolio of base services

(insurance services, travel services and so on). Therefore, the Product Configurator (Configurator) is becoming the killer application for the management of companies' business processes: in fact, customer orders drive the full chain of processes [2].

The Configurator, however, has also to furnish the best solution for the customer and check the actual feasibility of the product. In other words, a good Configurator aims to be the perfect seller being able to satisfy customers' needs 24/7. Product development based on customer preferences is important for obtaining larger market share and faster sales growth for organizations. The Internet has fostered the creation of e-businesses and the building of a real, interesting and distributed market. Many corporations have introduced on their websites interactive Configurators allowing customers to experiment with options and get support in the product selection process [3]. Even though this issue represents a very challenging and interesting research topic, there is a lack of research activities. Current Configurator implementations include simple rule-based systems, a hierarchical questionnaire or a passive system that collects with no logic the preferences of the customer. In these cases, the Configurator is just a product viewer for the customer and does not implement any reasoning logic or user adaptive approach. Customer requests are passively received without further reasoning, requiring the customer to be familiar with both the product structure and its functions and so often the final configuration is not the best. Recently, there have been some proposals [1][4][5] for the introduction of an Intelligent Configurator. They propose an ontology-based approach [6] that seems to be an effective methodology for the improvement of the actual configurator.

In this paper, a framework for the assisted product configuration, based on the use of ontology formalism and methodologies, is proposed. It works mainly by the use of four different ontologies: the functionality ontology obtained by the analysis of the user's request, the component's ontology obtained by the support of an expert, the context ontology and the ontology of previous configurations. Moreover, the proposed Configurator follows a Slow Intelligence System's approach.

Chang proposes the Slow Intelligence System (SIS) approach in [9]. SISs are general-purpose systems characterized by being able to improve performance over time through a process involving *enumeration, propagation, adaptation, elimination and concentration*. A SIS continuously learns, searches for new solutions and propagates and shares its experience with other peers. A SIS differs from expert systems in that the learning is not always obvious. A SIS seems to be a slow learner because it analyzes the environmental changes and carefully and gradually absorbs that into its knowledge base while maintaining synergy with the environment. From the structural point of view, a SIS is a system with multiple decision cycles such that actions of slow decision cycle(s) may override actions of quick decision cycle(s), resulting in poorer performance in the short run but better performance in the long-run.

In the SIS approach, the main processing blocks of enumeration, propagation, adaptation, elimination and concentration can each be aided by an ontology. The proposed ontology-based Configurator follows the SIS approach by composing these SIS blocks aided by ontologies to configure customized products and services. These SIS blocks are *ontological transformations*. This paper therefore also demonstrates the power of such SIS ontological transformations.

This paper is organized as follows: Section 2 explains in details the proposed Configurator, Section 3 presents a sample scenario and Section 4 presents a prototype and experimental results. Finally, Section 5 presents conclusions and future works.

## 2 Process Configurator Model

This section describes the design of the ontological Configurator. First, some general definitions on the configuration problem are introduced.

### 2.1 Definitions

**Definition 1:** the configuration problem (CP) is formulated as  $CP = \{C, P, C_T, R\}$ , where  $C$  is a set of components that may constitute a customizable product,  $P$  is a set of properties of components,  $C_T$  is a set of constraints imposed on components due to technical and economical factors and  $R$  is a set of customer requirements, specified in the form of constraints. The CP collects the user's request  $D_{UR}$

**Definition 2:** the configuration solution (CS) is defined as:  $CS = \{I, V\}$  where  $I$  is a set of individuals which are instances of components and  $V$  is a set of values which are assigned to properties of those individuals. The CS is

expressed by the description of the customized product  $D_{CP}$

**Definition 3:** the ontology of the functionalities ( $O_F$ ) defined as  $\{C_F, A_F, H_F, R_{TF}, R_F, A_{XF}\}$  where  $C_F$  is the concept set.  $c \in C_F$  expresses one concept and in each ontology there is a root concept marked as "Thing". In particular, for each  $c \in C_F$  there exists a descendant node set ( $C_{DN}$ ) containing all its lower layer concepts and an ancestry node set ( $C_{AN}$ ) containing all upper layer concepts.  $A_F$  is the concept attributes set. For  $c \in C_F$  its attributes set is expressed as  $A_C = \{a_1, \dots, a_n\}$  where  $n$  expresses the number of attributes related to  $c$ .  $H_F$  expresses the concept hierarchy set. The formalism  $(c_i, c_j)$  means that  $c_i$  is the sub-concept of  $c_j$ . In other words, this set contains the *is\_a* relations among the classes.  $R_{TF}$  is the set of semantic relation types.  $R_{TF} = R_{TFD} \cup R_{TFU}$ .  $R_{TFD}$  means the set of predefined relations (*same\_as, disjoint\_with, equivalent*) while  $R_{TFU}$  means the set of user defined relation types. The formalism  $(c_i, c_j, r)$  with  $r \in R_{TF}$  means that a relation  $r$  exists between  $c_i$  and  $c_j$ . The set  $RelR_{TF}(c_i, c_j)$  contains the relation  $r$  between  $c_i$  and  $c_j$ .  $R_F$  is the set of non-hierarchical relations. The formalism  $(c_i, c_j, r)$  with  $r \in R_F$  means that between  $c_i$  and  $c_j$  there is the  $r$  relation. The set  $Rel(c_i, c_j)$  contains the relation  $r$  between  $c_i$  and  $c_j$ ,  $A_{XF}$  is the set of axioms that are in the ontology. The aim of this ontology is the representation of the product's functionalities requested by the user.

**Definition 4:** the ontology of components ( $O_{CP}$ ) defined as  $\{C_{CP}, A_{CP}, H_{CP}, R_{TCP}, R_{CP}, A_{XF}\}$  where  $C_{CP}$  is the concept set.  $c \in C_{CP}$  expresses one concept and in each ontology there is a root concept marked as "Thing". In particular, for each  $c \in C_{CP}$  there exists descendant nodes set ( $C_{DN}$ ) containing all its lower layer concepts and an ancestry nodes set ( $C_{AN}$ ) containing all upper layer concepts.  $A_F$  is the concept attributes set. For  $c \in C_{CP}$  its attributes set is expressed as  $A_C = \{a_1, \dots, a_n\}$  where  $n$  expresses the number of attributes related to  $c$ .  $H_{CP}$  expresses the concept hierarchy set. The formalism  $(c_i, c_j)$  means that  $c_i$  is the sub-concept of  $c_j$ . In other words, this set contains the *is\_a* relations among the classes.  $R_{TCP}$  is the set of semantic relations type.  $R_{TCP} = R_{TCPD} \cup R_{TCPU}$ .  $R_{TCPD}$  is the set of predefined relations (*same\_as, disjoint\_with, equivalent*) while  $R_{TCPU}$  means the set of user defined relation type. The formalism  $(c_i, c_j, r)$  with  $r \in R_{TCP}$  means that between  $c_i$  and  $c_j$  there is the  $r$  relation. The set  $RelR_{TCP}(c_i, c_j)$  contains the relation  $r$  between  $c_i$  and  $c_j$ .  $R_{CP}$  is the set of non-hierarchical relations. The formalism  $(c_i, c_j, r)$  with  $r \in R_{CP}$  means that between  $c_i$  and  $c_j$  there is the  $r$  relation. The set

$Rel(c_i, c_j)$  contains the relation  $r$  between  $c_i$  and  $c_j$ ,  $A_{XCP}$  is the set of axioms that are in the ontology. The aim of this ontology is the representation of the product's components.

**Definition 5:** The ontology builder module builds an ontology starting from some inputs furnished by users or obtained from the environment. In particular, this module contains a meta-ontology representing in a very general way the expected ontology. The inputs are the nodes and the related relations that have to be considered in this meta-ontology in order to obtain the desired ontology. If some inputs are not represented in the meta-ontology the user will define the new nodes, the attributes and the relations with the other nodes of the meta-ontology.

**Definition 6:** the ontologies' comparison, simplifier and merging module manipulates the input ontologies in order to obtain as output an ontology representing them. In particular, the following operations are accomplished:

- Merging: the merging operation is defined as:  
 $M : O_1 \times O_2 \rightarrow O_{1 \cup 2}$ . The aim of this operation is to merge two ontologies. This operation first searches similar nodes in the two ontologies and then adds the nodes that are not both in them. The same operation is accomplished with the relations. The details of the merging operation follow:
  - Given the function  $M(O_A, O_B) = O_C$  with  $O_A = \{C_A, A_A, H_A, R_{TA}, R_A, A_{XA}\}$  and  $O_B = \{C_B, A_B, H_B, R_{TB}, R_B, A_{XB}\}$  the resulting  $O_C = \{C_C, A_C, H_C, R_{TC}, R_C, A_{XC}\}$  is created by:
    - Step 1: Set  $O_C = O_A$
    - Step 2:  $\forall c_i \in C_B \wedge C_A$  nothing is added to the  $C_C$  set
    - Step 3:  $\forall c_i \in C_B \wedge \notin C_A$  and with no relations with any  $c_j \in C_C$ , add  $c_i$  to the root of the  $C_C$  set.
    - Step 4:  $\forall c_i \in C_B \wedge \notin C_A$  with ancestor hierarchical relations with any  $c_j \in C_C$ , add  $c_i$  to  $c_j$
    - Step 5:  $\forall c_i \in C_B \wedge \notin C_A$  with descendant hierarchical relations with any  $c_j \in C_C$  add  $c_j$  to  $c_i$  and add if  $c_j$  has only the 'Thing' node as its parent then add  $c_i$  to the 'Thing' node else attach  $c_i$  to the parents of  $c_j$
    - If there are still some  $c_i \in C_B$  go to Step 2 else  $\forall c_i \in C_B$  added to the new ontology insert the relative attributes, axioms and the not hierarchical relations
- Simplifying: the simplifying operation is defined as:  
 $S : O_a \times O_b \rightarrow O_c$ . The aim of this operation is to simplify an ontology erasing nodes, relations or attributes. This operation involves the use of another ontology containing the information that will simplify the first one. The simplifying operation  $S(O_A, O_B)$  is composed by the following steps:
  - Step 1: erase all the nodes of  $O_A$  that are not in  $O_B$  or that are not related to nodes that are in  $O_B$
  - Step 2: erase all the relations among nodes of  $O_A$  that are not in  $O_B$
  - Step 3: erase all the attributes that are in the nodes in  $O_A$  and not in  $O_B$
- Comparing: the comparing operation is defined as:  
 $C : O_a \times O_b \rightarrow R$ . The aim of this operation is to compare two ontologies giving a positive or negative grade that represents the number of similar nodes, relations and attributes that are between the two ontologies. The comparison function  $C(O_A, O_B)$  is composed by the following steps:
  - Step 1: count all the nodes of  $O_A$  that are in  $O_B$ . For each shared node, increase the similarity node counter (SNC) by a unit. At the end divide the SNC by the number of different nodes that are in  $O_A$  and  $O_B$
  - Step 2: count all similar relations among nodes that are both in  $O_A$  and in  $O_B$ . For each shared relations increase the similarity relations counter (SRC) by a unit. At the end divide the SRC by the number of different relations that are in  $O_A$  and  $O_B$
  - Step 3: count all similar attributes of the nodes that are both in  $O_A$  and in  $O_B$ . For each shared attribute, increase the similarity attributes counter (SAC) by a unit. At the end divide the SAC by the number of different attributes that are in  $O_A$  and  $O_B$
  - Step 4: calculate the function Ontology Similarity Index represented as:  $OSI = \alpha SNC + \beta SRC + \gamma SAC$ . The indexes ( $\alpha$ ,  $\beta$ , and  $\gamma$ ) represent the weights controlling the importance of a term.
- Selection: the selection operation is defined as:  
 $Sel : O \times O \times T \rightarrow O$ . The aim of this operation is to select an ontology according to the grade  $t \in T$  obtained by the use of the comparing function. This function uses the comparison function previously described to calculate the similarity index that will be compared to the threshold. In the case of the function  $Sel(O_A, O_B, t)$ , if the obtained index is bigger than the fixed threshold  $t$  the output will be the ontology with the maximum number of concepts, setting the attribute to the common nodes to the values that are otherwise in the ontology  $O_A$ .

- Deletion Node: the deletion node operation is defined as:  $DN : O \times O \rightarrow O$ . The operation  $DN(O_A, O_B)$  deletes all the nodes in  $O_A$  that are in  $O_B$
- Deletion Relation: the deletion relation operation is defined as:  $RN : O \times R \rightarrow O$ . The operation  $DN(O_A, R_i)$  delete all the relations  $R_i$  among the nodes of  $O_A$
- Set Attribute: the set attribute operation is defined as:  $SA : O \times O \rightarrow O$ . The operation  $SA(O_A, O_B)$  set the attributes for each node  $C_i$  which belongs both to  $O_A$  and  $O_B$  to the value expressed in  $O_B$

## 2.2 System architecture

The proposed architecture is based on a four-layer architecture as shown in Figure 1.

User Layer	User Request
Functionality Layer	Functionalities' Ontology
Component Layer	Components' Ontology
Instance Layer	Configuration Instances

**Figure 1: Four-Layer architecture**

The first layer expresses the requests furnished by the user. The requests are collected either from a graphical user interface or from the command line. The second layer maps the collected requests to the functionalities that the product has to have. The analysis of the user's requests infers the product's functionalities expressed by the use of the ontological formalism. The third layer selects the components satisfying the requested functionalities and adopts the ontological formalism for their representation. The last layer translates the components' ontology to the final configuration of the product. In this layer, the output is an XML file that can be used to build the customized product.

## 2.3 System Framework in the context of SIS

The implementation of the previous layered architecture adopts the SIS approach. Figure 2 depicts the framework. This system is composed of SIS blocks as *ontological*

*transformations* to configure customized products and services. The following main SIS blocks are identified:

- Enumerator Block: this block has as inputs the user request and the product definition and as output the ontology  $O_{AP}$ . It has two ontology builder components as well as comparison, merging and simplifier modules. This block can be characterized by the following function:  $F_{EN} : O_F \times O_{CP} \rightarrow O'_{AP}$ . At the end of this block, the ontology representing a first rough version of the customized product is obtained.
- Adaptor Block: this block has as inputs the ontology  $O'_{AP}$ , representing a first rough version of the customized product, and the information about the context and as output the ontology  $O''_{AP}$ . It has as components an ontology builder as well as a comparison, merging and simplifier modules. This block can be characterized by the following function:  $F_A : O'_{AP} \times O_{EN} \rightarrow O''_{AP}$ . At the end of this block, the ontology representing a context-adapted product is obtained.
- Eliminator Block: this block has as inputs the context-adapted ontology  $O''_{AP}$  and product ontologies developed in the past by other similar Configurators that work in other part of the system. The aim of this block is the tuning of the  $O''_{AP}$  according to the previous configurations obtained in similar contexts. This block is characterized by the following function:  $F_{EL} : O''_{AP} \times S_{OPA} \rightarrow O_{AP}$ . The output of this block is the ontology  $O_{AP}$  that represents the adapted product.
- Concentrator Block: this block has as inputs the ontology  $O_{AP}$  and as output the configuration of the product. The aim of this block is the generation of an XML file representing the ontology of the customized product. This block can be characterized by the following function:  $F_C : O_{AP} \rightarrow D_{CP}$

In this way, the configuration problem CP can be formulated in its general formulation as the composition of ontological transformations:  $F_C(F_{EL}(F_A(F_{EN}(U_R, U_P))))$ . Similar to a SIS, the proposed Configurator can follow a slow and a fast process of solution inference. So, the previous formulation can be defined as the slow process, while the fast process can be defined as a simplified sequence of ontological transformations:  $F_C(F_{EN}(U_R, U_P))$ .

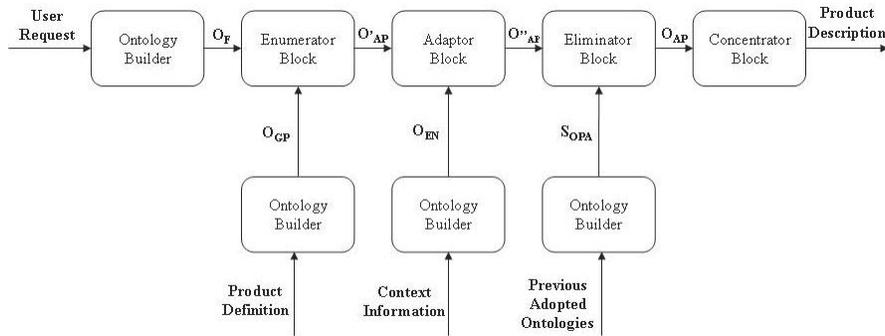


Figure 2: The System Architecture

### 3 A Scenario

To demonstrate how the system works, this section presents an example of configuring a personal computer. A customer would like to buy a Personal Computer in order to play videogames and surf on the internet. He knows that he needs an operating system, a web browser and an antivirus package. In particular, the user prefers a Microsoft Windows operating system. He lives in the United States and prefers to have a desktop. He also prefers low cost components. Using the graphical user interface, the user can easily define the ontology  $O_F$  that contains the required functionalities. The general ontology  $O_{GP}$  contains all the functionalities and components involved in a personal computer. So these ontologies,  $O_{GP}$  and  $O_F$ , can be used in function  $F_E$  obtaining the ontology  $O_{AP}$ .

At this point, the system uses the Adaptor module. In particular, the aim of this module is to introduce new concepts to the set of attributes of the  $O_{AP}$  according to the information obtained by the environment where the product will work and from experts. In the proposed configuration, there is a node for “keyboard” with the following attributes: US English layout, Italian layout, Japanese layout and so on. In this case, the attribute US layout is selected and the other ones are deleted. After this phase completes, the ontology  $O''_{AP}$  is obtained and the eliminator block can begin. As previously, said this module has the aim, by the use of a selector function, to compare the ontology  $O''_{AP}$  with those developed by other Configurators that work in the environment as well as by the same Configurator in the past. If for example a previous developed configuration has the same

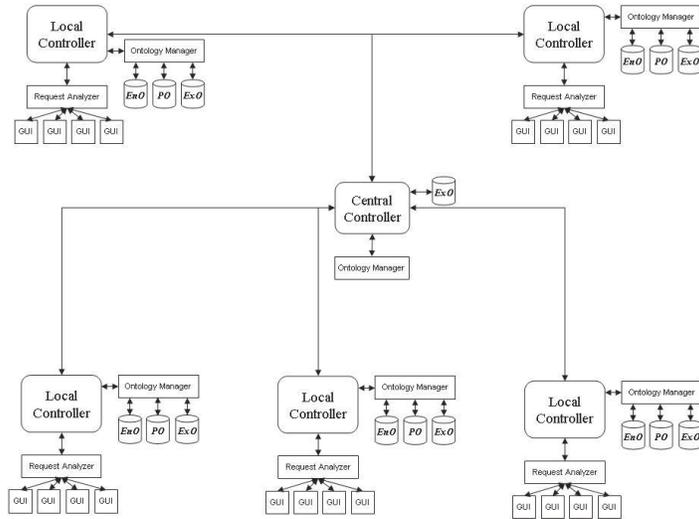
configuration as  $O''_{AP}$  but contains a node representing a component for a particular kind of peripheral such as a Joystick which improves the video game playing experience, then this ontology becomes the new ontology  $O_{AP}$ . At the end of this module, the concentrator block will define the final configuration by selecting the components that correspond to the leaves in the ontology  $O_{AP}$ . Each node, representing real components, contains attributes on the main characteristics of the products (price, colour, connection, etc).

The selection task is accomplished according to the information furnished by the user (i.e. select the cheaper components), by the comparison of previous configuration furnished in the past and by the rules codified in the ontology (i.e. an Intel processor needs an Intel motherboard). At the end of the process, the system generates an XML file containing the final configuration of the personal computer. The graphical user interface presents this file to the customer, in a user-friendly manner, giving the user the opportunity to check the correctness of the configuration. The final configuration of the product can use the same XML file.

## 4 Experiment setup and results

### 4.1 Experiment prototype

This section describes a first prototype of the ontological Configurator. The proposed prototype has to satisfy and offer the services previously described according to a SIS approach. Figure 3 depicts the topology of the prototype environment.



**Figure 3: Ontology Configurator Prototype**

Five instances of the Configurator work in separate operating contexts. Each instance of the configuration represents a different geographical area and a well-defined operating environment. Users submit requests for configurations by expressing the functionalities and planned uses of the personal computer using a graphical user interface. A Request Analyzer component analyzes the requests and extracts the concepts in order to build the ontological representation of the desired personal computer. The building process carries on using the approach developed in [10]. The resulting user ontology represents the desired functionalities of the user. The Simplify Ontology function uses the user ontology and the general personal computer ontology, defined by experts, to identify the functionalities and the required components requested by the user. The process to identify the requested functionalities from the user ontology also involves the discovery of dependent functionalities and components. The Local Controller sends the request to the Ontology Manager, which builds the ontology representing the recommended configuration. The Ontology Manager compares the ontology obtained by the analysis of the user's request with previous ontologies available from the Previous Ontology (PO) database, the ontology suggested by experts available from the Expert Ontology (ExO) database, and the Local Environment Ontology (EnO) database, which contains the ontology with the allowed attributes for a particular environment.

The Central Controller enables the communication between separate instances of the system. It also serves as a repository of a global Expert Ontology (ExO) database.

As previously said we introduced five different instances of the system that work on the following scenarios: an Italian Computer Store, an American Computer Store, a British Computer Store, an Indian Computer Store and a Japanese Computer Store.

For each of these “virtual” stores, an expert described the ontology of the context and stored it in the EnO database of the Local Controller. This context ontology is updated after a pre-defined period or when some event occurs that requires changes in the definition of the environment.

A set of common computer configurations based on usage scenarios were identified for evaluation. A group of experts was asked to define the functionalities and components required for a configuration to satisfy each particular usage of a personal computer. These experts were selected among people that work in the field of Computer Science and in particular are University Professors and are familiar with computer configurations. The allowed configurations for the personal computer are:

- Play\_Videogame (PV)
- Web\_Surfing (WS)
- Online\_Gaming (OG)
- Multimedia\_Design (MD)
- Computer\_Aided\_Design (CAD)
- Music (MUS)
- Word\_Processing (WP)
- School\_Work - Web\_Surfing and Word\_Processing (SW)

Each of these configurations represents a well-defined ontology structure with relative values for the attributes. Obviously not all the attributes related to the nodes of the ontology have a value because some of them will be defined by the analysis of the user request.

## 4.2 Evaluation parameters

To measure the efficiency of our suggested models we plan to gather data from two types of evaluations: user evaluation and system evaluation

### 4.2.1 User evaluation

The user evaluation method will interview users of the system to measure the satisfaction with the proposed configurations. This process asks them to select their desired configuration. After the system generates the XML file of the desired configuration, the users review and provide feedback on the solution. A User Satisfaction Index (USI) captures the feedback from users and represents how well the user requested functionalities are met by the suggested configuration. The range of values are [0, 1] inclusive. A value of 1 indicates that all user's requested functionalities are fully satisfied by a configuration solution. The USI index is calculated as:

$$USI = \frac{\text{Total number of functionalities}}{\text{Total number of satisfied functionalities}}$$

#### 4.2.2 System evaluation

In order to evaluate the quality of the obtained results quantitatively, the following parameters are introduced:

- **Functionality evaluation index (FEI):** this parameter expresses if the proposed configuration is correct or not. A correct functionality meets the functionality's constraints and is valid based on the experts' configuration. The FEI is obtained in this way:

$$FEI = \frac{\text{Total number of correct components}}{\text{Total number of components}}$$

- **Experts evaluation index (EEI):** the experts evaluate the obtained configuration and determine if it satisfies the request of the user or not. The results report the percentage of correct configurations.
- **Ontology similarity index (OSI):** this index measures the average similarity of the obtained ontology with the proposed one by the expert in the same category. In particular, the Comparing Operation, introduced in section 2, will be adopted. In this case the indexes  $\alpha$ ,  $\beta$ , and  $\gamma$  assume the following values: 0.5, 0.3, 0.2.
- **Ontology Evolution Index (OEI):** this index represents the variation of the OSI after each configuration. In other words, this parameter expresses the evolution of OSI index for a category after a new request.

### 4.3 Experimental results

We began the experimental phase by submitting to each Configurator about fifty configuration requests developed by fifty different users by the use of online forms. At the beginning we selected for each configurator only the requests for the same kind of configuration. In other words, we tried to specialize the Configurator for each kind of configuration. The obtained results are presented in Table 1

Store	Request	FEI	EEI	OSI	OEI (obtained after 50 requests)	USI (avg. value obtained after 50 requests)
Italian	PV	1.00	0.94	0.85	0.14	0.87
American	WP	1.00	0.86	0.83	0.11	0.88
British	CAD	0.96	0.88	0.79	0.17	0.83
Indian	MD	1.00	0.90	0.93	0.07	0.97
Japanese	WS	1.00	0.92	0.91	0.06	0.98

Table 1: Evaluation results for single configuration type

For this kind of request, the system shows good results. In particular, the FEI, the EEI and the OSI indexes show good values for each considered case. The worst case is for the British store that has to configure a personal computer for Computer Aided Design. In fact this configuration can easily be confused with the ones for "multimedia design" or "play videogame". The system reaches the best configuration after a reasonable number of requests, as shown by the OEI index. Users' evaluations show a good USI index. It is important to underline that the various systems work independently because they have no reason to exchange information with each other. In fact, they start to work on a well-defined kind of configuration and specialize themselves in the resolution of this problem.

Normally, not all users of a store would request the same personal computer configuration. Therefore, a new series of requests, developed by other 50 users, were submitted to the system but in this case, mixes of configurations were requested from the stores. The aim of this second evaluation is to determine how the Configurator updates its knowledge to account for not just a single type of configuration but also multiple types. Table 2 presents the results.

Store	Request	FEI (% of yes)	EEI (% of yes)	OSI (obtained after 100 requests)	OEI (obtained after 100 requests)	USI (avg. value obtained after 100 requests)
Italian	Mixed	0.93	0.89	0.83	0.14	0.83
American	Mixed	0.87	0.82	0.81	0.13	0.84
British	Mixed	0.93	0.87	0.73	0.18	0.80
Indian	Mixed	0.97	0.88	0.87	0.14	0.93
Japanese	Mixed	0.95	0.90	0.90	0.09	0.94

Table 2: Evaluation results for mixed configuration types

In this case, the results start to be worse than the first ones. In fact, in this case the various systems have to solve different types of configurations. At the beginning, the answers to the requests are not the best and the customization of the configuration for the new environments sometimes does not work well. This is the reason for the decrease of the FEI and EEI indexes. In addition, the OSI and OEI indexes show lower values because when the system has to answer to a new request it begins with previously obtained configurations and tries to customize them to the new case. So in this way the first configurations are not very good. At the same time, according to the SIS approach, the system asks for help to other systems that are working in other environments

obtaining information about how they faced the same problem. It is interesting to show how the system reacts to the changes in request types. To show the evolution in the determination of the best configuration a detailed analysis of the OSI index is presented. Figures 4 and 5 show the evolution of the OSI index obtained for the first 50 similar requests and for the 100 mixed requests respectively. Figure 4 shows how the systems, which acted initially on a single type of request, improve the proposed configuration request by request. In particular, after about 30 requests each system is able to achieve an OSI value in the range of 0.8 – 1.

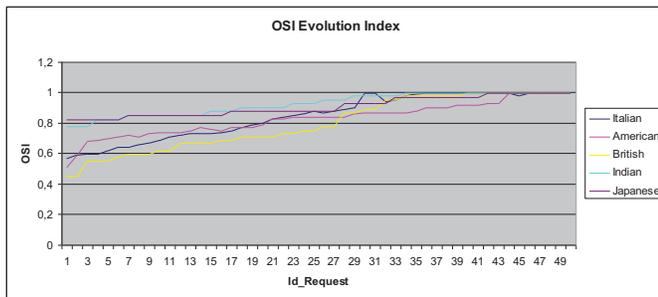


Figure 4: OSI Evolution Index for the first 50 requests

Figure 5 shows the OSI index after 50 requests of similar type followed by 50 additional requests of mixed type were submitted to the system. Following the first 50 similar requests, the system started to receive requests of different types and began to communicate with the other systems. In this case after a first phase where the system shows not good OSI results it is able to improve quickly its performance. In fact thanks to the interaction with the other systems it shows OSI values again in the range 0.8 - 1 after processing about 20 new requests of mixed type (after the first fifty). In this case, the performance improves because there is an effective exchange and sharing of data with the other systems.

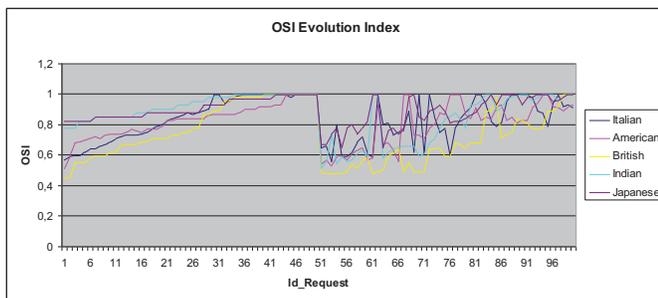


Figure 5: OSI Evolution Index for the first 100 requests

## 5 Conclusion

This paper presented an ontology-based Configurator system that finds the configuration that maximizes the users' needs starting from the desired requirements, the available components, the context information and

previous similar configurations. It follows the Slow Intelligence approach by composing the SIS blocks as ontological transformations to enumerate, adapt, eliminate and finally select the best configuration. Experimental results demonstrate the ability of the product Configurator to acquire new knowledge over time yielding in an improved product configuration. The paper also demonstrates the use of ontological formalisms to represent knowledge about the product configuration problem, and the power of the ontological transformations. Future research topics include the continuous improvement of the ontological transformations through evolutionary ontology, the process of exchanging knowledge between systems and studying algorithms that allow for the processing of knowledge as the amount of data increases. As the number of systems grows, so does the knowledge base. Therefore, it is important to devise efficient algorithms for processing the data in an efficient manner.

## 6 References

- [1] X. Ding, "Product Configuration on the Semantic Web Using Multi-Agent", Networking, Sensing and Control, 2008. ICNSC 2008. IEEE International Conference on, pp. 304 – 309, 2008
- [2] A. Felfernig, G. Friederich, D. Jannach, M. Zanker, "Semantic Configuration Web Services in the CAWICOMS Project", 6th IEEE International Symposium on Wearable Computers, ISWC 2002, LNCS 2342, pp. 192-205, 2002
- [3] N. Franke, F. T. Piller, "Configuration Toolkits for Mass Customization: Setting a Research Agenda", International Journal of Entrepreneurship and Innovation Management, No. 1, 2003
- [4] J.M. Park, J.H.Nam, Q.P. Hu, H.W. Suh, "Product Ontology Construction from Engineering Documents", Smart Manufacturing Application, 2008. ICSMA 2008. International Conference on, pp 305 – 310, 2008
- [5] H. Youliang, N. W. Keong, L. Haifeng, L. Wenfeng, S. Bin, L. Xiang, "Semantic Modeling and Extraction for Cross-family Product Configuration, Service Systems and Service Management, 2007 International Conference on, pp 1-6, 2007
- [6] T. Jepsen, , "Just What Is an Ontology, Anyway?", IT Professional, vol. 11, no. 5, pp. 22-27, Sep./Oct. 2009
- [7] O. Corcho, , A. Gómez-Pérez, , "A Layered Model for Building Ontology Translation Systems," International Journal on Semantic Web and Information Systems, 1(2): 22-48, 2005
- [8] P. Haase, , L. Stojanovic, , "Consistent evolution of OWL Ontologies", ESWC, Lecture Notes in Computer Science, vol. 3532, Springer, pp. 182-197
- [9] S.K. Chang, "A General Framework for Slow Intelligence Systems", International Journal of Software Engineering and Knowledge Engineering, Volume 20, Number 1, February 2010, pp. 1.
- [10] F. Clarizia, F. Colace, M. De Santo, P. Napoletano, "Semantic Indexing of Web Pages Via Probabilistic Methods – In Search of Semantics Project", in proceedings of ICEIS 2009, Milano, 2009

# UFOCoRe: Exploring Fuzzy Relations According to Specific Contexts

Mauricio Jacó Cerri, Cristiane A. Yaguinuma, Marcela Xavier Ribeiro, Marilde T. P. Santos  
Department of Computer Science, Federal University of São Carlos (UFSCar)  
P.O. Box 676 – 13565-905 – São Carlos – Brazil  
{mauricio\_cerri, cristiane\_yaguinuma, marcela, marilde}@dc.ufscar.br

**Abstract:** In this paper, we propose the UFOCoRe approach to represent multiple relationship strengths in only one domain ontology, in order to enable dynamic data analysis according to the context. An algorithm called context-NARFO was developed to support the proposed method to mine rich semantic association rules. This algorithm was employed to mine real data from a medical insurance company, producing expressive and meaningful patterns. Moreover, the results show significant semantic gain in the data patterns mined, which is a consequence of the data analysis performed according to the user view and needs. The proposed method also promotes an increase in productivity during the ontology building phase, since domain experts can analyze several contexts of the modeled domain in a single and straightforward way.

**Keywords:** Context Ontology, Fuzzy Ontology, Knowledge Representation, Data Mining

## I. INTRODUCTION

Several real-world situations require the knowledge of different specialists and distinct points of view to obtain a suitable representation and understanding of the conditions or actions that must be considered to solve some problem. In this paper, we defined one point of view as one context. To better understand the applicability of different contexts when analyzing two individuals, consider the problem of comparing two vegetables in two different contexts, e.g. tomato and khaki. If we consider appearance, it is possible to come out "Tomato is very similar to khaki". But, regarding biological classification, we can say "Tomato is bit similar to khaki". Appearance and biological classification represent two possible points of view (contexts) that can be employed to define the similarity between those vegetables.

Usually, the concept *Context* is associated to context-aware systems which are able to adapt their operations to the current context. For example, in [12] the context instances are classified in two different dimensions, referring to physical and logical context. In that work, physical dimension refers to the context that can be measured by hardware sensors, i.e., location, light, temperature or air pressure, whereas logical context is mostly specified by the user or captured by monitoring user interactions, i.e., user's goals, tasks, working context, business processes, user's emotional state.

Most applications use traditional ontologies, which capture only complete or precise information, as they are based on classic logic. However, some domains may involve imprecise relationships such as the similarity relation [10] that expresses

the degree to which individuals are similar. Aiming to represent such kind of information, fuzzy set concepts [9] have been incorporated into ontologies, resulting in the so-called fuzzy ontologies [13]. In general, fuzzy ontologies model only a single fuzzy degree to each fuzzy relationship that involves the same set of individuals. For example, the similarity between  $x$  and  $y$  individuals can be represented as  $\text{sim}(x,y) = 0.7$ , but this value remains fixed for any application that uses the ontology, regardless of context.

Data mining is a key step of knowledge discovery in large databases [1]. One important topic in data mining research is concerned with the discovery of interesting association rules [2]. There is an increasing number of data mining approaches that concerns semantics of mined data, aiming to improve the quality of obtained knowledge. In this sense, ontologies have been employed to represent semantic information defined by knowledge experts, and can also be applied to enhance association rule mining. Some works [3] [4] have extended the mining process to obtain association rules that represent relationships among basic data items as well as among items at any level of the related taxonomy or ontology, resulting in the so-called generalized association rules.

Some applications that employ ontologies, including data mining ones, demand the representation of distinct strengths to the same relationship when considering different contexts. In medical domain, an example of such application is the ICD-10 ontology (International Statistic Classification of Diseases and Related Health Problems) [8]. In this ontology, a domain expert may define that the similarity between Neonatal aspiration syndromes and Interstitial emphysema and related conditions originating in the prenatal period diagnoses, in the context of child mortality is 0.9, while it decreases to 0.7 in the context of inpatient rates. Thus, a medical application based on this ontology can reason over these relationships according to the most suitable context (whether child mortality or inpatient rates). Some approaches address this problem by using two or more ontologies to express relationship strengths regarding several contexts. In this research, we will tackle this problem by employing a single domain ontology that models different contexts.

In this sense, we propose the Upper Fuzzy Ontology With Context Representation (UFOCoRe), a new approach that represent multiple relationship strengths in a single ontology, so that it is possible to express different relationship semantics



for vegetables that are similar to khaki regarding flavor, then such fuzzy degree may not be suitable for answering this query. Therefore, some real-world applications require a model to represent not only fuzzy relations but also the contexts to which they belong, in order to provide better information accuracy. Aiming to deal with this shortcoming, next section presents a new approach based on a single ontology for modeling fuzzy relations in different contexts.

### B. Association Rules

Association rule mining is a descriptive task of data mining, whose goal is to find interesting relationships among data items. The problem of mining association rules was firstly stated in [7] as follows. Let  $I = \{i_1, \dots, i_n\}$  be a set of literals called items. Let  $D$  be a set of database transactions where each transaction  $T$  is a set of items such that  $T \subseteq I$ . Let  $A$  be a set of items. A transaction  $T$  is said to contain  $A$  if and only if  $A \subseteq T$ . An association rule is an expression of the form  $A \rightarrow B$ , where  $A$  and  $B$  are itemsets, and  $A \subset I$ ,  $B \subset I$ , and  $A \cap B = \emptyset$ . The itemset  $A$  is called body or antecedent of the rule, and  $B$  is called head or consequent of the rule. The support of a rule  $A \rightarrow B$  is the percentage of transaction in  $D$  that contains  $A \cup B$ . The confidence of rule  $A \rightarrow B$  has confidence is the percentage of transactions in  $D$  containing  $A$  that also contain  $B$ . The problem of mining association rules, as it was firstly stated, involves finding rules that satisfy the restrictions of minimum support and minimum confidence specified by the user.

NARFO (*Mining Non-redundant and Generalized Association Rules Based on Fuzzy Ontologies*) [6] is an algorithm that mines generalized association rules applying a fuzzy ontology as background knowledge. NARFO generalizes the items, considering the infrequent itemsets (set of items). It uses the measures minimum support (*minsup*), minimum confidence (*minconf*), maximum redundancy (*R-interest*) and the minimum similarity (*minsim*) to mine the rules. The *R-interest* measure is employed to eliminate redundant and inconsistent rules and the minimum similarity is used to eliminate the items whose similarity degrees are less than the *minsim* parameter. For example: if the parameter *minsim* is defined with the value 0.7 (*minsim*=0.7), and the instances have the similarity degree = 0.1, then these items are eliminated and are not employed to generate the rules.

### III. PROPOSED METHOD: THE UPPER FUZZY ONTOLOGY WITH CONTEXT REPRESENTATION (UFOCoRE)

As stated before, the fuzzy ontology does not support the representation of fuzzy relations involving several contexts. Thus, we have enhanced that ontology to model, in a single ontology, distinct relationship strengths according to different contexts. In this sense, we propose the *Upper Fuzzy Ontology with Context Representation* (UFOCoRE), an upper ontology whose concepts and relationships should be inherited and/or instantiated by domain ontologies. This ontology does not extend OWL language syntax, so existing OWL DL reasoners and editors keep compatible and can be straightforwardly used by domain experts to model fuzzy relationships in specific contexts. Notice that such reasoners do not support fuzzy

inference, but [13] implements some reasoning tasks such as transitivity and symmetrical relationship checking. UFOCoRE can serve as a supporting tool for many applications that demand context inference within an ontology, which represents an important step toward the W3C's vision for the Semantic Web. Fig. 3 shows the constructs added to the fuzzy ontology (identified by *ctx:* prefix) used to model distinct relationship strengths according to different contexts.

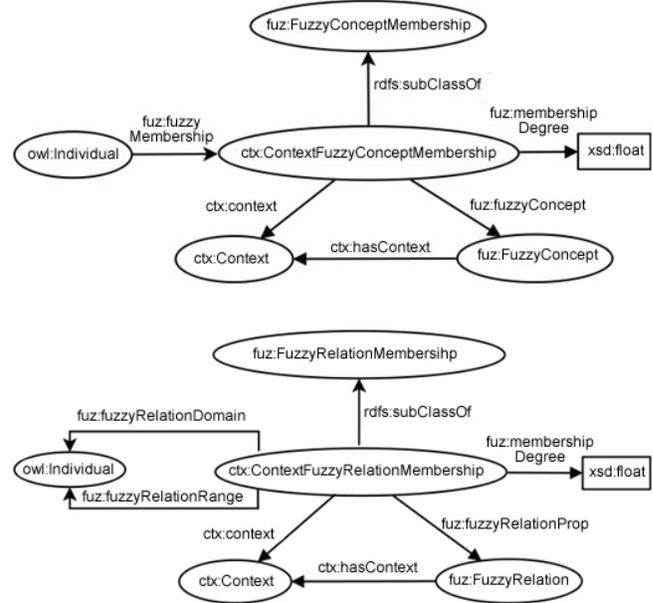


Fig. 3. Modelling context, fuzzy concepts and relationships in UFOCoRE.

In Fig. 3, we incorporate context information in two situations: when modeling fuzzy concepts (or classes) and when representing fuzzy relationships. Regarding fuzzy concepts, we introduce the *ctx:ContextFuzzyConceptMembership* class, which is subclass of the *fuz:FuzzyConceptMembership* class from the fuzzy ontology. For this reason, *ContextFuzzyConceptMembership* inherits properties such as *fuz:fuzzyMembership*, *fuz:membershipDegree* and *fuz:fuzzyConcept*, as illustrated in Fig. 3. Contexts are modeled by the *ctx:Context* class, which represents the different perspectives considered in domain ontologies. Then, a fuzzy class can be associated to several contexts (*ctx:hasContext* relationship between *fuz:FuzzyConcept* and *ctx:Context*). In order to represent different membership degrees depending on specific contexts, we add the *ctx:context* relationship between *ctx:ContextFuzzyConceptMembership* and *ctx:Context* classes. These new constructs make it possible to model several contexts and their respective membership degrees.

With regard to fuzzy relationships, we introduce the *ctx:ContextFuzzyRelationMembership* class (Fig. 3), responsible for associating fuzzy relationships to several contexts. *ctx:ContextFuzzyRelationMembership* is subclass of the *fuz:FuzzyRelationMembership* class from the fuzzy ontology, thus it inherits *fuz:fuzzyRelationDomain*, *fuz:fuzzyRelationRange*, *fuz:fuzzyRelationProp* and *fuz:membershipDegree* properties. The context association is represented by *ctx:hasContext* and *ctx:context* properties,

which link contexts to fuzzy relationships (*fuz:FuzzyRelation*) and fuzzy degrees (*ctx:ContextFuzzyRelationMembership*), respectively. By using such constructs, a domain expert can model fuzzy relationships from different perspectives, with specific fuzzy degrees according to each context.

To model an example of fuzzy relationship involving different contexts, consider the vegetable ontology illustrated in Fig 4. As we have pointed before, the fuzzy ontology is not able to represent the distinct similarity degrees between tomato and khaki when appearance and flavor contexts need to be analyzed. With the proposed UFOCoRe approach, domain experts can model such semantics, as presented in Fig.4 under OWL syntax. In this figure, we define appearance and flavor contexts as instances of *ctx:Context* class defined on UFOCoRe. Then, we model two instances of *ctx:ContextFuzzyRelationMembership* so that tomato and khaki can have different similarity degrees according to defined contexts. In this example, tomato is similar to khaki with degree 0.7 considering appearance context, while such relationship has degree 0.3 under flavor perspective. By analyzing these semantic constructs, applications can achieve better data accuracy, by making suitable inferences depending on the context required by the user.

```

<!-- Contexts -->
<ctx:Context rdf:ID="appearance"/>
<ctx:Context rdf:ID="flavour"/>

<!-- Classes -->
<owl:Class rdf:ID="Vegetable"/>

<!-- Instances -->
<Vegetable rdf:ID="tomato">
  <fuz:similarTo_directly rdf:resource="#khaki"/>
</Vegetable>

<Vegetable rdf:ID="khaki"/>

<!-- Fuzzy relationships in different context -->
<ctx:ContextFuzzyRelationMembership>
  <fuz:fuzzyRelationProp rdf:resource="#fuz:similarTo_directly"/>
  <ctx:context rdf:resource="#appearance"/>
  <fuz:fuzzyRelationDomain rdf:resource="#tomato"/>
  <fuz:fuzzyRelationRange rdf:resource="#khaki"/>
  <fuz:membershipDegree rdf:datatype="xsd:float">
    0.7</fuz:membershipDegree>
</ctx:ContextFuzzyRelationMembership>

<ctx:ContextFuzzyRelationMembership>
  <fuz:fuzzyRelationProp rdf:resource="#fuz:similarTo_directly"/>
  <ctx:context rdf:resource="#flavour"/>
  <fuz:fuzzyRelationDomain rdf:resource="#tomato"/>
  <fuz:fuzzyRelationRange rdf:resource="#khaki"/>
  <fuz:membershipDegree rdf:datatype="xsd:float">0.3
</fuz:membershipDegree>
</ctx:ContextFuzzyRelationMembership>

```

Fig. 4. Appearance and flavor contexts represented in vegetable ontology.

Different from the multi-ontology approach, the UFOCoRe method is based on a single ontology to model contexts, which avoids the definition of multiple ontologies representing the same domain. Hence, domain experts can model contexts and fuzzy relationships by editing a single ontology in a straightforward way. Next section describes a real-world application of our approach, regarding a medical ontology applied in a data mining task.

#### IV. EXPERIMENTS

In this section we describe the methodology employed to apply the *Upper Fuzzy Ontology with Context Representation* (UFOCoRe) to support knowledge discovery in medical data

involving patients and exams from a Brazilian Health Insurance Company.

##### A. Data Gathering and Mining Algorithm

The analysis of medical data involves to apply mining algorithms that leads with complex data (images, exams and diagnosis) for discovery useful and interesting patterns. An important concern of healthcare corporations today is to promote a preventive patient care, avoiding possible costs to treat illnesses already suffered by the patient, which could be prevented or controlled. In fact, the prevention of illnesses usually requires that several medical exams be performed by the patients, which also increases the cost of the medical care, since many exams are becoming more complex and expensive. In order to control such high cost, some Health Insurance companies employs a security parameter to avoid the fee of the execution of unnecessary exams. Analyzing this context, the UFOCoRe-based Medical Fuzzy Ontology (MedFuzOnto) was built. As the background of this ontology, it was used the formalized ontology ICD-10 (International Statistical Classification of Diseases and Related Health Problems), developed by The United Nations World Health Organization (WHO) [8].

The MedFuzOnto was employed by a new mining algorithm called Context-NARFO (discussed in the next subsection) to work with different context and to support fuzzy ontology. The experiment was carried on data of more than 40,000 patients' data of a Brazilian Health Insurance Company. The patients' data are composed of the whole medical exams, hospital procedures and medical care treatments made by these patients in the period between January/2009 and January/2010. A total of 627,618 records of patient's data were collect by the company in this period. For experimental purposes, we selected 16 values of the standard ICD-10, 21 medical specialties, 10 groups of age, 3 types of medical attendance and 3 types of medical providers to employ in this case study. A total of 36,093 patient's records where employed. This amount of records represents approximately 6% of all base company's data in the period.

The MedFuzOnto was built to express similarities between different diseases ICD-10 from the view of Group Similarity, Ancillary Service of Diagnosis and Therapy, Internment, Cost and Medical Specialty (see Fig. 5). The Group Similarity defines the similarity between two ICDs, named A and B, considering the closeness in the ICD-10 ontology. Ancillary Service of Diagnosis and Therapy represents the similarity between A and B considering the quantities of medical exams executed. Internment represents the similarity considering the number of internments provoked by A and B. Cost represents cost similarity of each ICD-10, and Medical Specialty represents the similarity considering the number of medical specialties required by A and B. Each similarity degree considering each of the five contexts was defined by experts and the model was built. Fig. 5 shows examples of the similarities degree between ICD -10 used in the model. We can observe different similarity degrees assigned to the same pair of ICD according to the context.

Similarity Contexts						
ICD10	ICD10	Context1	Context2	Context3	Context4	Context5
1	2	GROUP SIMILARITY	ANCILLARY SERVICE OF DIAGNOSIS AND THERAPY	INTERMENT	COST	MEDICAL SPECIALTY
H650	H669	0.9	0.5	0.1	0.5	0.6
K590	M255	0.6	0.1	0.5	0.8	0.5
Z010	Z340	0.8	0.5	0.9	0.7	0.3
Z001	J304	0.1	0.4	0.9	0.9	0.3
L811	M545	0.6	0.7	0.1	0.4	0.1
N859	M779	0.6	0.6	0.8	0.7	0.2
J303	I119	0.7	0.1	0.9	0.8	0.1
M542	M751	0.8	0.7	0.4	0.6	0.9

Fig. 5. Table of Similarity Degree of each Context.

Fig. 6 shows a part of the code of the MedFuzOnto which represents the similarTo relationships between the ICD-10 H650 and ICD-10 H669 (first line illustrated in Fig. 5) according to context 3 (Internment). The similarity degree is set to 0.1.

```
<ctx:ContextFuzzyRelationMembership>
  <fuz:fuzzyRelationProp rdf:resource="#fuz:similarTo_directly"/>
  <ctx:context rdf:resource="#context3"/>
  <fuz:fuzzyRelationDomain rdf:resource="#H65_0"/>
  <fuz:fuzzyRelationRange rdf:resource="#H66_9"/>
  <fuz:membershipDegree rdf:datatype="&xsd:float">0.1
</fuz:membershipDegree>
</ctx:ContextFuzzyRelationMembership>
```

Fig. 6. Representation of context similarity in ICD-10.

We employed a new algorithm called Context-NARFO (Non-redundant and generalized Association Rules based on Fuzzy Ontologies Considering Context) to mine association rules regarding the MedFuzOnto Ontology. The Context-NARFO algorithm extends and enhances the NARFO algorithm [6]. Context-NARFO takes the advantage of dynamically considering the context defined by experts in the mining process. It allows a dynamic data analysis according to the user required background in a given time, mining association rules considering the context requested by the specialist. The Context-NARFO adds the steps of context checking and similar items identification regarding context in the previous algorithm NARFO. To mine association rules using Context-NARFO, the expert should choose the context to analyze the data.

### B. Carrying out the Experiments

Our experiments was carried out employing the parameters values: minimum support ( $minsup$ )=0.2, minimum confidence ( $minconf$ )=0.2, and the minimum similarity ( $minsim$ ) varies from 0.2 to 0.8, in steps of 0.2 (in order to show the different results for each context). This study employed the MedFuzOnto with five scenarios as shown in Fig. 5. The Jena Framework [5] was used to allow the navigation through the ontology concepts and relations, making Context-NARFO able to obtain similar items and similar degree values for each context.

To illustrate the results obtained using the UFOCoRe Ontology proposed here, we executed the Context-NARFO algorithm using the MedFuzOnto ontology created according to the Upper Ontology UFOCoRe. Fig. 7 illustrates examples

of rules generated for the Context 1 (Group Similarity) and Context 5 (Medical Specialty), using the minimum similarity ( $minsim$ )=0.6.

```
Context-1 (minsim)=0.6
ICD Similarity -> [[M751, M542, 0.8], [N859, M779, 0.6], [J303, I119, 0.7],
[H669, H650, 0.9], [M545, L811, 0.6], [Z340, Z010, 0.8], [M255, K590, 0.6]]
Pass #1: 59 Candidates
Pass #2: 91 Candidates
Pass #3: 24 Candidates
Pass #4: 1 Candidates
Itemset Candidates: 175
Rules: 132
```

```
Rule 60: J303~I119->MEDICAL sup=0.13103104 conf=0.9746836
Rule 61: MEDICAL->J303~I119 sup=0.13103104 conf=0.14609376
Rule 62: SEX->MEDICAL sup=0.8968969 conf=0.8968969
Rule 63: MEDICAL->SEX sup=0.8968969 conf=1.0
Rule 64: MALE, H669~H650->MEDICAL sup=0.19114114 conf=0.862661
```

```
Context-5 (minsim)=0.6
ICD Similarity -> [[M751, M542, 0.9], [H669, H650, 0.6]]
Pass #1: 54 Candidates
Pass #2: 78 Candidates
Pass #3: 19 Candidates
Pass #4: 1 Candidates
Itemset Candidates: 152
Rules: 120
Rule 29: PEDIATRICES->MEDICAL_0 - 18 years sup=0.17417417 conf=0.9157894
Rule 30: H669~H650->SEX sup=0.35395396 conf=1.0
Rule 31: SEX->H669~H650 sup=0.35395396 conf=0.35395396
```

Fig. 7. Rules generated by Context-NARFO.

In the example showed in Fig. 7, the Rule 61 MEDICAL->J303~I119 generated for Context 1 (Group Similarity), means that the MEDICAL providers that are associate to obstetrics and gynecology services (ICD J303) also are associated to neurosurgery services (ICD I119). The symbol "~" represents that there is a satisfactory similarity between J303 and I119 according to the Context 1. This means that in all rules where one of these items, the other will also be considered. The values of similarity are given after the pair of items. For instance, the notation [M751, M542, 0.9] means that the CIDs M751 and M542 have similarity 0.9 in the context being analyzed (in the Context 5).

It is possible to observe in Fig. 7 that all the pairs of ICDs with similarity degree less than 0.6 were eliminated from the candidates' generation. For example, in Fig. 7, if the Context 1 is focused, we can observe that just the pair (Z001~J304=0.1) was not considered in the candidate generation process. Another example that shows the differences between the Contexts 1 and 5, is the similarity J303~I119=0.7 illustrated by the underlined line in Fig. 7, this similarity was not used to find candidate itemsets in the Context 5, because the similarity degree in this context is much smaller 0.1. Thus, the simple selection of context may cause a considerable change in the mining process. Such change can be noted by the different number of candidate itemsets that are generated. Fig. 8 shows the graphs of the number of candidate itemsets (itemsets that satisfy the of minimum support threshold) generated by each context (columns). Each graph illustrated in different color represents the number of candidate itemsets generated by each context (columns). The data from Fig. 8 indicates that Context-NARFO generates different quantities of candidate itemsets for each context selected by the user.

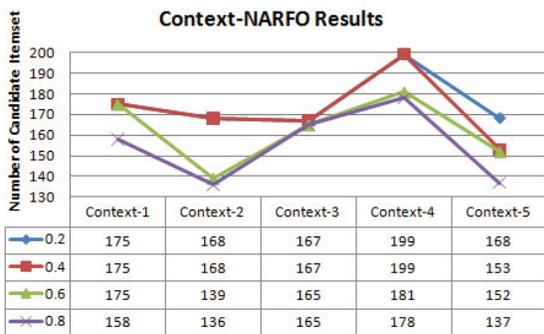


Fig. 8. Number of candidates for each context.

Observe that the Context-4 (Cost) leads to the largest number of candidates (see Fig. 8), while the Context-2 (Ancillary Service of Diagnosis and Therapy), leads to the smallest ones. This may occur because the diseases associated with the CIDs employed in the experiment are more similar regarding the cost values than regarding the service required for diagnosis and treatment. The values of the minimum similarity (*minsim*) that leads to the higher numbers of candidates are the less ones (0.2 and 0.4). This is expected because, the smaller the value of *minsim*, the higher the number of itemsets considered similar employed to increase the support that are not pruned by the algorithm.

We compared the NARFO and Context-NARFO algorithm by employing both in the same medical database, considering only the Context 2 (Ancillary Service of Diagnosis and Therapy). The number of candidates generated in the mining phase of NARFO for a context is the equal to the generated by Context-NARFO, considering only the same context. The number of candidates generated by NARFO is the same of Context-NARFO for the Context 2. However, Context-NARFO allows the analysis of multiple contexts, generating a different numbers of candidates according to the context employed. To change the context without using the proposed approach UFOCoRe, the specialist should run the NARFO algorithm loading another ontology with the similarity degrees regarding the new desired context. Using our approach this rework is not need. Therefore, the whole analysis process becomes less tiresome and more productive using the proposed approach. Observe that if the Context-2 is changed to Context-4, the number of candidates increase in 24% (similarity degree 0.8 - see Fig. 8). This increase can represent a considerable semantic gain in the data mining process, making it possible to find more representative and interesting patterns. The representation of the degree of similarity in the process of knowledge discovery generates fuzzy association rules semantically more relevant. A limitation of the algorithm NARFO is that it limits the pursuit of knowledge by the specialist because he/she will be restrict to employ the analysis of the dataset employing only a single view (context). This limitation is overcome with the Context-NARFO algorithm that uses the MedFuzOnto ontology created according to the upper ontology UFOCoRe proposed here, making more flexible the data analysis performed by the mining algorithm.

## V. CONCLUSIONS

This paper presented a new Upper Fuzzy Ontology With Context Representation (UFOCoRe). To do so, we represent multiple similarities between instances without requiring the building of another ontology, nor adapting an existing one, adding classes or relationships. The proposed method allows the user to make dynamical analysis of different contexts, using a single ontology. All development of upper ontology was performed using the language RDF / XML. This allows that any ontology, which meets the language standard, can be used with the proposed representation of context. Moreover, in this paper we presented the application of the proposed meta-ontology in a real dataset. The results showed the effective and applicability of the proposed meta-ontology in the mining of association rules, producing rich semantic rules regarding multiple contexts without the expensiveness of building multiple ontologies. Furthermore, the UFOCore can be applied in other domains besides medical domain. Future works includes adding relevance feedback technique to the method proposed.

## ACKNOWLEDGMENTS

We thank Unimed Araras and the Brazilian funding agencies FAPESP, CAPES and CNPq.

## REFERENCES

- [1]Chen, G., Wei, Q., Kerre, E. E. (2000). Fuzzy Data Mining: Discovery of Fuzzy Generalized Association Rules. Recent Issues on Fuzzy Databases. Wurzberg: Physica-Verlag. pp. 45–66.
- [2]Farzanyar, Z., Kangavari, M., Hashemi, S. (2006). A New Algorithm for Mining Fuzzy Association Rules in the Large Databases Based on Ontology. In IEEE Int Conf on Data Mining – Wsh. Hong Kong, China. pp. 18-22.
- [3]Srikant, R., Agrawal, R., (1995). Mining Generalized Association Rules. In Int Conf of VLDB. Zurich, Switzerland. pp. 11-15.
- [4]Hou, X., Gu, J., Shen, X., Yan, W. (2005). Application of Data Mining in Fault Diagnosis Based on Ontology. In Int Conf on Information Technology and Applications. Sydney, Australia. pp. 4-7.
- [5]Carrol, J. J., Dickinson, I., Dollin, C., Reynolds, D., Seaborn, A., Wilkinson, K. (2004). Jena: implementing the semantic web recommendations. In Int WWW Conf New York, USA. pp. 19-21.
- [6]Miani, R. G. et al (2009). NARFO Algorithm: Mining Non-redundant and Generalized Association Rules Based on Fuzzy Ontologies. In: ICEIS, Milan, Italy. pp. 415–426.
- [7]Agrawal, R; Imielinski, T; Swami, A. M. (1993). Mining Association Rules between Sets of Items in Large Databases. In: ACM SIGMOD Conference, Washington, USA. New York: ACM, pp. 207-216.
- [8]ICD-10, International Statistic Classification of Diseases and Related Health Problems. Available in: <https://dkm.fbk.eu/index.php/ICD-10\_Ontology>. Last access: Jan. 2010.
- [9]Zadeh, L. (1987a). Fuzzy Sets. In Fuzzy sets and applications: Selected Papers by L.A. Zadeh, pp. 29–44. Wiley-Interscience, New York, USA.
- [10]Zadeh, L. (1987b). Similarity Relations and Fuzzy Orderings. In Fuzzy Sets and Applications: Select Papers by L. A. Zadeh, pp. 81–104. Wiley-Interscience, New York, USA.
- [11]Smith, M. K., Welty, C., and McGuinness, D. L. (2004). W3C Proposed Recommendation: OWL Web Ontology Language Guide. Available in: <http://www.w3.org/TR/2004/REC-owl-guide-20040210>. Last access: Feb. 2010.
- [12]Baldauf, M., Dustdar, S., Rosenberg, F. (2007). A survey on context-aware systems. Int Journal of Ad Hoc and Ubiquitous Computing, vol. 2, 4th ed., pp. 263-277.
- [13]Yaguinuma, C., Santos, M., Biajiz, M. (2007). Meta-ontologia Difusa para representação de Informações Imprecisas em ontologias. II WOMSDE, pp. 57-67. (In Portuguese).

# A String Constraint Solver for Detecting Web Application Vulnerability

Xiang Fu  
Hofstra University  
cscxzf@hofstra.edu

Chung-Chih Li  
Illinois State University  
cli2@ilstu.edu

## Abstract

Given the bytecode of a software system, is it possible to automatically generate attack signatures that reveal its vulnerabilities? A natural solution would be symbolically executing the target system and constructing constraints for matching path conditions and attack patterns. Clearly, the constraint solving technique is the key to the above research. This paper presents Simple Linear String Equation (SISE), a formalism for specifying constraints on strings. SISE uses finite state transducers to precisely model various regular replacement operations, which makes it applicable for analyzing text processing programs such as web applications. We present a recursive algorithm that computes the solution pool of a SISE. Given the solution pool, a concrete variable solution can be generated. The algorithm is implemented in a Java constraint solver called SUSHI, which is applied to security analysis of web applications.

## 1 Introduction

Defects in user input validation are usually the cause of the ever increasing attacks on web applications and other software systems. In practice, it is interesting to automatically discover these defects and show to software designers, step by step, how the security holes lead to attacks.

In our previous work [6], we proposed a unified symbolic execution framework (as shown in Figure 1) for tackling the above challenge. Given the bytecode of a target system (e.g., a web application) and a collection of attack patterns, the framework starts with instrumenting the bytecode to prepare the system for symbolic execution [14]. Then the target system is executed as usual except that program inputs are treated as symbolic literals. Path conditions, constraints built upon symbolic literals, are used to trace the execution. A path condition records the conditions to be met by the initial values of the program input, so that the program will execute to a location. At critical points, e.g., where a SQL query is submitted, path conditions are paired with attack patterns. Solving these equations leads to attack

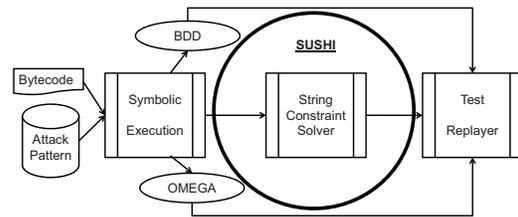


Figure 1. Unified Symbolic Framework for Vulnerability Detection

signatures and error traces that reveal vulnerabilities.

This paper presents the core algorithms of SUSHI, a constraint solver, which is the key to the aforementioned research. We study a formalism called Simple Linear String Equation (SISE), for representing path conditions and attack patterns. We present a recursive algorithm which can solve SISE constraints effectively. Intuitively, a SISE equation can be regarded as a variation of the word equation problem [17]. It is composed of word literals, string variables, and various frequently seen string operators such as substring, concatenation, and regular substitution. To solve SISE, an automata based approach is taken, where a SISE constraint is broken down into a number of *atomic* string operations. Then the solution process consists of a number of *backward image computation* steps. We show that SISE can be used to discover deeply hidden SQL injection and Cross-Site Scripting vulnerabilities. They are “corner cases” of the insufficient user input sanitation procedures, which are often neglected by security inspection tools.

The rest of the paper is organized as follows. §2 formalizes the notion of string equation system. §3 briefly describes modeling of various regular replacement semantics using finite state transducer. §4 presents a recursive algorithm for computing the solution pool of a SISE equation and analyzes the complexity of the algorithm. §5 introduces tool support. §6 presents several case study examples of SUSHI. §7 discusses related work, and §8 concludes.

## 2 String Equation System

This section formalizes the notion of SISE. Let  $N$  denote the set of natural numbers and  $\Sigma$  a finite alphabet. If  $\omega \in \Sigma^*$ , we say that  $\omega$  is a word. We use  $\epsilon$  to represent an empty word. Let  $R$  be the set of regular expressions over  $\Sigma$ . If  $r \in R$ , let  $L(r)$  be the language represented by  $r$ .

### 2.1 String Operators

The string equation framework supports a set of five essential operators for static program analysis. The set of operators is denoted using  $O = \{\circ, [i, j], x_{r \rightarrow \omega}, x_{r \rightarrow \omega}^-, x_{r \rightarrow \omega}^+\}$ . They are concatenation ( $\circ$ ), substring ( $[i, j]$ ), declarative regular replacement ( $x_{r \rightarrow \omega}$ ), reluctant regular replacement ( $x_{r \rightarrow \omega}^-$ ), and greedy regular replacement ( $x_{r \rightarrow \omega}^+$ ).

Regular replacement has several different semantics. Let  $s, \omega \in \Sigma^*$  and  $r \in R$ . The declarative replacement, i.e.,  $s_{r \rightarrow \omega}$ , denotes the set of all possible strings that can be obtained from  $s$  by substituting  $\omega$  for every occurrence of a substring that matches  $L(r)$ . Notice that one declarative replacement might result in multiple words as results. The procedural replacement, i.e., the greedy and reluctant, will be different. Both of them use *left-most* matching. The greedy replacement  $s_{r \rightarrow \omega}^+$  tries to match the longest string and the reluctant tries to match the shortest. The following example demonstrates their difference.

**Example 2.1** Consider the following two cases. (i) If  $s = aaab, r = (aa|ab)$ , and  $\omega = c$ , then  $s_{r \rightarrow \omega} = \{cc, acb\}$ , and  $s_{r \rightarrow \omega}^- = s_{r \rightarrow \omega}^+ = cc$ . (ii) If  $s = aaa, r = a^+$ , and  $\omega = b$ , then  $s_{r \rightarrow \omega} = \{b, bb, bbb\}$ ,  $s_{r \rightarrow \omega}^- = bbb$ , and  $s_{r \rightarrow \omega}^+ = b$ .  $\square$

We assume that there are infinitely many distinguishable string variables and let this set of variables be denoted by  $V$ . Intuitively, a *string expression* is a regular expression over  $\Sigma$  with occurrences of variables in  $V$  and operators in  $O$  (such as  $\circ, [i, j]$ , and  $x_{r \rightarrow \omega}$ ). A string equation is composed of two string expressions.

**Definition 2.2** Let  $E$  be the set of string expressions. A string equation is denoted by  $\mu \equiv \nu$  with  $\mu, \nu \in E$ . A string equation system is a conjunction of a finite set of string equations.  $\square$

### 2.2 Simple Linear String Equation

The purpose of SISE is to capture the constraints that arise from a symbolic execution. This naturally leads to a *restricted* form of string equation, i.e., all string variables appear only on the LHS (left hand side), and the RHS (right hand side) is a regular expression.

**Definition 2.3** A Simple Linear String Equation (SISE)  $\mu \equiv r$  is a string equation such that  $\mu \in E, r \in R$  provided that every string variable occurs at most once in  $\mu$ .  $\square$

**Definition 2.4** A solution to a SISE equation is a mapping  $\rho : V \rightarrow R$  which makes the LHS equivalent to RHS (as regular expression). Let  $\mu \equiv r$  be a SISE and suppose string variable  $v$  occurs in  $\mu$ . The *solution pool* for  $v$ , denoted by  $sp(v)$ , is defined as  $sp(v) = \{\omega \mid \omega \in r_2 \text{ and } \rho(v) = r_2 \text{ where } \rho \text{ is a solution to } \mu \equiv r\}$   $\square$

It is shown later that  $sp(v)$  is a regular language for any SISE. In §4, we will describe an algorithm that takes a SISE as input and constructs as output regular expressions that represent the solution pools for all string variables in the equation.

## 3 Modeling Regular Replacement

Regular replacement is widely used by web application designers for sanitizing user input. This section introduces a finite state transducer model for handling various string replacement semantics. It is necessary because ignoring the difference of regular replacement semantics leads to imprecise analysis.

Consider a PHP snippet called “postMessage” in Listing 1. The servlet takes a message from an anonymous user and posts it on a bulletin. To prevent Cross-Site Scripting attack, the programmer calls `preg_replace()` to remove any pair of `<script>` and `</script>` tags and the contents between them. Notice that the wild card operator `*?` is a *reluctant* operator, i.e., it matches the shortest string possible. For example, given word `<script>a</script></script>`, the call on line 3 returns `</script>`. If `*` (the greedy operator) is used, the call on line 3 returns an empty string.

```

1 <?php
2     $msg = $_POST["msg"];
3     $sanitized = preg_replace(
4         "/\<script.*?\>.*?\</script.*?\>/i",
5         "", $a);
6     save_to_db($sanitized)
7 ?>

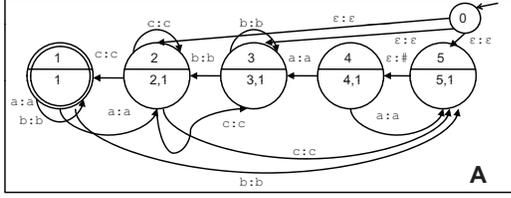
```

**Listing 1. Vulnerable XSS Sanitation**

SUSHI is able to generate the following attack signature. Readers can verify that it is indeed effective.

```
<<script></script>script>alert('a')</script>
```

Now, a natural question following the above analysis is: *If we approximate the reluctant semantics using the greedy semantics, could the static analysis be still effective?* The



**Figure 2. FST for Inserting Begin Markers**

answer is clearly negative: When the  $\star?$  operators in Listing 1 are treated as  $\star$ , SUSHI reports no solution for the aforementioned SISE equation, i.e., a false negative report on the actually vulnerable sanitation. Thus, a precise modeling of the various regular replacement semantics is necessary.

Finite state transducer was used for processing phonological rules in [12]. We found it useful for modeling string replacements. Due to the space limit, in this section we summarize our findings and more technical details can be found in [5].

**Definition 3.1** A finite state transducer (FST) is a quintuple  $(\Sigma, Q, q_0, F, \delta)$  where  $\Sigma$  is the alphabet,  $Q$  is the set of states,  $q_0 \in Q$  is the initial state,  $F \subseteq Q$  is the set of final states, and  $\delta$  is the transition function, and  $\delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Sigma \cup \{\epsilon\}) \times Q$ .  $\square$

Note that an FST accepts a regular relation (a subset of  $\Sigma^* \times \Sigma^*$ ), just like a Finite State Automaton (FSA) accepts a regular language. Given two FST's  $M_1$  and  $M_2$ , let  $M_1 || M_2$  denote an FST such that,  $L(M_1 || M_2) = \{(s, \omega) \mid (s, \eta) \in L(M_1) \text{ and } (\eta, \omega) \in L(M_2) \text{ for some } \eta\}$ . Intuitively,  $M_1 || M_2$  pipes the contents of the second tape of  $M_1$  into the first tape of  $M_2$ , and simulates  $M_1$  and  $M_2$  in parallel.

The key to modeling the greedy and reluctant semantics is to capture the left-most matching. Let  $\# \notin \Sigma$  be a special “begin marker”, and  $\$ \notin \Sigma$  be the special “end marker”.

A begin marker insertion performer,  $\mathcal{A}$  (an FST on alphabet  $\Sigma \cup \{\#\}$ ), can be constructed for marking the beginning of pattern  $r \in R$  in any words  $s \in \Sigma^*$ . For example, the FST  $\mathcal{A}$  in Figure 2 marks the beginning of regular pattern  $a^+b^+c$ . For instance,  $(aabbcc, \#a\#abbcc) \in L(\mathcal{A})$ .

With begin marker insertion performer  $\mathcal{A}$ , the reluctant semantics can be modeled by piping  $\mathcal{A}$  with another transducer  $\mathcal{A}_2$ , which, whenever sees a begin marker  $\#$ , enters the state of conducting the replacement, and whenever a regular pattern  $r$  is matched, enters immediately the status waiting for  $\#$ .  $\mathcal{A} || \mathcal{A}_2$  allows to precisely model the pure reluctant semantics. The modeling of greedy semantics relies more on the power of nondeterminism and needs the composition of several more transducers. First, an end marker insertion performer nondeterministically inserts an end marker after each match of the pattern  $r$  in the input

word. Then a number of filter transducers are composed to identify the “longest” match and remove extra markers. The construction details can be found in [5].

The following lemma summarizes our finding: for each of the regular replacement operators, there exists a corresponding FST. Let them be  $\mathcal{M}_{r \rightarrow \omega}$ ,  $\mathcal{M}_{r \rightarrow \omega}^-$ , and  $\mathcal{M}_{r \rightarrow \omega}^+$ .

**Lemma 3.2** For any  $r \in R$  and  $\omega \in \Sigma^*$  the following three finite state transducers  $\mathcal{M}_{r \rightarrow \omega}$ ,  $\mathcal{M}_{r \rightarrow \omega}^-$ ,  $\mathcal{M}_{r \rightarrow \omega}^+$  can be constructed s.t. for any  $s, \eta \in \Sigma^*$  (i)  $(s, \eta) \in L(\mathcal{M}_{r \rightarrow \omega})$  iff  $\eta \in s_{r \rightarrow \omega}$ ; and (ii)  $(s, \eta) \in L(\mathcal{M}_{r \rightarrow \omega}^-)$  iff  $\eta = s_{r \rightarrow \omega}^-$ ; and (iii)  $(s, \eta) \in L(\mathcal{M}_{r \rightarrow \omega}^+)$  iff  $\eta = s_{r \rightarrow \omega}^+$ .

## 4 Constraint Solving

In this section we introduce the recursive algorithm that solves SISE constraints. The algorithm is able to find all solutions of a SISE constraint.

Notice that we distinguish the concept of “solution pool” and “concrete solution”. Intuitively, for each variable, its solution pool is the set of all possible values it could take in some concrete solution. In our algorithm for solving SISE, the solution pool is computed first. Then concrete solutions are derived from the solution pool. The use of solution pool allows some interesting directions in our future work, e.g., it can be used for estimating the size of solution space, thus permitting the use of random testing for security analysis.

### 4.1 Atomic Cases

Solving a SISE can be reduced to solving four basic cases of string operators. The atomic case is trivial. That is, for SISE  $E \equiv r$  if  $E = x$  and  $x \in V$ , then the solution pool of  $x$  is simply  $L(r)$ . We next consider the other three cases.

**Substring case:**  $\mu[i, j] \equiv r$  where  $\mu \in E$  and  $i, j \in N$  with  $i \leq j$ . The following equivalence is straightforward by which we can remove a substring operator.

**Equivalence 1** For any SISE of the form  $\mu[i, j] \equiv r$  where  $\mu \in E$  and  $i, j \in N$  with  $i \leq j$ ,  $\rho$  is a solution to  $\mu[i, j] \equiv r$  iff it is a solution to  $\mu \equiv \Sigma^i r [0, j - i] \Sigma^*$ .

**Concatenation case:**  $\mu\nu \equiv r$  where  $\mu, \nu \in E$ .

The solution is obvious when  $\nu \in R$ . Consider  $xr_1 \equiv r_2$  where  $x \in V$  and  $r_1, r_2 \in R$ . This can be easily solved using *right quotient* of regular expression [10]. By convention, the right quotient  $r_2/r_1 = \{x \mid xw \in r_2 \text{ and } w \in r_1\}$ . Similarly, the *left quotient* is defined as  $r_2 \setminus r_1 = \{x \mid wx \in r_2 \text{ and } w \in r_1\}$ . We know that if  $r_1$  and  $r_2$  are regular, so are  $r_2/r_1$  and  $r_2 \setminus r_1$ . The algorithm for computing regular quotient is standard.

**Equivalence 2** For any SISE of the form  $\mu r_1 \equiv r_2$  ( $r_1 \mu \equiv r_2$ ) where  $\mu \in E$  and  $r_1, r_2 \in R$ ,  $\rho$  is a solution to  $\mu r_1 \equiv r_2$  ( $r_1 \mu \equiv r_2$ ) iff  $\rho$  is a solution to  $\mu \equiv r_2/r_1$  ( $\mu \equiv r_2 \setminus r_1$ ).

Now consider the general case  $\mu\nu \equiv r$  where both  $\mu$  and  $\nu$  are non-trivial string expressions. Let  $approx(\nu)$  be the result of replacing every variable in  $\nu$  with  $\Sigma^*$ . Clearly,  $approx(\nu)$  is a regular expression. We then have the following.

**Equivalence 3** For any SISE of the form  $\mu\nu \equiv r_2$ ,  $\rho$  is a solution to  $\mu\nu \equiv r_2$  iff  $\rho$  is a solution to  $\mu \circ approx(\nu) \equiv r_2$ .

The solution to  $\nu$  can be computed similarly. The proof can be based on the fact that a string variable cannot appear in both  $\mu$  and  $\nu$ .

**Replacement case:**  $\mu_{r_1 \rightarrow \omega} \equiv r_2$  ( $\mu_{r_1^- \rightarrow \omega} \equiv r_2$ ,  $\mu_{r_1^+ \rightarrow \omega} \equiv r_2$ ) where  $\mu \in E$ ,  $r_1, r_2 \in R$ , and  $\omega \in \Sigma^*$ .

In the following we discuss the solution to  $\mu_{r_1 \rightarrow \omega} \equiv r_2$ . The solution to procedural replacements will be similar, because all of them use finite state transducer algorithms.

Clearly, a possible solution to  $x_{r_1 \rightarrow \omega} = r_2$  is a word  $s$  such that  $s_{r_1 \rightarrow \omega} \subseteq L(r_2)$ . Thus,  $sp(x) = \{s \mid s_{r_1 \rightarrow \omega} \subseteq L(r_2)\}$ . Our goal is to construct an FST that accepts only  $(s, \eta)$  such that  $\eta \in L(r_2)$  and  $\eta$  is obtained from  $s$  by replacing every occurrence of patterns in  $r_1$  with  $\omega$ . In other words, we want an FST, denoted by  $\mathcal{M}_{r_1 \rightarrow \omega \Rightarrow r_2}$  s.t.

$$(s, \eta) \in L(\mathcal{M}_{r_1 \rightarrow \omega \Rightarrow r_2}) \Leftrightarrow \eta \in L(r_2) \text{ and } \eta \in s_{r_1 \rightarrow \omega}$$

We now construct  $\mathcal{M}_{r_1 \rightarrow \omega \Rightarrow r_2}$ . Let  $\mathcal{M}_1$  be the FST that accepts the identity relation  $\{(s, s) \mid s \in L(r_2)\}$ . Let  $\mathcal{M}_{r_1 \rightarrow \omega}$  be the FST defined in Lemma 3.2, i.e.,  $(s, \eta) \in L(\mathcal{M}_{r_1 \rightarrow \omega})$  iff  $\eta \in s_{r_1 \rightarrow \omega}$ .  $\mathcal{M}_{r_1 \rightarrow \omega \Rightarrow r_2}$  can then be constructed as  $\mathcal{M}_{r_1 \rightarrow \omega} \parallel \mathcal{M}_1$ . Similarly,  $\mathcal{M}_{r_1^- \rightarrow \omega \Rightarrow r_2}$  and  $\mathcal{M}_{r_1^+ \rightarrow \omega \Rightarrow r_2}$  can be constructed for the pure reluctant and greedy semantics, respectively.

**Equivalence 4** For any SISE of the form  $\mu_{r_1 \rightarrow \omega} \equiv r_2$  where  $\mu \in E$ ,  $r_1, r_2 \in R$ , and  $\omega \in \Sigma^*$ .  $\rho$  is a solution to  $\mu_{r_1 \rightarrow \omega} \equiv r_2$  iff it is a solution to  $\mu \equiv r$  where  $L(r) = \{s \mid (s, \eta) \in L(\mathcal{M}_{r_1 \rightarrow \omega \Rightarrow r_2})\}$ .

Clearly,  $L(r)$  can be easily computed by projecting the FST  $\mathcal{M}_{r_1 \rightarrow x \Rightarrow r_2}$  to its input tape, which results in a finite state machine, representing a regular language. The same applies to the pure greedy and reluctant semantics, using  $\mathcal{M}_{r_1^+ \rightarrow \omega \Rightarrow r_2}$  and  $\mathcal{M}_{r_1^- \rightarrow \omega \Rightarrow r_2}$ . Consequently, we have the following.

**Theorem 4.1** Given SISE  $\mu \equiv r$ , for any variable  $v$  in  $\mu$ , its solution pool  $sp(v)$  is a regular language.

```

function solve( $\mu \equiv r$ )
switch ( $\mu$ ):
case  $x \in V$ : return  $\{(x, r)\}$ 
case  $r_1 \in R$ : if  $L(r_1) \cap L(r) \neq \emptyset$  return  $\emptyset$  o.t. return  $\perp$ 
case  $\mu[i, j]$ : return solve( $\mu \equiv \Sigma^i r[0, j - i] \Sigma^*$ )
case  $\mu_{r_1 \rightarrow \omega}$ : return solve( $\mu \equiv \{s \mid (s, \eta) \in L(\mathcal{M}_{r_1 \rightarrow \omega \Rightarrow r})\}$ )
case  $\mu_{r_1^+ \rightarrow \omega}$ : return solve( $\mu \equiv \{s \mid (s, \eta) \in L(\mathcal{M}_{r_1^+ \rightarrow \omega \Rightarrow r})\}$ )
case  $\mu_{r_1^- \rightarrow \omega}$ : return solve( $\mu \equiv \{s \mid (s, \eta) \in L(\mathcal{M}_{r_1^- \rightarrow \omega \Rightarrow r})\}$ )
case  $\mu\nu$ :
  Let  $r_1$  be  $approx(\mu)$  and  $r_2$  be  $approx(\nu)$ 
  return solve( $\mu \equiv r/r_2$ )  $\cup$  solve( $\nu \equiv r \setminus r_1$ )

```

Figure 3. Computing Solution Pool

## 4.2 Recursive Algorithm

Based on Equivalences 1 to 4, and Theorem 4.1, we can develop a recursive algorithm for generating the solution pool for all variables in a SISE. The algorithm is shown in Figure 3. Function `solve()` returns a set of tuples with each tuple representing a solution pool (note: not solution) for a variable. We use  $\perp$  to represent “no solution”. When applying any set operation (e.g., intersection and union) on  $\perp$ , the result is  $\perp$ . Note  $\perp$  is not the same as empty set  $\emptyset$ .

**Theorem 4.2** The worst complexity of the algorithm in Figure 3 is  $O(|\mu| \times 2^{6 \times 2^{|\mu|} + |r|})$ .

The complexity analysis needs the detailed transducer composition algorithm in [5]. We briefly discuss the sketch here. In Figure 3 the most expensive computation is the case  $\mu_{r_1^+ \rightarrow \omega}$ , which needs  $\mathcal{M}_{r_1^+ \rightarrow \omega}$ , a composition of six transducers. Among them, the size of the largest (i.e., the number of transitions plus states) is  $2^{2^{|r_1|}}$  where  $r_1$  is the regular pattern to search. Since  $|r_1| < |\mu|$ , we can approximate the worst complexity of handling the  $\mu_{r_1^+ \rightarrow \omega}$  case as  $O(2^{6 \times 2^{|\mu|} + |r|})$ , because we need an additional composition operation on  $\mathcal{M}_{r_1^+ \rightarrow \omega}$  with the identity relation of  $r$ . Finally, the upper bound of the recursion depth of the algorithm in Figure 3 is  $|\mu|$ , because there are at most  $|\mu|$  operators in the LHS. This eventually leads to the complexity in Theorem 4.2.

Given the solution pool, a concrete solution can be generated by concretizing the valuation of a variable one by one using a counter loop on the number of variables in the equation. In each iteration, nondeterministically instantiate one variable from a value contained in its solution pool. Thus a new SISE equation is obtained. Solving this equation would lead to the solution pool to be used in the next iteration. Starting from the initial solution pool, the concretization process will always terminate with a concrete solution generated.

## 5 SUSHI Constraint Solver

SISE constraint solving is implemented in a Java library called SUSHI. This section presents some details

of the tool implementation and discusses SUSHI's efficiency as a constraint solver. SUSHI includes a self-made package for supporting FST operations, and it relies on `dk.bricks.automaton` package [18] for manipulating FSA. In practice, to perform inspection on user input, FST has to handle a large alphabet represented using 16-bit Unicode. This is handled by a compression approach called SUSHI FST Transition Set. Special algorithms are developed for finite state transducer operations on SFTS. Details are not presented here due to space limit.

### 5.1 Evaluation of Efficiency

We are interested in the performance of SUSHI as a constraint solver. In Figure 4 we list five SISE equations for stress-testing the SUSHI package and the running statistics. Note that each equation is parametrized by an integer  $n$  (ranging from 1 to 50). For example, when  $n$  is 50, the size of the RHS of eq4 is 100.

ID	Equation
eq1	$x \circ a\{n, n\} \equiv (a b)\{2n, 2n\}$
eq2	$x[n, 2n] \equiv a\{n, n\}$
eq3	$x \circ a \circ y[0, n] \equiv b\{n, n\}ab\{n, n\}$
eq4	$x_{a^+ \rightarrow b\{n, n\}}^+ \equiv b\{2n, 2n\}$
eq5	$uname=' \circ x_{i-,,}[0, n] \circ ' \text{pwd}=' \equiv uname=' [^' ']*'$

Figure 4. Sample Benchmark Equations

Clearly, the sample set covers all string operations we discussed earlier. In Figure 5, the first two diagrams present the size of the FST (the number of states and the number of transitions) used in the solution process, the third diagram is the size of the FSA used for representing solution pools, and the last shows the time spent on running each test. As shown in Figure 5, SUSHI scales well in most cases. The figure also suggests that the solution cost of a SISE equation mainly depends on the complexity of the automata structure of the resulting solution pool (e.g., readers can compare the cost of eq4 and eq5). In addition, the experimental data (see Figures 5(a) and 5(b)) clearly indicate that to model greedy regular replacement (e.g., eq4) is expensive because the modeling process involves composition of six transducers.

## 6 Application Examples

This section presents two application examples of how SUSHI constraint solver can be used for discovering attack signatures of web application vulnerabilities.

Notice that the in-depth discussion of symbolic execution is out of the scope of this paper. Without the loss of generality, we assume that SISE constraints can be constructed by standard symbolic execution technique. In an

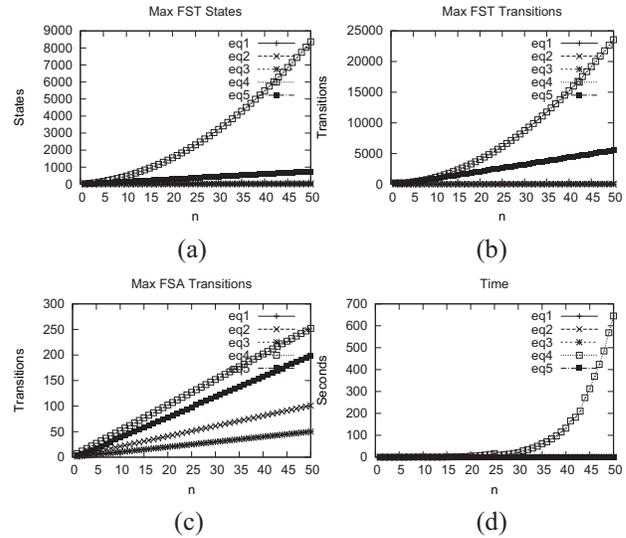


Figure 5. Constraint Solving Cost

other word, the application of SISE and its solver SUSHI library is not restricted to a certain programming language. In fact, for both of the application examples below, the SISE constraints are constructed manually. The second stage of the security analysis, i.e., attack signature generation, is automated using SUSHI. The whole process could be automated if its integration with symbolic execution engine is ready.

### 6.1 Discovering Password Bypassing

Consider a Java servlet called `BadLogin`, adapted from the example introduced in [6]. As shown in Listing 2, the user authentication is performed by a function `processRequest`, which reads the user input (a user name and a password), and verifies their existence in the back-end database.

To avoid attacks like SQL injection, a message function is defined for applying a number of sanitation procedures. For each user input, it replaces the single quote character (a special control character in SQL) with its escaping form (a sequence of two single quotes). In addition, it limits the size of each user input string to 16 characters. Note that here 16 can be any positive integer.

The length restriction protection intends to limit the room of attackers for playing tricks. *Good intention, however, may not eventually lead to desired effects!* Combined with the string substitution, it actually causes a delicate vulnerability and the following is the *shortest* attack signature discovered by SUSHI. Notice that the pair of strings are shorter than the one originally given in [6], because the constraint solving algorithm in this paper is able to find all possible solutions.

```
sUsername = a''''''''
sPwd = '' OR uname<>'
```

```

1 protected void processRequest(
2   HttpServletRequest request ...)
3   throws ServletException {
4     PrintWriter out = response.getWriter();
5     try {
6       String sUname = request.getParameter("sUname");
7       String sPwd = request.getParameter("sPwd");
8       Connection con = DriverManager.getConnection("...");
9       Statement stmt = con.createStatement();
10      String strCmd= "SELECT * FROM users \nWHERE username="
11        + message(sUname) + "' AND pwd="
12        + message(sPwd) + "'";
13      ResultSet srs = stmt.executeQuery(strCmd);
14      if (srs.next()){
15        out.println("Welcome_" + sUname);
16      } else {
17        out.println("Login_fail!");
18      }
19    } catch (Exception exc) {...}
20  }
21
22  protected String message(String str){
23    String strOut = str.replaceAll("'", "'");
24    if (strOut.length() > 16) return strOut.substring(0, 16);
25    return strOut;
26  }

```

**Listing 2. Vulnerable Authentication**

The trick is explained briefly as below. By applying the `message()` function on `sUname`, each of the 8 single quotes is converted to a sequence of two quotes. However, the last quote is chopped off by the `substring()` function (at the 16'th character as shown in Listing 2). This results in the following SQL query, which bypasses password checking. Notice that the logical structure of the `WHERE` clause has been changed by the attack string, because the single quotes are treated differently – each pair of single quotes between a and `AND` is regarded as the escaping form of one single quote by SQL parser.

```

SELECT uname, pwd FROM users
WHERE uname='a'''''''''''''''''''' AND pwd=''''' OR uname<>'

```

Clearly, the vulnerability discussed in this section cannot be discovered by black-box testing like [4, 11, 20], because without prior knowledge of the implementation, it is very hard to craft the input strings that could pass the sanity check by `message()`. The string analyses such as [1, 3, 13] cannot discover the bug either, because regular replacement is not supported. Only when the control/data flow information is taken advantage by the automated vulnerability scanner, such deeply hidden bugs can be revealed.

**Detection using SUSHI:** We now briefly describe how the aforementioned attack could be discovered by SUSHI. When `processRequest` is symbolically executed, the value of variables `sUname` and `sPwd` should be initialized with symbolic literals and let them be  $x$  and  $y$ . Then, by executing the statements one by one, at line number 13 where the SQL query is submitted, the symbolic value of `strCmd` is a concatenation of the following string terms.

1. constant word `SELECT * FROM users \nWHERE uname='`

2. term  $x_{r_{\rightarrow}}^+[0, 16]$
3. constant word `' AND pwd='`
4. term  $y_{r_{\rightarrow}}^+[0, 16]$
5. constant word `'`

Let the string expression (concatenation of the above terms) be  $S_1$ . Here  $x_{r_{\rightarrow}}^+[0, 16]$  represents the output of the `message()` function on `sUname`, i.e., to replace every single quote with its escaping form and then to perform a substring operation on the user input. Similar is  $y_{r_{\rightarrow}}^+[0, 16]$ . Now by associating the symbolic string expression with predefined attack patterns, we can construct SISE equations. For example, the following is a sample SISE equation, based on one of the pre-collected SQL injection attack patterns:

$$S_1 \equiv \text{uname=}([\^']|')^* \text{ OR *uname}<>''$$

Intuitively, the SISE equation asks the following question: after all the sanitation procedures are applied, is it feasible to make the `WHERE` clause of the SQL query essentially a tautology (by `"OR uname<>' "`)? Using SUSHI, we are able to generate the *shortest* attack strings in 1.4 seconds. The detailed information is shown in Figure 6. The first four columns show the size of FST and FSA used in solving the SISE equation. The last column shows the overall time cost.

## 6.2 Generating Shortest XSS Exploits

We demonstrate how *reusable* attack pattern rules are represented in SISE. We show how SUSHI is used for analyzing one recently discovered XSS vulnerability [16] in Adobe Flex SDK 3.3. A file named `index.template.html` is used for generating wrappers of application files in a FLEX project. It takes a user input in the form of `"window.location"` (URL of the web page being displayed), which is written into the DOM structure of the HTML file using `document.write(str)`.

Clearly, the unfiltered input could lead to XSS (a tainted analysis [19] could identify the existence of the vulnerability). However, to precisely craft a working exploit is still not a trivial work, as several constraints have to be satisfied before the injected JavaScript code could work. For example, the injected JavaScript tag should not be contained in the value of an HTML attribute (otherwise, it will not be executed). In addition, the resulting HTML should remain syntactically correct, at least until the parser reaches the injected JavaScript code.

Example	FST_States	FST_Trans	FSA_States	FSA_Trans	Time
Bypass	238	1634	17	62	1.4s
XSS	0	0	272	4217	74.1s

**Figure 6. Attack Signature Generation Cost**

Rule	Regular Expression Pattern
XSS	<code>.*&lt;script&gt;alert('XSS found!')&lt;/script&gt;.*</code>
EffectiveScript	<code>.*[a-zA-Z0-9_]+ *="[^"]*"&lt;script.*&gt;.*</code>
MatchTag	<code>.*&lt;embed[^&gt;]*&gt;.* ∩ .*&lt;/embed[^&gt;]*&gt;.*</code>

**Figure 7. Rules in RHS**

SUSHI can help generating the attack string precisely. In fact, SUSHI generates the following attack string which is, first of all *working*, and is *shorter* (if not the shortest) than the exploit given in the original securitytracker post [16].

```
\ "<script>alert('XSS found!')</script>
```

In the following, we briefly describe how the SISE equation is constructed for generating the exploit. The LHS (left hand side) of the equation is a concatenation of three strings, two constant words and one variable. The variable represents the unsanitized user input. The two constant words represent the other parts of the parameters collected and combined by the vulnerable JavaScript code snippet. The two constant words are obtained by manual analysis that simulates symbolic execution on the vulnerable JavaScript snippet. The size of the constant words in LHS (combined) is 445 characters long. The RHS is a conjunction of a number of attack patterns and filter rules as shown in Figure 7. Notice that these attack patterns, apparently, are *reusable*.

The XSS pattern is straightforward. It requires that the JavaScript `alert()` function eventually shows up in the combined output. Then the `EffectiveScript` rule forbids the JavaScript snippet to be embedded in any HTML attribute definition (thus ineffective). The `MatchTag` rule requires that an HTML beginning tag must be matched by an ending tag (in our case the “`<embed>`” tag). Clearly, the above rules are general and can be applied to analyzing other XSS attacks.

The cost of finding the shortest solution is shown in Figure 6. Notice that the solution cost is expensive because the equation constructed is unusually large (the size of LHS is 445). This is reasonable compared with similar efforts in the area, e.g., HAMPI [13].

## 7 Related Work

String analysis, i.e., analyzing the set of strings that could be produced by a program, emerged as a novel technique for analyzing web applications, e.g., compatibility check of XHTML files against schema [2], security vulnerability scanning [6, 7], and web application verification [1]. Solving string constraints is one of the many directions for tackling command injection attacks (e.g., tainted analysis [19], forward string analysis [3], run-time hardening [8]), black-box testing [11]). While the aforementioned

techniques are mainly *dynamic* analysis, string analysis is mostly static. Static analysis helps to discover vulnerabilities before web applications are deployed.

In general, there are two interesting directions of string analysis: (1) *forward analysis*, which computes the image (or its approximation) of the program states as constraints on strings and other primitive data types; and (2) *backward analysis*, which usually starts from the negation of a property and computes backward. Most of the related work (e.g., [1–3, 15]) fall into the category of forward analysis. The work presented in this paper is *backward*. Compared with forward string analysis, the backward string analysis is able to generate attack strings as hard evidence of a vulnerability and avoids *false positives*. However, it suffers from false negatives, i.e., there are cases where vulnerabilities are ignored by the analysis.

SISE can be regarded as a variation of the *word equation* problem [17]. Note that in a word equation, only word concatenation is allowed. In SISE, various popular `java.regex` operations are supported.

SISE is a continuation of our earlier efforts of building a unified symbolic execution framework [6]. The SISE framework subsumes the string constraint framework outlined in [6], in which a number of incomplete heuristics algorithms are used for handling string concatenation and constant string replacement, suffering from imprecision. This paper provides a both sound and complete algorithm, which is able to produce all possible solutions of a SISE constraint. As shown in §3 of this paper, the modeling of various regular replacement semantics largely improves the analysis precision. The SUSHI constraint solver is implemented and is applied to a number of case study examples, which allows us to verify the effectiveness of the proposed string analysis in practice.

The closest work to ours is probably the HAMPI string constraint solver [13], which is also a backward analysis. HAMPI supports solving string constraints with context-free components, which are essentially unfolded to regular language within a bound. HAMPI, however, does not support string replacement nor regular replacement, which significantly limits its ability to reason about sanitation procedures. In addition, the unfolding of context-free components limits its scalability. Similarly, Hooimeijer and Weimer’s work [9] in the decision procedure for regular constraints does not support regular replacement. Another close work to ours is Yu’s automata based forward/backward string analysis [22]. Yu *et al.* use language based replacement [23] to handle regular replacement. Imprecision is introduced in the over-approximation during the language based replacement. Conversely, our analysis considers the delicate differences among the typical regular replacement semantics.

In [21] Wassermann and Su combined string analysis and

taint analysis for discovering Cross-Site Scripting attacks. Their analysis is forward and does not directly generate attack signatures. Finite state transducer is used in [21] for modeling regular replacements, however, it does not distinguish between the various semantics such as the greedy and the reluctant, which is addressed by this paper.

## 8 Conclusion

This paper introduces a string constraint solver called SUSHI. We show that a fragment of the general string equation problem, called Simple Linear String Equation (SISE), can be solved using automata based approach. Finite state transducer is used for precisely modeling several different semantics of regular substitution. This helps to reduce false positives of a string analysis. The SUSHI constraint solver is implemented and it is applied to analyzing security of several small web applications. The experimental results show that SUSHI works efficiently in practice. We plan to integrate SUSHI with existing symbolic execution engines to automate vulnerability exploration. Future directions include expanding the solver to consider context-free components, and incorporating temporal logic operators for modeling malicious attack behaviors. We are also interested in applying the work to automated grading in computer science education.

**Acknowledgment:** this work is partially supported by NSF under contract DUE-0836859 and DUE-0837020.

## References

- [1] N. Bjørner, N. Tillmann, and A. Voronkov. Path feasibility analysis for string-manipulating programs. In *Proceedings of the 15th International Conference on Tools AND Algorithms for the Construction and Analysis of Systems (TACAS 2009)*, pages 322–336. Springer, 2009.
- [2] A. S. Christensen, A. Møller, and M. I. Schwartzbach. Extending java for high-level web service construction. *ACM Trans. Program. Lang. Syst.*, 25(6):814–875, 2003.
- [3] A. S. Christensen, A. Møller, and M. I. Schwartzbach. Precise analysis of string expressions. In *Proc. 10th International Static Analysis Symposium, SAS '03*, volume 2694 of *LNCS*, pages 1–18. Springer-Verlag, June 2003. Available from <http://www.brics.dk/JSA/>.
- [4] CIRT INC. Nikto. available at <http://www.cirt.net/nikto2>.
- [5] X. Fu and C. Li. Modeling regular replacement for string constraint solving. In *Proc. of the 2nd NASA Formal Methods Symposium*, pages 67–76, 2010.
- [6] X. Fu, X. Lu, B. Peltserger, S. Chen, K. Qian, and L. Tao. A Static Analysis Framework for Detecting SQL Injection Vulnerabilities. In *Proceedings of 31st Annual International Computer Software and Applications Conference (COMP-SAC 2007)*, pages 87–96, 2007.
- [7] C. Gould, Z. Su, and P. Devanbu. JDBC Checker: A Static Analysis Tool for SQL/JDBC Applications. In *Proceedings of the 26th International Conference on Software Engineering*, pages 697–698, 2004.
- [8] W. Halfond and A. Orso. AMNESIA: Analysis and Monitoring for NEutralizing SQL-Injection Attacks. In *Proceedings of the 20th IEEE/ACM international Conference on Automated Software Engineering*, pages 174–183, 2005.
- [9] P. Hooimeijer and W. Weimer. A decision procedure for subset constraints over regular languages. In *Proceedings of the 2009 ACM SIGPLAN conference on Programming language design and implementation*, pages 188–198, 2009.
- [10] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, 1979.
- [11] Y. Huang, S. Huang, T. Lin, and C. Tsai. Web application security assessment by fault injection and behavior monitoring. In *Proceedings of the 11th International World Wide Web Conference (WWW 2003)*, 2003.
- [12] R. M. Kaplan and M. Kay. Regular models of phonological rule systems. *Computational Linguistic*, 20(3):331–378, 1994.
- [13] A. Kiezun, V. Ganesh, P. J. Guo, P. Hooimeijer, and M. D. Ernst. Hampi: A solver for string constraints. In *Proc. of 2009 International Symposium on Testing and Analysis (ISSTA'09)*, 2009.
- [14] J. C. King. Symbolic execution and program testing. *Communications of the ACM*, 19(7):385–394, 1976.
- [15] C. Kirkegaard and A. Møller. Static analysis for Java Servlets and JSP. In *Proc. 13th International Static Analysis Symposium, SAS '06*, volume 4134 of *LNCS*, August 2006.
- [16] labs@gdssecurity.com. Adobe flex sdk input validation bug in 'index.template.html' permits cross-site scripting attacks. <http://www.securitytracker.com/alerts/2009/Aug/1022748.html>, 2009.
- [17] M. Lothaire. *Algebraic Combinatorics on Words*. Cambridge University Press, 2002.
- [18] A. Møller. The dk.brics.automaton package. available at <http://www.brics.dk/automaton/>.
- [19] A. Nguyen-Tuong, S. Guarnieri, D. Greene, J. Shirley, and D. Evans. Automatically hardening web applications using precise tainting. In *Proceedings of the 20th IFIP International Information Security Conference*, 2005.
- [20] SPI Dynamics. Webinspect: Security throughout the application lifecycle. Datasheet. [http://www.spidynamics.com/assets/documents/WebInspect\\_DataSheets.pdf](http://www.spidynamics.com/assets/documents/WebInspect_DataSheets.pdf).
- [21] G. Wassermann and Z. Su. Static detection of cross-site scripting vulnerabilities. In *Proc. of the 30th International Conference on Software Engineering*, pages 171–180, 2008.
- [22] F. Yu, M. Alkhalaf, and T. Bultan. Generating vulnerability signatures for string manipulating programs using automata-based forward and backward symbolic analyses. In *Proceedings of the 24th IEEE/ACM International Conference on Automated Software Engineering (ASE 2009)*, 2009.
- [23] F. Yu, T. Bultan, M. Cova, and O. H. Ibarra. Symbolic string verification: An automata-based approach. In *Proc. of the 15th SPIN Workshop on Model Checking Software*, pages 306–324, 2008.

# Towards a Structured Model for Software Vulnerabilities

Ming Huang, Yisha Lu, Qingkai Zeng

State Key Laboratory for Novel Software Technology, Nanjing University  
Department of Computer Science and Technology, Nanjing University  
Nanjing, P.R. China  
minghai1984@gmail.com, zqk@nju.edu.cn

**Abstract**—As the core of various security problems, software vulnerability is the main challenge in the information security field. Particularly, software vulnerability modeling is an actively defensive measure for software security, aiming to detect and eliminate the potential vulnerabilities before they have been exploited. In this paper, a structured and graphic method for modeling vulnerability is proposed. The method combines the advantages of existing methods to depict and reason about security vulnerabilities, which would be useful for a better understanding of the nature of vulnerabilities, and become an effective way to detect and prevent the software vulnerabilities. The practical application result shows that GSM could reveal some related information and properties that existing methods cannot find in the vulnerability databases in public.

**Keywords**—software vulnerability; vulnerability modeling; taxonomy; vulnerability database; information security

## I. INTRODUCTION

Along with the rapid development of computer and network technology, software plays an increasingly important role in everyday life. Although this change brings about many benefits, new dangers arise as well, since the internet-related security events are continually growing, and security issues have become more acute. It is well recognized that software vulnerabilities are the root of most security incidents on the internet [1]. Despite many efforts being done to eliminate or mitigate them, the vulnerabilities in software do not disappear and are still emerging.

An increased understanding of the nature of software vulnerabilities, their manifestations, and the mechanisms to eliminate or prevent them can be achieved by the creation of taxonomies for software vulnerabilities, and the application of learning, visualization, and statistical methods or tools on a representative collection of software vulnerabilities. Nowadays, analysis of security vulnerabilities has typically been approached in one of two ways: (i) using real data to perform statistical analysis and develop a classification (or modeling) [2,3], and (ii) providing a degree of formalism by modeling vulnerabilities and attack characteristics [4,5].

In this paper, we present a structured and graphic method to depict and reason about security vulnerabilities. Our method combines the advantages of existing methods mentioned above: the vulnerability information is analyzed, in conjunction with a formal modeling method, to develop a Graph-based Structured Model (GSM) to formalize and reveal the nature of security

vulnerabilities. It would be useful for a better understanding of the nature of vulnerabilities, and help the software development community beware of the security vulnerabilities that they might have introduced and get feedback to improve continuously. The practical application result shows that GSM could reveal some related information and properties that existing methods cannot find in the vulnerability databases in public. Further analysis indicates that GSM can be a foundation for other related research, such as model checking or classification of security vulnerabilities.

## II. BACKGROUND AND RELATED WORK

### A. Taxonomy of vulnerabilities

During the past thirty years, a number of vulnerability classifications have been proposed to abstract observed vulnerabilities into easy-to-understand classes. Representative examples include the PA study [6], RISOS [7], Spafford's taxonomy [8], Landwehr's taxonomy [2], and Aslam's taxonomy [9]. In addition to providing taxonomies, [2] and [3] perform statistical analysis of actual vulnerability data based on their own taxonomies.

The taxonomy and statistical analysis of actual vulnerability data can help system builders, administrators, and users beware of the types of security vulnerabilities and develop strategies to prevent or counter them. Vulnerability modeling could still benefit from classification efforts, as vulnerabilities in the same class are likely to have similar causes and features.

### B. Modeling security vulnerabilities

The research on vulnerability modeling has two key branches: modeling towards general vulnerabilities or specific static detection technology. Michael et al. [10] employ a FSM model constructed using system call traces. Chen et al. [11] find that exploits must pass through multiple elementary activities. By developing an elementary FSM (eFSM) for each elementary activity, it is feasible to develop FSM models of vulnerable operations and possible exploits. Wilander [12] proposes dependence graphs decorated with type and range information as a generic way of modeling security properties of code. However, the problem that converting the process of analyzing integer flaws to graph isomorphism problem is an NP-complete. D. Byers et al. [13] introduce the VCGs to describe the vulnerabilities. This method provides an easier understanding of the vulnerability. The use of this method may result in a loss of information about the vulnerability.

### III. THE MODEL

#### A. Characteristics and Observations

Based on the serious observations [14,15], it is able to quickly and definitely arrive at a solution that all the work classified attacks or vulnerabilities based on some inherent characteristic of the attack or vulnerability itself. By integrating several security-related properties or characteristics of the vulnerabilities, we think that it is a great way for us to understand the nature of software vulnerabilities. These characteristics and observations motivate us to develop the GSM modeling method, capable of expressing the software vulnerabilities.

A vulnerability must be due to at least one flaw, but it is possible for a flaw not to cause any vulnerability. Therefore, a flaw is one of the most important and direct factors for a vulnerability. This conclusion can also be drawn by analyzing the basic dimensions of typically vulnerability taxonomies [15,16]. Characteristics which are also commonly called features or attributes are the bases for the development of successful classifications. Benefiting from these classifications, we introduce security flaw to be used as an element (denoted by "Flaw") for our GSM model building.

A vulnerability has to exist within a context [17], and hence it is described in terms of environmental factors. In fact, there is no "ubiquitous" vulnerability. The software development language, the software build environment, the runtime environment, etc. are the objective conditions that contribute to the existence of the vulnerability. Most software vulnerabilities result from a mismatch for a variety of reasons between the programmer's original expectation and the environmental conditions in which programs execute. Our approach relies on an empirically supported belief [18,19,20] that the environment plays a significant role in triggering security flaws that lead to security violations (denoted by "Pre-Condition").

The result (impact) of vulnerability describes the possible consequences of a vulnerability if exploited, and usually is used as a synonym for vulnerability itself. The immediate result of a vulnerability is the loss of some security properties. However, once a single property is breached, many more attacks might be carried out with the privileges an attacker has acquired to impact the system's confidentiality, integrity, and availability. In order to maintain compatibility with existing usage, we will also regard the result of vulnerability as an important composition of our method (denoted by "Result").

Formally, each characteristic discussed above is defined as a triple  $(C_a, N_a, V_a)$  in our method, where:

1.  $C_a$  denotes a set of characteristic categories, which specifically refer to the types of vulnerability characteristics. Namely,  $C = \{\text{Pre-Condition, Flaw, Result}\}$ .
2.  $N_a$  is set of characteristic names. Each characteristic is uniquely identified with a name.
3.  $V_a$  denotes a set of values  $\{v_1, v_2, \dots, v_n\}$ , where  $v_i$  and  $v_j$  ( $i \neq j$ ) may belong to a same characteristic named by  $N_i$ , that is to say, a characteristic might have more than one value.

#### B. Visual Representation

Benefiting from the VCG method [13], we introduce a formal graph representation for GSMs so that allowing the developer to get an overview of the connections between the

vulnerabilities and their characteristics. The model element is corresponding to the vulnerability characteristic discussed in Section 3.1 (Figure 1), and there will be no confusion to figure out at which abstraction level the causes should be modeled.

Visual representation	Model element
	Pre-Condition node
	Flaw node
	Conjunction node <sup>a</sup>
	Result node

a. A conjunction node must contain two or more Pre-Condition and/or Flaw nodes; arbitrary combinations are permitted.

Figure 1. Visual representation of GSMs

A Pre-Condition node or Flaw node can also be called simple node. For each non conjunction node, there also has a label, a triple  $\langle C_a(N), N_a(N), V_a(N) \rangle$ , to show the connotation.  $C_a(N)$  can be (oftentimes) omitted, since the visual shape has suggested its characteristic category. Some concepts and components that will be applied to construct the model should be introduced here in advance.

*Predecessor and Successor:* The predecessor-successor relationship in a GSM graph is represented with an arrow which points to the successor node from the predecessor one. The condition represented by the predecessor node is a necessary condition for the existence of the successor one.

*GSM chain:* A one-way chain which contains all nodes and edges along the path from one of the uppermost nodes until to the result node is called a *GSM chain*. Sequences in the GSM chain represent conditions that are ordered by some form of causality. The area covered by oval-shaped shadow shows an instance in figure 2.

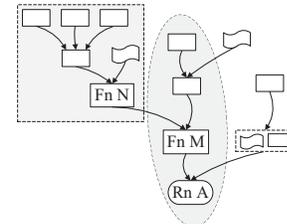


Figure 2. GSM Chain and GSM Branch

*GSM branch:* As shown in figure 2, the sub-graph in the rectangle area with shadow constructs a *GSM branch* (called *GSM branch N*). If node N has a successor node M, then *GSM branch N* is the predecessor branch of node M.

A GSM graph is a directed acyclic graph in which all nodes represent causes but the Result node, and edges represent relationships between causes. Every GSM has a designated Result node, the only node without successors. A node may have more than one predecessor but is required to have only one successor if at all. A conjunction node can have direct influence upon its successor only when all the clauses of a conjunction must hold or be eliminated.

#### IV. THE IMPLEMENTATION OF MODEL

##### A. Selecting the characteristics

Prior to constructing the model, we have studied deeply many vulnerabilities, and focused on the characteristics we discussed to analyze the information about vulnerability. Table I lists some analysis results based on our method.

TABLE I. THE CHARACTERISTICS AND THEIR VALUES (PARTLY)

Name	Category	Value
Using Memory Buffer	Pre-Condition	using heap buffer; using stack buffer
Memory Exploited Mechanism	Result	unbounded transfer(CVE-2002-1337); out-of-bound read(CAN-2004-1940); out-of-bound write(CVE-2004-2620)
Length Parameter Error	Flaw	inconsistency(CVE-2000-0655); simple math error(CAN-2005-3120); incorrectly updating parallel counters; not accounting for size differences between formats(CVE-2001-0334);
Parameter Error	Flaw	missing parameter(CVE-2004-0276); extra parameter(CAN-2003-1014); undefined parameter(CAN-2002-1488); unsupported parameter(CVE-2001-0650); wrong type(CVE-2004-0270); uncompleted parameter(CAN-2003-0195)
Special Element	Pre-Condition	delimiter; input termination; input leader; control character; quoting element; comment; macro symbol; whitespace; wildcard element; null(character or byte)

b. A typical instance in US-CERT/NIST is given in parenthesis

It is unambiguous to find out that one characteristic name may map to more than one characteristic value. This situation is an important phenomenon which has been brought to our great attention. To avoid these mistakes, we will make use of more precise vocabularies to describe and characterize the relevant result of vulnerability if exploited.

##### B. Constructing the GSM model

Several conventions are given firstly for conveniently:

1. For a simple node  $N$ ,  $C_a(N)$  denotes the characteristic category;  $N_a(N)$  is the characteristic name which is also the unique identifier in the model;  $V_a(N)$  is the characteristic value.

2. Two auxiliary databases are introduced: (i) *Attribute\_DB* is used to store the vulnerability characteristics in the form of a triple  $\langle C_a(N), N_a(N), V_a(N) \rangle$ , and (ii) *Cause\_DB*, is used to maintain the causal relationships between nodes in a GSM.

The modeling process is divided into two phases.

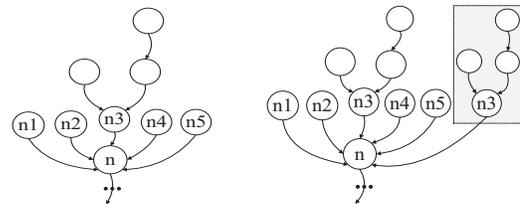
###### 1) Preliminary modeling phase.

Preliminary modeling phase starts with a single result node, representing the vulnerability being modeled. (a) Analyze and label the result node. Add this node to the *UnAnalyzedSet*. All of the aided sets of nodes are empty at the beginning. (b) Examine whether the *UnAnalyzedSet* is empty. If not, continue; otherwise go to the *phase 2*. (c) Select a node  $N$  from the *UnAnalyzedSet* and call the algorithm 1. (d) Call the algorithm 2 and get the *PredecessorSet* of node  $N$ . (e) Add all elements of *PredecessorSet* to be the predecessor nodes of node  $N$  in the GSM; Conjoin the *PredecessorSet* and *UnAnalyzedSet* to the *UnAnalyzedSet*, and after that empty the *PredecessorSet*. (f) Make use of various means to acquire the *NewPredecessorSet* to enrich and improve the model. (g) Repeat a similar process

in *step e* on the *NewPredecessorSet*. Call the algorithm 3. (h) Check the predecessor nodes of node  $N$  whether there are two nodes or more having the conjunctive relations, and put the conjunction nodes if existed into the *ConjunctionSet*. (i) Remove the node  $N$  from the *UnAnalyzedSet* and go back to *step b* above.

###### 2) Optimized modeling phase.

This phase examines each conjunction node in the *ConjunctionSet* combined in *step 1.h* (*step h* of phase 1) or newly produced in *phase 2*, and removes any redundant conjunction node. (a) Examine whether the *ConjunctionSet* is empty. If not, continue; otherwise go to the *step j*. (b) Compare each conjunction node with others in the *ConjunctionSet*, and remove any redundant conjunction node whose clauses all belong to another one at the same time. (c) Select a node  $CN$  from the *ConjunctionSet*. For each simple node in  $CN$ , add all of its predecessor nodes into the *PredecessorSet*. (d) Call the algorithm 4. (e) Examine whether the *CommonNodeSet* is empty. If true, continue; otherwise for a node  $N'$  in *CommonNodeSet*, and if it has the successor node, then copy *GSM branch  $N'$*  as its predecessor, and remove the node  $N'$  from the *CommonNodeSet*. Repeat *step e*. (f) Combine all the clauses of node  $CN$  and replace them with node  $CN$ . (g) Add all elements of *PredecessorSet* to be the predecessor nodes of node  $CN$ , and keep other nodes intact. (h) Remove the node  $CN$  from the *ConjunctionSet*. (i) Analyze all the nodes in *PredecessorSet* to check whether there are two nodes or more having the conjunctive relations, and put the conjunction nodes if existed into the *ConjunctionSet*. After that, empty the *PredecessorSet* and go to *step 2.a*. (j) Adjust the name of the nodes in the GSM to avoid duplication.



(a) Before the operation of COPY (b) After the operation of COPY

Figure 3. Example of copying GSM branch

TABLE II. DESCRIPTION OF THE ALGORITHMS

Algorithm	FUNCTION
alg. 1	Make use of the information in the <i>Cause_DB</i> to produce the <i>CandidateSet</i> of node $N$ and add the new vulnerability characteristics to the <i>Attribute_DB</i> .
alg. 2	Produce the <i>PredecessorSet</i> of node $N$ by checking the every node in <i>CandidateSet</i> .
alg. 3	Analyze the nodes in <i>NewPredecessorSet</i> to get vulnerability characteristics to the <i>Attribute_DB</i> , and add the causal relationships between these vulnerability characteristics and the current node being analyzed to the <i>Cause_DB</i> . Empty the <i>NewPredecessorSet</i> at last.
alg. 4	Add the common nodes between the node $CN$ and other conjunction nodes in <i>ConjunctionSet</i> to the <i>CommonNodeSet</i> , used to store each simple node that belongs to at least two different conjunction nodes.

Figure 3 gives a visual example for *step 2.e*. As a matter of convenience, here we do not differentiate the types of the nodes

and each simple node is simply expressed as a circle. And the operation of adjusting the name of nodes in *step 2.j* will solve the problem of duplication in such transformation of copying GSM branch. Due to space limitations, any algorithm appeared in the steps above is described simply in table II.

## V. EXPERIMENT AND DISCUSSION

We have applied the GSM approach to analyze a lot of security vulnerabilities cataloged by CERT/CC. This section gives a typical example to demonstrate the modeling process.

### A. NULL HTTPD Heap Overflow Vulnerability

*Null HTTPD* is a multithreaded web server for Linux and Windows platforms. There is a heap-based buffer overflow, known as CVE-2002-1496 [21], in *Null HTTP Server* 0.5.0 and earlier version. The source code of the vulnerable function is shown in Figure 4.

```

1  postData=calloc(contentLen + 1024, sizeof(char)); x=0; rc=0;
2  pPostData=postData;
3  do{
4      rc=recv(sock, pPostDta, 1024, 0);
5      if(rc==-1){
6          colseconnect(sid,1);
7          return;
8      }
9      pPostData+=rc;
10     x+=rc;
11 } while ((rc==1024)||(x<contentLen));

```

Figure 4. Source Code of the function *ReadPOSTData*

We now describe how to construct the GSM model for this known vulnerability.

#### 1) Preliminary modeling phase:

The result node is picked up for analysis. It could cause the impact of “Out-of-bound Write” if exploited, so it is marked with the label *<Result, Buffer Exploited Mechanism, Out-of-bound Write>*, and add it to the *UnAnalyzedSet* with the name RN. Then the node RN is used as the parameter to call the algorithm 1, and get the *CandidateSet* with a node FN 1, labeled *<Flaw, Length Parameter Error, Inconsistency>*. According to the algorithm 2, the node FN 1 is added to the *PredecessorSet*, and the *CandidateSet* is empty again. According to the *step 1.e*, the node FN 1, the only element of *PredecessorSet*, is added as the predecessor node of node RN, and then the *UnAnalyzedSet* equals to the set {RN, FN 1}.

By code review in *step 1.f*, the function *ReadPOSTData* copies a user specified string from a socket as an *input*, which is also a necessary precondition for “Out-of-bound Write”. We add it to the *NewPredecessorSet* named PN 1 with the label *<Pre-Condition, User Provided Input Copy, Socket>*. After *step 1.g*, the node PN 1 is added to be the predecessor node of node RN, and the *UnAnalyzedSet* equals to the set {RN, FN 1, PN 1}. In *step 1.h*, it is required to analyze the predecessor of node RN to check whether the nodes PN 1 and FN 1 have conjunctive relations. Through analysis, we introduced a conjunction node CN 1 which contains the nodes PN 1 and FN 1, and add it to the *ConjunctionSet*. We remove the node RN from the *UnAnalyzedSet* and go back to *step 1.b*. Eventually we complete the preliminary modeling phase (figure 5).

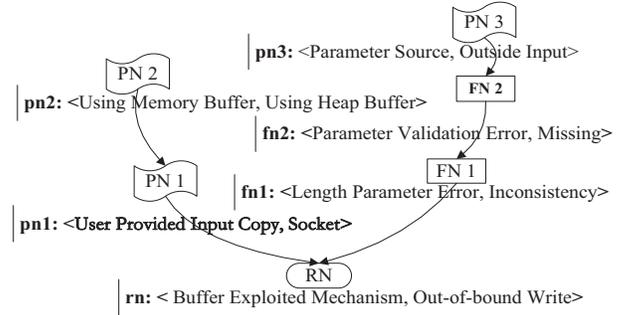


Figure 5. The preliminary model of *phase 1*

#### 2) Optimized modeling phase:

Since there is only one conjunction node CN 1, we get a new *PredecessorSet* with the nodes PN 2 and FN 2 by *step 2.c.*. The node CN 1 is used as the parameter to call the algorithm 4, and only an empty *CommonNodeSet* is got. Then the nodes PN 1 and FN 1 are combined and replaced with node CN 1 in the GSM graph. All elements of *PredecessorSet* (PN 2 and FN 2) are added to be the predecessor nodes of CN 1, and the *ConjunctionSet* would be empty again. In *step 2.i*, we again analyze whether the nodes PN 2 and FN 2 have conjunctive relations and gain a new conjunction node CN 2 which contains the nodes PN 2 and FN 2. Before going back to *step 2.a*, the node CN 2 is added to the *ConjunctionSet* and the *PredecessorSet* is reset empty. Finally we complete the optimized modeling phase (figure 6).

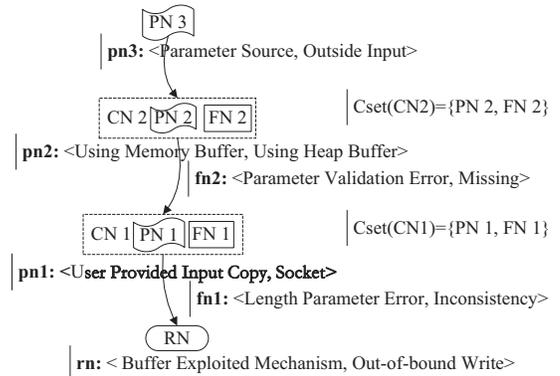


Figure 6. The Optimized model of *phase 2*

### B. Discussion

With increasing complexity and distribution of software, the importance of heeding to security is becoming more and more critical. While the vulnerabilities are often very similar, the lack of knowledge about their nature often leads to reintroduction of the vulnerabilities in new software. From the example of CVE-2002-1496, we can find out that comparing to the existing public vulnerability databases [22,23], which describe the security vulnerabilities with natural language, GSM model provides a more comprehensive method to depict and reason about security vulnerabilities. The analysis of the vulnerability characteristics and their causal relation would be useful for a better understanding of the vulnerability and reusing the analytic results of the vulnerability.

We also have applied GSM model to several other known vulnerabilities (e.g. CVE-2002-0902, CVE-2006-0806, and CVE-2006-0437), and found some similarities between the vulnerabilities. For example, despite the fact that four years separate the vulnerabilities CVE-2002-0902 and CVE-2006-0437 and that they concern the same software, it appears that the cross-site scripting (XSS) vulnerabilities are often basically the same. This indicates that vulnerabilities could be avoided by using some kind of vulnerability modeling, like GSM model. By taking advantage of the GSM chain, for example pattern-matching, our GSM method might contribute positively to model-based checking of security vulnerabilities. With the benefits of giving details of the vulnerability characteristics, it is quite possible to eliminate or mitigate some characteristics of the vulnerabilities to prevent a vulnerability. This three-dimensionality structure  $\langle C_a(N), N_a(N), V_a(N) \rangle$  contributes to help those who, with different knowledge background, need to detect or understand the security vulnerabilities.

## VI. CONCLUSION AND FUTURE WORK

Comprehensive analysis of vulnerabilities should be an integral part of software maintenance. In this paper, we develop a GSM method to formalize and reveal the nature of security vulnerabilities by combining the advantages of existing methods. It would be useful for a better understanding of the vulnerability and reusing the analytic results of the vulnerability, and can be a foundation for other related research on software vulnerabilities, such as model checking or classification of security vulnerabilities. He/she even can complement the model with the information that he/she concerned for some specific applications.

The information of the vulnerability characteristics and the causal relationships is determined by analysis of the information in the vulnerability databases carefully and the corresponding source code in question. we may have to use other techniques, like static analysis tools, execution traces, and live debugging, to gather more information to facilitate the model analysis and building. The next step of our work will be to improve our vulnerability modeling on software vulnerabilities with more vulnerability characteristics, and develop the usable tool to support practical application.

## ACKNOWLEDGMENT

This work was supported by the National Science Foundation of China under Grant No. 60773170, 60721002, 90818022, the National 863 High Technology Research and Development Program of China under Grant No. 2006AA01Z432, and the Specialized Research Fund for the Doctoral Program of Higher Education of China under Grant No. 200802840002. We are grateful to the anonymous reviewers for their helpful comments on the earlier drafts of this paper.

## REFERENCES

[1] Frank Piessens, A taxonomy of causes of software vulnerabilities in internet software. Supplementary Proceedings of the 13th International Symposium on Software Reliability Engineering, USA, 2002: 47-52.  
 [2] Landwehr C E, Bull A R, McDermott J P, et al. A taxonomy of computer program security flaws, with examples. ACM Computing Surveys, September 1994, 26(3): 211-254.

[3] U. Lindqvist and E. Jonsson. How to systematically classify computer security intrusions. Proceedings of the 1997 IEEE Symposium on Security and Privacy, pages 154-163, Oakland, CA, May 4-7, 1997.  
 [4] R. Ortalo, Y. Deswarte and M. Kaaniche, Experimenting with quantitative evaluation tools for monitoring operational security. IEEE Transactions on Software Engineering, 25(5): 633-650, 1999.  
 [5] O. Sheyner, J. Haines, S. Jha, et al. Automated generation and analysis of attack graphs. Proceedings of the 2002 IEEE Symposium on Security and Privacy, pp.254 -265, 2002.  
 [6] Bisbey, R. II & Hollingsworth, D. Protection analysis project final report (ISI/RR-78-13, DTIC AD A056816). Marina del Rey, CA: University of Southern California Information Sciences Institute, 1978.  
 [7] Abbott R P, Chin J S, Donnelley J E, et al. Security analysis enhancements of computer operating systems. Technical Report NBSIR 76-1041, Institute for Sciences Technology, National Bureau of Standards, Washington, D.C., April 1976.  
 [8] Eugene H. Spafford, Common system vulnerabilities. Proceedings of the Workshop on Future Directions in Computer Misuse and Anomaly Detection, pp. 34-37, 1992.  
 [9] Taimur Aslam. A taxonomy of security faults in the unix operating system. M.S. Thesis, Purdue University, 1995.  
 [10] C. Michael, A. Ghosh. Simple, state-based approaches to program-based anomaly detection. ACM Transactions on Information and System Security. Pages: 203-237. Vol.5 No.3. Aug. 2002.  
 [11] Shuo Chen, Zbigniew Kalbarczyk, Jun Xu, Ravishankar K. Iyer. A data-driven finite state machine model for analyzing security vulnerabilities. Proceedings of the 2003 International Conference on Dependable Systems and Networks, San Francisco, 2003: 605-614.  
 [12] John Wilander. Modeling and visualizing security properties of code using dependence graphs. Proceedings of the 5th Conference on Software Engineering Research and Practice in Sweden (SERPS'05), Vasteras, Sweden, October 2005.  
 [13] D. Byers, S. Ardi, N. Shahmehri, C. Duma. Modeling software vulnerabilities with vulnerability cause graphs. Proceedings of the 22nd IEEE International Conference on Software Maintenance (ICSM'06), pages 411-422, Philadelphia, PA, USA, September 24-27, 2006.  
 [14] Yisha Lu, Qingkai Zeng. Survey of software vulnerability taxonomies. Journal of Computer Applications, 2008, 28(9): 2244-2248.  
 [15] Ming Huang, Qingkai Zeng. Research on attributes of software vulnerability taxonomies. Computer Engineering. 2010, 36(1): 184-186.  
 [16] V.M. Ijure, R.D. Williams, Taxonomies of attacks and vulnerabilities in computer systems, Communications Surveys & Tutorials, IEEE, 10(1): 6-19, 2008.  
 [17] Sufatrio, Roland H. C. Yap, Liming Zhong. A machine-oriented vulnerability database for automated vulnerability detection and processing. Proceedings of the 18th Large Installation System Administration Conference (LISA'04), pages 47-58, USA, 2004.  
 [18] W. Du, A. P. Mathur. Testing for software vulnerability using environment perturbation. International Conference on Dependable Systems and Networks(DSN 2000). New York, USA, 2000: 603-612.  
 [19] S. Garfinkel and G. Spafford. Practical unix & internet security. O'Reilly & Associates, Inc., 1996.  
 [20] W. L. Fithen, S. V. Hernan, et al. Formal modeling of vulnerability. Bell Labs Technical Journal 8, 4 (2004), 173-186.  
 [21] National Vulnerability Database (NVD). <http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2002-1496>.  
 [22] US-CERT/NIST. National vulnerability database. <http://nvd.nist.gov>.  
 [23] SecurityFocus. SecurityFocus vulnerability database. <http://www.securityfocus.com/vulnerabilities>.

# Dynamic and semantic social networks analysis: a new model based on a multidisciplinary approach.

Christophe Thovex\*, Francky Trichet\*

\*LINA, University of Nantes

Laboratoire d'Informatique de Nantes Atlantique (UMR-CNRS 6241)

2 rue de la Houssiniere, BP 92208 - 44322 Nantes cedex 03, France.

E-mail: (christophe.thovex) (francky.trichet)@univ-nantes.fr

## 1. Abstract

In 1977, L. C. FREEMAN published new metrics for Social Networks Analysis (SNA), mainly based on static graph-mining models. The objective of our work is to introduce new dynamic SNA models dedicated to SNA and to take the conceptual aspects of enterprises and institutions social graph into account. These conceptual aspects are embedded in trades oriented ontologies extracted from the endogenous information, connate to the studied social networks. The originality of our work is based on the definition of new multidimensional measures in SNA for new decision-making functions in Human Resource Management (HRM). Defined in the context of a new multidisciplinary approach (mainly physics and cognitive sciences), the contributions presented in this paper are: (1) a measure of *tension* of a social network, (2) an electrodynamic and predictive system for semantic recommendations about social graphs evolutions and (3) a measure of *reactance* of a social network used to evaluate the individual *stress* of its members.

## 2. Introduction

Current trends and needs of communication permanently require new functions and applications of social networking, as demonstrated by the constant eruption of new mode of socialisation (*cf.* Twitter, Facebook, Diigo). In comparison with the real spaces of exchange, these virtual spaces facilitate the static analysis and the emergence of metrics and methods dedicated to Social Networks Analysis (SNA). Measures of *centrality* such as those introduced by L. C. FREEMAN are the basic foundations of SNA [3]. A social network can be formalised with a (not)oriented, (not)labelled and weighted graph. From such a structure, two kinds of SNA can be differentiated: *dynamic SNA* and *semantic SNA*.

*Dynamic SNA* studies evolution of a social graph from a state  $S$  at a time  $t$ , to a state  $S'$  at a time  $t'$ . It is based on models and measures dedicated to structures and flows<sup>1</sup>. A *flow* is characterised by a count of units circulating between two vertices - *e.g.* electrical or hydraulic networks, road networks - and the possible ability to show implicit relationships between individuals involved into the social graph. *Dynamic SNA* enables the probabilistic prediction of social graphs evolution.

*Semantic SNA* studies the conceptual aspects of a social graph. It is based on the principles underlying conceptual graphs theory and semantic networks theory [9]. Semantic SNA usually refers to Semantic Web, Ontology Engineering [5] and logical inferences, in correlation with cognitive sciences. With the exponential growth of social networks and information flows, semantic SNA becomes crucial for knowledge discovery and knowledge management, from the enterprise content to the large communities of the Web.

Currently, not many works try to integrate the differentiated forms of analysis (dynamic and semantic). The purpose of our work consists in filling this gap by defining a new convergent system based on both dynamic and semantic analysis of Enterprises and Institutions Social Networks (EISN). Our approach is multidisciplinary since it is based on physics and cognitive sciences. It leads to the definition of a multidimensional model enabling the development of new decision-making tools for the optimisation of work or social and human capital management. In its current version, this model includes two new measures and a predictive system: (1) a measure of *tension* of a social network, (2) an electrodynamic and predictive system for semantic recommendations about social graphs evolutions and (3) a measure of *reactance* of a social network used to evaluate the individual *stress* of its members.

Our work is funded by the French State Secretariat for prospective and development of the digital economy, in the

<sup>1</sup>Flows models and measures [8], [10].

context of the SOCIOPRISE project. It is developed in collaboration with a French IT service and software engineering company which provides industry-leading software and implementation services dedicated to human capital management.

The rest of this paper is structured as follows. Section 3 introduces, in a synthetic way, the principles and methods respectively used for dynamic SNA and semantic SNA. Section 4 presents in details the approach we advocate to integrate dynamic and semantic SNA. Our contributions are based on (1) a bridge-building between radio-electronic principles to complete a dynamic analysis and (2) a bridge-building between our new physical measures of SNA and knowledge engineering. Our work is devoted to Enterprises and Institutions Social Networks Analysis - EISNA.

### 3. Unidimensional approaches

#### 3.1. Radio-electrical principles for dynamic EISNA

Some physics models are treated with help of graphs for the understanding and discovery of theoretical principles. In the electricity area, the KIRCHHOFF's *law of nodes* and *law of meshes* are the most well-known illustration of this trend. The originality of our work is to introduce the notion of *tension* of social network. This tension of a network is defined according to the notions of *crossing flow intensity* and vertex *resistance*. A vertex  $s$  directly connected with two other vertices  $r$  and  $t$  can be likened to a dipole which resistance is noted  $R$ . We use OHM's laws:

$$U_{rt} = R_s \cdot I_{rt} \text{ and } P_s = R_s \cdot I_{rt} \cdot I_{rt}^2 = U_{rt} \cdot I_{rt}^2 / R_s = U_{rt} \cdot I_{rt},$$

where  $U_{rt}$  represents the electrical tension depending on  $R_s$  and  $I_{rt}$ , and  $P_s$  represents the delivered power by a vertex of which maximal admissible power is noted  $P_{max}$ , with  $U_{max} = \sqrt{R \cdot P_{max}}$  and  $I_{max} = \sqrt{P_{max} / R}$ .

By applying OHM's law upon a social graph, it is possible to compute a *charge-capacity* ratio of the enterprise social network, by analogy with  $P_s, P_{max}$ . Our goal is to introduce a *stress* measure of individuals. This measure uses the *Joule effect* to estimate the enterprise social network components *warm-up* and to prevent risks of performance degradation, instability or breakdown (psychosocial troubles). It must be considered that the *social material* is *a priori* abstracted as a constant by initialising algorithms with  $\rho = 1$ , let  $T \cdot \rho = W = R \cdot I^2 \cdot \Delta t$ .

The experiment of E. BRANLY about radio-conduction (1890) has demonstrated that some flows can appear between points without visible connections. We use the radio-conduction principle to state a hypothesis that some significant flows of information can exist between two vertices which are not directly connected by an edge. To fulfil this hypothesis, our thesis stands on both the notions of induc-

tance and electromagnetic fields to introduce the emergence of implicit social networks induced by the flows between closest neighbours in an explicit social network.

AMPERE's theorem (1820) defines the circulation of the magnetic induction as a vector. This vector describes a set of circles  $\tau$  centred on a rectilinear conductor charged by an electrical intensity  $I$ , with  $d\vec{l}$  an infinitesimal movement, where  $\vec{B}$  is a magnetic induction expressed in Tesla ( $T = 10^4$  Gauss) and  $\mu_0$  the vacuum permeability index ( $= 4\pi \cdot 10^{-7}$ ) [6].

$$\oint_{\tau} \vec{B} \cdot d\vec{l} = \mu_0 \cdot I \quad (1)$$

According to this proposal, we add a simplification of an electromagnetic induction  $B$  with  $d$ , distance (m) between the conductor and a point of an euclidian plan, orthogonal to the conductor.

$$B = \frac{\mu_0}{2\pi} \cdot \frac{I}{d} \quad (2)$$

Ratio between the magnetic flow  $\Phi$  of a simple (*vs.* composed) and isolated electrical circuit and the electrical current running within the circuit is called proper inductance  $L$  of the circuit.

$$L = \frac{\Phi}{I} \quad (3)$$

In a composed circuit, also called network, it is convenient to take the mutual induction phenomenon into account. Mutual inductions is the electromagnetic interaction between magnetic elements depending on their respective polarization, orientation and proximity. Mutual inductance of circuits 1 and 2 is noted  $M_{1/2}$ .

$$M_{1/2} = \frac{\Phi_2}{I_1} \quad (4)$$

The geometrical dimension, usually induced and thus implicit in graphical representations of networks is just a facility for conceptualisation. It is a deviation of the strict logical representation of social networks in which, most often, there are no geometrical distances between people but only separation degrees<sup>2</sup>. That is the reason why we prefer to use adjacency matrices.

To introduce our new dynamic model, we assimilate the edges of a social graph to conductors transporting electrical flows. We assume that percentages of read, written or shared common documents (*e.g.* office, mails, instantaneous messages), exchanged data packets (ToIp, VoIp) or other numerical communication marks between individuals, are transposed into electrical intensities, tensions, and powers. We apply electromagnetic laws to compute in a predictive way the elements of  $E'$ , set of edges of a graph  $G(V, E)$ , at the end of an indivisible time interval  $[t; t + 1]$ . The figure 1 shows a conceptual illustration of how we use

<sup>2</sup>The separation degree between two vertices is the minimal count of edges connecting them.

our radio-electric analogy to construct an electrodynamic structural evolution of  $E$ , from a time  $t$  to a time  $t + 1$ .

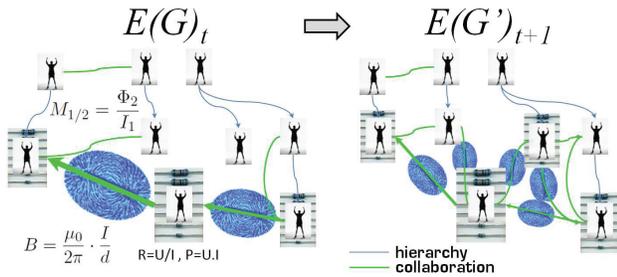


Figure 1: Predictive electrodynamic on structure and flows.

The work of [10] about resistance and currents of finite networks, demonstrating the unity and continuity of flows within large graphs, brings a new hypothesis to be validated in SNA. Finally, superposition of ontologies to these models will add a semantic dimension necessary to a conceptual, qualitative and faceted classification of flows and structures in EISNA.

### 3.2. Semantic SNA

Semantic SNA studies the conceptual aspects of social graphs. It is founded on conceptual graphs and ontologies coupled with SNA principles [4]. Currently, to our knowledge, a few significant work has been published in the domain, but the attractiveness of the subject is visible.

J. JUNG AND J. EUZENAT comment the description of a three-dimensional view of semantic SNA, putting together social graphs, annotations and ontologies [7]. Their proposal overlays and makes the three dimensions coincide in order to build *consensual* ontologies<sup>3</sup> where annotations are linked to the social graph. The first work of [2] about semantic SNA paves the way of semantic and statistic analysis. The purpose of this work is to make the outline of SNA operational, by integrating it to models and languages of the Semantic Web (*i.e.* OWL, RIF, FOAF, SIOC, MOAT, POWDER).

Rules and inferences systems, in correlation with cognitive sciences, bring a main line of SNA development towards a semantic dimension. This development seems to be submitted to vertices and edges annotations, by automatic means such as statistic learning and natural language processing (NLP), or human treatments such as *social tagging*. *Reciprocal evaluation* between members of a social network shows how human interaction produces a valuation on which a reliable *degree of confidence* can be com-

<sup>3</sup>An ontology is an explicit specification of a shared conceptualisation [5].

puted<sup>4</sup>. T. GRUBER cheers on initiatives which tend to integrate semantic web principles and languages to social networks for the development of *Collective Intelligence* and *Collective Knowledge Systems* [4]. From large Web communities to enterprises social networks, semantic SNA can bring real progresses in different domains, such as global marketing linked to globalisation, social and human capital management or work-groups and work-methods optimisation within professional organisations.

## 4. Multidimensional synergies in EISNA

The main objective of our work is to exhibit multidimensional synergies between the static, dynamic and semantic aspects in Enterprises and Institutions Social Networks Analysis - EISNA. The specificities of EISNA are: (1) social graphs composed of up to 100 000 nodes, (2) endogenous data restricted to a few specific and connate domains and (3) intensive collaborative work with trade oriented information sharing.

The methodology we have adopted respects the segmentation of the problematics:

- For dynamic SNA, our contribution mainly consists in providing relevant bridge-building of known methods and identified models, originally from physics or cognitive sciences. The results we provide concern a new predictive model of social graphs, devoted to EISNA. It consists in the application of radio-electricity laws to add or delete edges in a graph  $G(V, E)$  and to set up an electrodynamic simulation of its evolution in a time interval (*cf.* section 4.1).
- Semantic SNA is developed by integrating social graphs, conceptual graph, ontologies and inferences rules. The contributions we provide can only be applied to EISNA and they are specially devoted to work organisation and social/human capital management. Currently, these contributions consist in defining a notion of *tension* of a social network, paired with a new measure of *reactance* which aims to evaluate individual stress (*cf.* section 4.2).

The results we provide are jointly afforded to converge in a multidimensional model, leading to the development of decision-making tools for enterprises and institutions social networks.

### 4.1. Dynamic EISNA, Physics models and Cognition

The objective of our work on dynamic EISNA is to provide a new predictive flows and structures model, where

<sup>4</sup>We talk about *favours network* when the graph structure depends on peer to peer evaluations.

flows are naturally depending on structures and *vice-versa*. An electrodynamic model is proposed for predictive analysis of oriented invisible flows in section 3.1. We introduce a conceptual dimension to refine this model by semantic superposition of social graph individual data, with trades-oriented ontologies extracted from endogenous data of enterprises and institutions content. By the way, we set up a new dynamic and semantic model of EISNA that the figure 2 represents in a conceptual way.

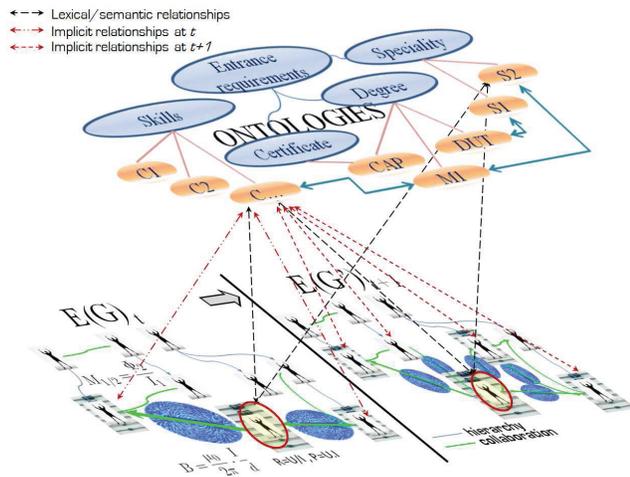


Figure 2: Predictive electrodynamic and linked-data system of EISNA.

We set up trades-oriented ontologies above our predictive and evolutionist system. As shown in figure 2, this approach enables (1) to link moving communities data to semantic networks by a set of explicit associations, and (2) to discover implicit trades-specific knowledge associated to individuals by logical inferences. The domain ontologies integrated to this innovative model bring a semantic dimension, powerful in the context of an exploitation devoted to decision-making functions for human and social capital management.

## 4.2. Semantic EISNA

In sections 3.1 and 4.1, we have introduced an analogy between flows or structures analysis within social networks, and some principles of radio-electricity that seem to be relevant. We have put forward the notions of *resistance*, *charge*, *capacity*, *warm-up* and *powers*. This context is used to characterise implicit or explicit relationships  $R_s(i, j)$  between the vertices of a social graph. Our goal is to cross these relationships with semantic properties (object or data properties) represented by one or more domain ontologies, in order to conceptualise interactions within the social graph.

The notion of reactance already exists in electrodynamic and social psychology. In electrodynamic, the reactance (in Ohms) describes the energy opposed to an alternative current, depending on the kind of element crossed. In psychology, the reactance characterises *a state of negative motivation following a menace (supposed to be real) of individual freedom restriction that is translated into a influence resistance* [1].

In our work, we propose to use the *reactance*  $\Psi$  as a measure of individual stress. From the notion of *tension* introduced in section 3.1, we draw up the following assertions: Let a graph  $G(V, E)$  where vertices of  $V$  are connected by the edges of  $E$ , respecting the following properties:

- Each element  $v$  of  $V$  intrinsically holds coefficients resulting from classical measures of social networks (*cf.* Freeman) or possible refinements.
- $\forall (u, v) \in V$  connected by  $e \in E$ ,  $(u, v)$  intrinsically holds analogical values, computable in  $\mathbb{Q}$ , of *resistance*, *charge*, *capacity*, *warm-up*, *powers* depending on  $V, E$ .
- $\forall e \in E$  assimilated to an uncharacterised flow  $\varpi$ , owning a quantifiable value  $\varphi_\varpi \neq 0$ ,  $e$  is intrinsically described by values of *resistance*, *charge*, *capacity*, *warm-up* and *powers*. For  $e$ ,  $\vec{\varpi}$  or  $\varphi_\varpi$  are measured as a pseudo-tension  $T_e$  or pseudo-intensity  $I_e$ .

From these assertions and the results of experiments managed in the context of the SOCIOPRISE project (*i.e.* a project dedicated to human and social capital management) within trade-oriented organisations, we offer a first set of knowledge dedicated to the identification of individual stress. This set of knowledge can be represented by the following rules and axioms:

- \* **rule 1:**  
If  $CC_u = \frac{charge_u}{capacity_u}$  increases and  $CC_u < 80\%$ , then  $\Psi_u$  increases<sup>5</sup>.
- \* **rule 2:**  
if  $P_u = \frac{resistance_u \cdot intensity_{(e1,u,e2)^2}}{Pmax_u}$  increases and  $P_u \leq 1$ , then  $\Psi_u$  and *warm-up* increases ( $P_u$  represents a used power).
- \* **rule 2 bis (inference learning on rule 2):**  
if *warm-up* increases, then  $\Psi_u$  increases.
- \* **rule 3:**  
if  $P_u$  increases and  $P_u > 1$ , then  $\Psi_u$  decreases,  $Pmax_u$  decreases and *warm-up* quickly increases ( $P_u$  has exceeded  $Pmax_u$ ).

<sup>5</sup>By analogy with electronic power networks, we integrate the notions of minimal charge threshold under which the performance collapses.

- \* **rule 3 bis** (*inference learning on rule 3 and experts supervision*):  
if  $\Psi_u$  decreases and  $warm - up_u$  increases,  
then quick decreasing of  $Pmax_u$  and destruction risk.
- \* **axiom 1** (*inference supervised learning on rule 1*):  
if  $CC_u \leq 0.8$ ,  
then risk to lose socio-professional performances.
- \* **axiom 2** (*inference learning on rule 3 and 3 bis*):  
if  $P_u > 1$ ,  
then risk of psychosocial troubles.
- \* **axiom 3** (*inference supervised learning on axioms 1 + 2 and their premisses*):  
performance optimisation is equivalent to  $CC_u > 0.8$   
and  $P_u \leq 1$ .
- \* **axiom 4** (*learning from symmetry on axiom 3 and his premisses*):  
risk of psychosocial troubles is equivalent to risk of  
loss of socio-professional performances.

From the equations system underlying these rules and axioms, we are currently formalising a scalar measure of reactance  $\Psi_u$ , paired with the notion of tension and dedicated to EISNA. This measure will be useful to optimise workforce and working groups performance by identifying bottlenecks and hotspots related to activities or keywords, in the enterprise or institution social network. It will help to provide recommendations for training plans, development plans, corporate organisation and prevention of psychosocial troubles epidemics.

## 5. Conclusion

The purpose of our work is to define a model of enterprises and institutions social networks analysis (EISNA). The main originality of this model is to integrate the dynamic and the semantic dimension of EISNA. Our current proposal is based on 2 measures and an innovative system defined in the context of a multidisciplinary approach. The new measures are respectively dedicated to evaluation of the *tension* and the *reactance* of a social network  $N$  notions, while the electrodynamic system enables predictions and recommendations the evolutions of  $N$ .

These new measures correlate statistic, dynamic and conceptual dimensions through endogenous resources and scientific interdisciplinarity.

This work is a baseline for the development of new decision-making functions and tools, for psychosocial risk prevention, performances loss risk prevention and social risk prevention in human and social capital management of enterprises and institutions.

From an applicative point of view, our proposal is currently evaluated in the context of an experiment related to the SOCIOPRISE project. From a theoretical point of view, this work is currently in progress towards the integration of static and semantic aspects of EISNA. We plan to use FREEMAN's measures of centrality and intermediarity in order to advocate a static, dynamic and semantic analysis of social networks structures and flows [3]. The main applicative perspective of this approach is to assist the optimisation of work-groups, performance and health at work. The main theoretical perspective is to formalise *a complex and multidimensional model (static, dynamic and semantic) dedicated to enterprises and institutions social network analysis*.

## References

- [1] J.W. Brehm. *A Theory of Psychological Reactance*. New York Academic Press, 1966.
- [2] G. Erétéo, F. Gandon, J. De Santo, and O. Corby. Semantic social network analysis. In *Proceedings of the WebSci'09: Society On-Line, 18-20 March 2009, Athens, Greece*, 2009.
- [3] L.C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40:35–41, 1977.
- [4] R. T. Gruber. Collective knowledge systems: Where the social web meets the semantic web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(1):4–13, February 2008.
- [5] T.R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human Computer Studies*, 43(5/6):907–928, 1995.
- [6] J. Jackson. *Electrodynamique classique*. Dunod, 2001.
- [7] J. Jung and J. Euzenat. Towards semantic social networks. In *ESWC '07: Proceedings of the 4th European conference on The Semantic Web*, pages 267–280, Berlin, Heidelberg, 2007. Springer-Verlag.
- [8] V. Latora and M. Marchiori. Efficient behavior of small-world networks. *Physical Review Letters*, 87(19), 2001.
- [9] J.F. Sowa. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks Cole Publishing Co., 2000.
- [10] C. Thomassen. Resistances and currents in infinite electrical networks. *J. Comb. Theory Ser. B*, 49(1):87–102, 1990.

# Incremental Construction of Topic Hierarchies using Hierarchical Term Clustering\*

Ricardo M. Marcacini and Solange O. Rezende  
Mathematical and Computer Sciences Institute - ICMC  
University of São Paulo - USP - São Carlos, SP, Brazil  
{rmm, solange}@icmc.usp.br

## Abstract

*Topic hierarchies are very useful for managing, searching and browsing large repositories of text documents. The hierarchical clustering methods are used to support the construction of topic hierarchies in a unsupervised way. However, the traditional methods are ineffective in scenarios with growing text collections. In this paper, an incremental method for the construction of topic hierarchies are presented, allowing the update of a topic hierarchy without repeating the clustering process. The experimental results on several benchmark text collections show that our method obtains topic hierarchies with quality similar to traditional non-incremental algorithms.*

## 1. Introduction

The online platforms for publishing digital content has contributed significantly to the growth of several text repositories. Due to the need to extract useful knowledge from these repositories, methods for automatic organization of text collections have received great attention in the literature [7, 5, 9]. The use of topic hierarchies, such as the Yahoo Directory! and Dmoz, is one of the most popular approaches for this organization because they allow users to explore the collection interactively by topics that indicate the contents of the documents available.

The construction of topic hierarchies using supervised methods usually requires an intense human effort to building a model. Moreover, those methods become impracticable for growing text collections. Thus, in these situations, one possible solution is to use unsupervised hierarchical clustering methods, which allow the construction of topic hierarchies in a automatic way.

Hierarchical clustering methods organize a collection text on a hierarchy of cluster and subclusters. The

most generic knowledge is represented by clusters of higher hierarchy levels while their details, or more specific knowledge by clusters of lower levels. Thus, users can explore the text collection in various levels of granularity, finding the desired information more easily and quickly [4].

Unfortunately, most existing methods for text clustering have limitations that affect the construction and maintenance of topic hierarchies [5]. A serious limitation is the fact that the clustering is done statically, i.e., it is necessary that all text documents are available before starting the clustering process. Thus, the construction of a topic hierarchy should be repeated whenever there are changes in the text collection, which greatly increases the computational cost and can make the solution unfeasible in large volumes of data. Another limitation of traditional methods for text clustering is the difficulty in interpreting the results because there are no labels associated with the clusters. In general, it is necessary to apply a specific algorithm for cluster labeling after the hierarchy construction, further increasing the computational cost of the process.

In view of this, this paper presents a incremental method for the construction of topic hierarchies in text collections. Thus, it is possible to update a topic hierarchy in a growing text collection without repeating the clustering process. The method is based on hierarchical clustering of terms (words) that finds similar terms using the document distribution in which they occur. The clusters of similar terms are used as labels for the set of retrieved documents, helping the interpretation of results. The experimental evaluations demonstrate that the proposed method presents good results and provides a competitive alternative for building dynamic topic hierarchies with understandable description of the document clusters.

To describe the proposed method, this paper is organized as follows. In Section 2, a brief review of concepts about text clustering and the well-know algorithms in the literature is provided. In Section 3, the incremental method for hierarchical term proposed in this work is presented.

\*This work was sponsored by FAPESP and CNPq.

An experimental evaluation using several benchmark text collections is carried out and the results are discussed in Section 4. Finally, in Section 5, an overview of the work is presented along with a discussion of some directions for future work.

## 2. Background and Related Works

In this work, we consider the automatic construction of topic hierarchies using non-supervised hierarchical text clustering.

To perform a process of text clustering it is necessary to define a representation model of textual data, a similarity measure among the documents and a strategy for the cluster formation [4]. The space-vector model [10] is the most used for the representation of texts. In this model, each text document  $d$  is represented by a vector of terms,  $d = (t_1, \dots, t_m)$ , where each term  $t_i$  has an associated value, for example, the frequency of occurrence. In order to obtain the similarity between two documents the cosine measure is commonly used. Given vectors of two documents  $d_i$  and  $d_j$ , the cosine measure is defined as follows:

$$\cos(d_i, d_j) = \frac{d_i \cdot d_j}{\|d_i\| \|d_j\|} \quad (1)$$

Finally, we define a clustering strategy. The agglomerative hierarchical clustering methods are widely used, and start by considering each document as a separate cluster. Then, the pairs of the most similar documents are merged iteratively until all the documents belong to only one cluster. There are three classic algorithms of this strategy: Single-Link, Complete-Link and UPGMA [2].

The Single-Link is one of the simplest hierarchical clustering algorithms and uses the technique of the nearest neighbor, where the similarity of two clusters is the similarity of their most similar members. Unlike the Single-Link, in the Complete-link clustering the similarity of two clusters is the similarity of their most dissimilar members. In the UPGMA algorithm, the similarity between two clusters is defined as the average of the similarities between all pairs of documents in each cluster. The UPGMA eliminates many problems related to dependence on the size of the cluster, and is considered one of the best algorithms in textual collections in terms of clusters quality [12]. The clusters obtained by these three algorithms do not have labels to describe their content, doing it hard to interpret the results.

In recent years, alternative methods have been proposed for text clustering in order to obtain clusters with associated labels [3, 5, 7]. These methods aim the clustering of the most similar terms (words) in the collection, ie, terms with some value of co-occurrence. Each cluster of terms has a set of documents associated, resulting in document

clusters with labels. The FIHC [5] is one of such method, and uses the Apriori algorithm [1] to find the terms with co-occurrence in the text collection, called frequent itemsets. The text collection is organized hierarchically and the frequent itemsets are used as cluster labels. One disadvantage of the FIHC algorithm is that the parameters used to find the frequent itemsets are difficult to define in practice [2].

The method proposed in this paper also performs the document clustering using the co-occurrence between terms. A technique called co-occurrence graphs [11] is used, where terms are organized in a graph and two terms are connected if there is a significant co-occurrence value between them. From the graph, the similarity between two terms,  $t_i$  and  $t_j$ , is calculated by the SNN (Shared Nearest Neighbor) measure:

$$SNN(t_i, t_j) = \frac{|N(t_i) \cap N(t_j)|}{|N(t_i) \cup N(t_j)|} \quad (2)$$

where  $N(t)$  represents the set of neighbors of  $t$ . This similarity measure uses only the topology of the graph, and has been considered very useful when trying to identify topics in a given corpus. In our method, the Incremental Hierarchical Term Clustering (IHTC) algorithm is presented, which constructs the co-occurrence graph and clustering similar terms in an incremental way. The hierarchical term clustering is then used for the construction and management of topic hierarchies.

One focus of this paper is to evaluate the topic hierarchies generated by the incremental algorithm proposed and to compare the results with the traditional non-incremental methods.

## 3. Incremental Hierarchical Term Clustering

This section presents an incremental method for constructing topic hierarchies based on a hierarchical term clustering. The text collection is represented in a graph, where the vertices are terms from the text collection and the edges indicate significant co-occurrence between them. The most similar terms are clustered using the SNN measure, resulting in a hierarchical structure known as dendrogram. To illustrate, consider the co-occurrence graph in Figure 1. A possible result for the hierarchical term clustering is the dendrogram in the Figure 2.

The dendrogram is a binary tree that represents the sequence of the clustering and the similarity between the terms. Each node of the dendrogram is a possible topic of the collection, represented by a set of terms. The set of documents of a topic is obtained by the union of subsets of documents retrieved by each term in the topic. This structure allows a user to explore the textual collection by browsing the topics that describe the information of interest.

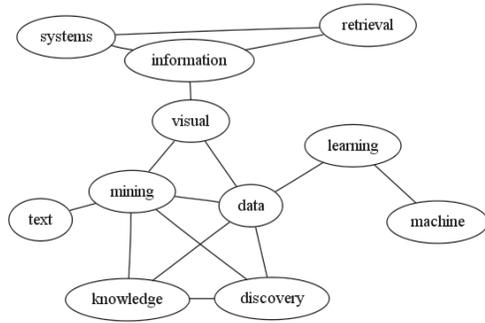


Figure 1. Co-occurrence term graph

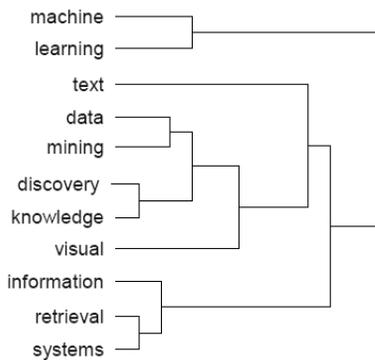


Figure 2. Dendrogram representation of the hierarchical term clustering

Below, we present the algorithm IHTC, which implements the proposed method. The algorithm performs texts preprocessing, constructs a graph of co-occurrence of terms and their related dendrogram. This whole process is performed incrementally.

### 3.1. Incremental Text Preprocessing and Clustering

Textual collections can easily contain thousands of terms, many of them redundant and unnecessary, that slow the clustering process and decrease the quality of results. In our method, the goals of the text preprocessing is to maintain a subset of representative terms and organize them in a co-occurrence graph.

Each document inserted in the text collection is pre-processed individually. Thus, for a new document  $d$ , the stopwords are removed and a stemming technique applied to transform the terms in its primitive form. Finally, the most significant terms of the document using a technique of extracting keywords are selected [3]. In this work, we used a simple technique where the most frequent  $k$  terms are selected.

The terms selected from document  $d$  should be added in

the co-occurrence graph  $G$  of the text collection. However, computing the co-occurrence between the terms for each new document using a brute force approach, would make the process very expensive computationally. Thus, we propose an approximate method to obtain the pairs of terms with significant co-occurrence, based on a technique called Top-K Frequent Elements [8]. In this technique,  $m$  is the number of counters that monitor the occurrence of pairs of terms. For each pair of terms  $e = \{t_i, t_j\}$  presented, If the pair is already monitored, then the value of its counter is incremented by 1. Otherwise, the pair is associated with a free counter and receives the initial value 1. If there aren't free counters, the new pair  $e$  replaces the pair  $e_{min}$  that has the lowest counter  $F_{min}$ . The value for the counter  $e$  is updated with  $F_{min}+1$ , in order to allow the emergence of new pairs.

When the value of a counter is increased, an event to update the co-occurrence graph is called. Then the pair of terms is inserted into the dendrogram, which is updated dynamically in order to maintain the correct values of similarity between the clusters terms. An overview of IHTC algorithm is illustrated in Figure 3. The process of updating the dendrogram is detailed as follow.

#### Algorithm IHTC

Parameters:

S: TextSource, G: Graph, H: Dendrogram, List: Counters

```

for each new document  $D$  in  $S$ 
  Stopwords Removal
  Stemming
  TermSet = getKeywords( $D$ )
  for each different pair of term  $e = \{t_i, t_j\}$  in TermSet{
    if  $e$  is monitored {
      Increment the counter of  $e$ 
      updateGraph( $e, G$ )
      updateDendrogram( $e, H$ )
    } else {
      if there are available counters
        Assign counter for  $e$  with value equal to 1
      else {
        Let  $e_{min}$  be the element with least value,  $F_{min}$ 
        Replace  $e_{min}$  with  $e$  and assign  $F_{min}+1$  for the counter of  $e$ 
      }
    }
  }
}

```

Figure 3. Overview of the IHTC algorithm

### 3.2. Update Dendrogram Tree

The structure of the dendrogram has important information about the hierarchical clustering. The height of the nodes that merge two clusters indicates the similarity between them. The lower the height, the more similar the clusters. In a valid dendrogram, it is necessary that the height of the parent clusters is greater than or equal to that of their children. This property is exploited in the construction and dynamic update of the dendrogram.

When a pair of terms  $t_i$  and  $t_j$  is presented, the value of similarity between them  $sim = SNN(t_i, t_j)$  is computed. Thus, we need to update this new value of similarity in the dendrogram. There are four possible scenarios for this update:

1. If  $t_i$  and  $t_j$  does not exists in the dendrogram, that is, both are new examples, then a new node with similarity equal to  $sim$  is created in the dendrogram, merging  $t_i$  and  $t_j$ ;
2. If  $t_i$  exists in the dendrogram and  $t_j$  is a new example, then a new node with the similarity equal to  $sim$  is created in the dendrogram, merging  $t_j$  and the ancestor of  $t_i$ ;
3. If  $t_i$  is a new example and  $t_j$  exists in the dendrogram, then a new node with the similarity equal to  $sim$  is created in the dendrogram, merging the ancestor of  $t_j$  and  $t_i$ ;
4. Finally, if both  $t_i$  and  $t_j$  exists in the dendrogram and there is no common ancestor between them, then a new node with the similarity equal to  $sim$  is created, merging the ancestors of  $t_i$  and  $t_j$ . Otherwise, the similarity of the first common ancestor between  $t_i$  and  $t_j$  is updated to  $sim$ , if  $sim$  is greater than the current value of this ancestor.

Scenarios 2, 3 and 4 can make the dendrogram invalid, i.e., the value of similarity of a cluster parent is greater than the similarity of a cluster child. In this case, a correction procedure is executed to correct the node  $N_{new}$  responsible for the state invalid: (i) remove  $N_{new}$  from dendrograma; (ii) promote the child node with lowest similarity ( $N_{child1}$ ); (iii) the other child node ( $N_{child2}$ ) must be reinserted in the dendrogram. If  $N_{child2}$  is ancestor of  $t_i$  then  $N_{child2}$  is inserted as child of the first ancestor of  $t_j$  that has similarity greater than  $N_{child2}$ . Otherwise,  $N_{child2}$  is inserted as child of the first ancestor of  $t_i$  that has the similarity greater than  $N_{child2}$ . Thus, the dendrogram back to its valid state and the terms remain linked according to their similarity values. This updating process is detailed in the Figure 4.

To exemplify this process, in Figure 5 is illustrated an example of dendrogram updating. In (a) there is a valid dendrogram with 5 terms: A, B, C, D and E. Next there is an updating in the co-occurrence graph and the similarity between the terms B and C is updated to 0.7 (b). Thus, the similarity of the node {ABCD}, first common ancestor of B and C, is updated to the value 0.7. After this update, the dendrogram is in an invalid state, because the value of similarity of the parent node {ABCD} is greater than the value of the child node {AB}. In (c), the correction procedure is executed on the node {ABCD}, where the child node with lowest similarity {AB} is promoted and the

```

Algorithm IHTC::AdjustDendrogram
Parameters
  TermPair: e={ti,tj}, Node: Nnew, H: Dendrogram

Nchild1 = child of Nnew with lowest similarity;
Nchild2 = child of Nnew with highest similarity;
sim = Nnew.similarity();
Remove(Nnew);
Promote(Nchild1);

if( Nchild2.contains(ti) ) starting = tj;
else starting = ti;

newParent = starting.getParent();
while( newParent.getParent() != null ){
  if(newParent.similarity() < sim) break;
  newParent= newParent.getParent();
}
newParent1 = child1 of newParent;
newParent2 = child2 of newParent;
if(newParent1.contains(starting)) brother = newParent1;
else brother = newParent2;

/* Reinserting Nchild2 */
newParent.removeChild(brother);
n = new dendrogramNode();
n.insertChilds(brother, Nchild2);
n.setSimilarity(sim);

newParent.insertChild(n);
AdjustDendrogram(n, e);

```

Figure 4. Dendrogram Adjustment

child node {CD} is reinserted. In (d), the dendrogram is in a valid state again.

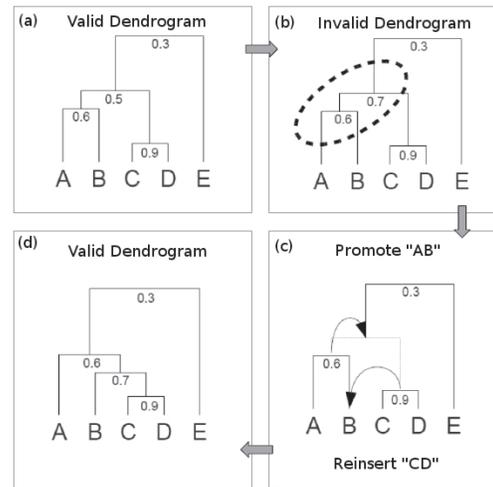


Figure 5. Example of Updating Dendrogram

The updates of the dendrogram is based on the nearest neighbors strategy, like the Single-Link. However, as only pairs of terms that co-occurrence has increased are updated, the resulting dendrogram is an approximation,

trying to focus on pairs of terms with co-occurrence more significant. The order of insertion of the terms can change the final structure of the dendrogram. The experimental evaluation shows that the algorithm is robust, i.e., the order of presentation of terms does not affect the quality of results significantly.

The next section presents an evaluation of the IHTC in benchmark text collections, and we compared the results with other algorithms in the literature.

#### 4. Experimental Evaluation

The IHTC was evaluated experimentally in 8 textual collections. These collections have predefined topics, allowing to measure the precision and recall of topic hierarchies and make comparisons with other algorithms in an objective way.

Table 1 presents a summary of the text collections used in experimental evaluation. The collections *la1*, *la2*, *tr31* and *tr41* were obtained from the repository of TREC (Text Retrieval Conference) and the documents are newspaper articles. The collections *k1b* and *wap* were provided by the WEBACE project, and consist of documents from Yahoo! Directory. Finally, the collections *re0* and *re1* were obtained from Reuters-21578. All collections are available for download in the site of CLUTO project [12].

**Table 1. Summary of Textual Collections**

Textbase	Source	# Docs	# Terms	# Topics
k1b	WebACE	2340	13879	6
la1	LA Times (TREC)	3204	21604	6
la2	LA Times (TREC)	3075	21604	6
re0	Reuters-21578	1504	2886	13
re1	Reuters-21578	1657	3758	25
tr31	TREC	927	10128	7
tr41	TREC	878	7454	10
wap	WebACE	1560	8460	20

The results obtained with the IHTC were compared with algorithms Single-link (SL), Complete-Link(CL), UPGMA and FIHC. Some details of the experiment are described as follow.

- In the experiments with the algorithms SL, CL, FIHC and UPGMA, textual collections were preprocessed with stopwords removal, stemming and selecting only the terms that occur in two or more documents. The IHTC uses a particular technique for text preprocessing.
- For the algorithms SL, CL and UPGMA, the collections were represented using the vector-space model and the hierarchical clustering obtained with the cosine measure.

- The algorithm FIHC have two parameters to obtain the frequent itemsets. We selected the best parameters for each text collection, performing several runs of the algorithm.
- For the algorithm IHTC, we use  $k=20$  most frequent terms of each document. The co-occurrence graph was constructed using  $m=1000$  counters to monitor the occurrence of pairs of terms. These parameters were used for all textual collections. Parameter values were obtained by preliminary experiments and used in all textual collections.

We obtained topic hierarchies for each text collection, using the algorithms IHTC, SL, CL, UPGMA and FIHC. The hierarchies were evaluated with the FScore measure [6, 12], which treats each cluster as the result of a query and each predefined topic as the set of relevant documents of this query. Thus, it is possible to calculate values of precision and recall of each cluster of the hierarchy. In general, the higher the value of the FScore, the better the quality of topic hierarchy solution.

The Table 2 shows the values FScore obtained by each method in each text collection. The FScore values of the IHTC algorithm was estimated by the average value of 100 different runs for each collection, to measure the effect of the order of presentation of data.

**Table 2. The FScores of the topic hierarchies**

	IHTC	SL	CL	FIHC	UPGMA
<b>k1b</b>	0.729 ± 0.014	0.655	0.764	0.750	0.892
<b>la1</b>	0.526 ± 0.021	0.369	0.364	0.453	0.654
<b>la2</b>	0.535 ± 0.022	0.365	0.449	0.489	0.709
<b>re0</b>	0.649 ± 0.006	0.465	0.495	0.693	0.584
<b>re1</b>	0.685 ± 0.017	0.445	0.508	0.472	0.695
<b>tr31</b>	0.783 ± 0.018	0.532	0.804	0.765	0.816
<b>tr41</b>	0.691 ± 0.021	0.674	0.758	0.713	0.826
<b>wap</b>	0.536 ± 0.017	0.435	0.569	0.561	0.640

The results indicated that our algorithm obtained topic hierarchies with quality similar to existing algorithms in the literature. In some cases, the results were better. Table 3 shows the comparison between IHTC and the other four algorithms. The Rank Difference column shows the difference between the algorithms based on the ranking of the FScore values. Similarly, the Mean Difference column shows the difference based on mean FScore of each algorithm. Positive differences indicate advantage of our algorithm against the algorithm compared and negative differences show when our algorithm obtained inferior performance.

The Rank and Mean Differences allow the application of the statistical tests of Friedman (nonparametric) and SNK (parametric), respectively. These tests analyze whether the differences in results are statistically significant. The value

“NO” indicates that there is no evidence to ensure that the compared algorithms are different, otherwise the value would be “YES”. Both tests were performed with 95% confidence level (p-value = 0.05).

**Table 3. Statistical analysis of the results**

	Rank Difference	Friedman Test	Mean Difference	SNK Test
IHTC x SL	16	NO	0.15	YES
IHTC x CL	1	NO	0.05	NO
IHTC x FIHC	0	NO	0.02	NO
IHTC x UPGMA	-13	NO	-0.09	YES

Based on the Friedman test, it is not possible to affirm that the our algorithm has better results than the other four algorithms. However, the SNK test indicates that our algorithm is more efficient than the SL but inferior than UPGMA. There is no evidence to indicate that the IHTC performance is superior or inferior than the CL and FIHC algorithms.

As expected, the UPGMA algorithm showed the best FScore values. However, this algorithm is not scalable to large databases [12], because of its high complexity  $O(n^3)$ . The FIHC obtained good results, but the computational cost is determined by the time of generation of frequent itemsets using the Apriori, which is usually high in textual data. The SL and CL algorithms, with  $O(n^2)$  complexity, had inferior results than our algorithm in most textual collections. The complexity of the proposed IHTC algorithm in the worst case is  $O(n \times k \times m \times t^2)$ , where  $k$  and  $m$  are the parameters of the algorithm and  $t$  the number of different terms in the co-occurrence graph. Moreover, the number of terms of a text collection converge to a constant number when the number of documents is large [5]. Thus, the complexity of IHTC is linear according to the number of documents and scalable in large textual collections.

Based on this experimental evaluation, the IHTC algorithm is an attractive alternative because it provides competitive results and allows the construction of topic hierarchies in a incremental way, with labels to facilitate the interpretation of the results.

## 5. Conclusions and Future Works

In this paper, we presented an incremental method for the construction of topic hierarchies using hierarchical term clustering. We introduced the algorithm IHTC that is able to preprocess the textual collections, obtain co-occurrence graphs and a dendrogram with the terms of the collection. The results obtained with the IHTC is comparable to traditional off-line clustering algorithms, however, our incremental algorithm has the ability to process growing text collections. Moreover, the proposed algorithm is

scalable because it has linear time complexity when the number of documents is large. We provide the IHTC source code, runtime performance analysis, and a online demonstration of the IHTC in the our project page<sup>1</sup>.

In the future, we plan to improve our evaluation method. For example, we will evaluate the effect of the parameters  $k$  and  $m$  used in the IHTC. Moreover, we intend to compare the IHTC with other incremental algorithms, commonly used in text streams.

## References

- [1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceeding of 20th International Conference on Very Large Data Bases, VLDB*, volume 1215, pages 487–499, 1994.
- [2] B. S. Everitt, S. Landau, and M. Leese. *Cluster Analysis*. Arnold Publishers, 2001.
- [3] R. Feldman, M. Fresko, Y. Kinar, Y. Lindell, O. Liphstat, M. Rajman, Y. Schler, and O. Zamir. Text mining at the term level. In *Proceedings of the PKDD'98*, pages 65–75. Springer Verlag, 1998.
- [4] R. Feldman and J. Sanger. *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press, 2006.
- [5] B. Fung, K. Wang, and M. Ester. Hierarchical document clustering using frequent itemsets. In *Proceedings of the SIAM International Conference on Data Mining*, 2003.
- [6] B. Larsen and C. Aone. Fast and effective text mining using linear-time document clustering. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 22. ACM, 1999.
- [7] Y. Li, S. Chung, and J. Holt. Text document clustering based on frequent word meaning sequences. *Data & Knowledge Engineering*, 64(1):381–404, 2008.
- [8] A. Metwally, D. Agrawal, and A. El Abbadi. Efficient computation of frequent and top-k elements in data streams. In *Proceedings of the 10th ICDT International Conference on Database Theory*, pages 398–412. Springer, 2005.
- [9] M. F. Moura, R. M. Marcacini, B. M. Nogueira, M. da Silva Conrado, and S. O. Rezende. A proposal for building domain topic taxonomies. In *Proceedings of International Workshop on Web and Text Intelligence - 19th SBIA Brazilian Symposium on Artificial Intelligence (SBIA)*, pages 83–84. São Carlos, Brazil, 2008.
- [10] G. Salton. *The SMART Retrieval System—Experiments in Automatic Document Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1971.
- [11] R. Sole, B. Murtra, S. Valverde, and L. Steels. Language Networks: their structure, function and evolution. *Trends in Cognitive Sciences*, 12(42):343–352, 2005.
- [12] Y. Zhao and G. Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *Proceedings of the 20th International Conference on Information and knowledge management*, pages 515–524. ACM New York, NY, USA, 2002.

<sup>1</sup><http://sites.labc.icmc.usp.br/marcacini/ihtc/>

# A Framework for Detecting Code Piracy Using Class Structure

Patrice Arruda, Pierre Chamoun, Dr. Dwight Deugo

*School of Computer Science, Carleton University  
Ottawa, Ontario, Canada*

*Email: patrice.arruda@gmail.com, pierrechamoun@hotmail.com, deugo@scs.carleton.ca*

## Abstract

*Open Source Software and Intellectual property violations are becoming a field of study. A new approach was devised to detect such violations and determine similarities between two JAR files. The main focus is based on the structure of the classes, more precisely the Class Dependency and Class Logic. A framework, JAR Piracy Detector (JDP), has been developed to help analyze these two methods. JPD is composed of two main operations: generate and compare. The generate operation generates a unique JAR file fingerprint, composed of particles (such as class dependency and class logic particles) where the compare operation compares two fingerprints. Each particle performs its own analysis to obtain a matching certainty percentage. The class dependency devises a graph from the dependency of classes in a JAR file and compare to another formed graph from a different JAR file. The class logic counts and compares the repetition of logical keywords, such as 'if' and 'while', in every class found in two different JAR files. Each particle returns an approximation percentage matching describing how identical are the two fingerprints. Several experiments were performed and showed that the class structure analysis can lead to a certainty that some violation has occurred.*

## 1. Introduction

Many organizations and governments institutions are becoming more involved in the Open Source Software (OSS) community. Software practices have been changing for the last several years to reduce development expenses and accommodate new technologies. Licensing violations and code piracy are problems that could arise when using open source software. JAR Piracy Detector (JPD) helps detecting these types of problems.

## 1.1. Problem

Several clone detection software techniques focus greatly on the content of each class individually [4] and not on the structural relation of the class. We believe that those copying software either ignore or can't change the structural relations between classes. Therefore these relations can be used to provide a unique signature to detect software clone.

## 1.2. Motivation

Our motivation is to provide a starting point for forensics analysis in detecting software piracy in JAR files, and to provide advance evidence that IP has been violated. There are cases where organizations use OSS without the consent of the original contributor. There is also a possibility that an abusive user could damage the reputation of highly reliable software by including pirated code. Therefore the detection of IP violations in the software we develop and use is very important.

## 1.3. Goals

Our main goal is to find structural and logical dependencies between two sets of classes that represent two fingerprints. The framework, JDP, will facilitate the ability:

- 1) To generate a unique fingerprint for each JAR file.
- 2) To perform comparison between two fingerprints.
- 3) To calculate the certainty percentage of the approximate matching between two fingerprints.
- 4) To add and remove particles from a fingerprint.

## 1.4. Objectives

From our set of goals, our main objectives are to generate a small fingerprint size for comparison with

speed and accuracy. More specifically:

- 1) Avoid large fingerprints by generating essential, compressed data.
- 2) Introduce new approaches, such as class dependency and class logic, to improve the accuracy of the comparison operation.
- 3) Compare only crucial information extracted from the JAR file.
- 4) Increase the accuracy of the comparison by adding or removing particles.

## 1.5. Outline

This paper begins with the background of issues raised in comparing two graphs. Section 3 describes our approach to the proposed problem in 1.1. The section explains the design detail of the framework and the approach used to generate and compare two fingerprints based on class dependency and logic. Section 4 lists and discusses the obtained results by performing experiments on a number of JAR files. This paper concludes in Section 5 and additional future work is discussed in Section 6.

## 2. Background

Extensive research has been conducted to find proper tools to generate and analyze class dependency. The typical approach was to analyze software that can generate Unified Modeling Language (UML) models from compiled classes. The main issue of using UML was a lot of unnecessary data was generated, where we are only interested in essential data. We simplified the UML generation process by collecting specific references between classes. These references can form a graph that can be later converted to a relationship matrix (relationship matrix is a unique type of incidence matrix [1] where the entries represent the edge type between two nodes).

Comparison between two graphs has been determined to be NP-Complete [6]. Our approach uses approximation to find the best-matching subgraph between two graphs. This approximation algorithm can produce misleading results where one node in the first graph can match multiple nodes in the secondary graph. To reduce the misleading results, the class logic was introduced to confirm the matching of two nodes.

## 3. Approach

In this section, we explain in detail the design of the framework and the approach used to analyze two fingerprints and how the certainty percentage is calculated.

## 3.1. Program's Architecture, Operations, and Behaviour

The modularity feature of the framework made it efficient and easy to add new modules (also known as particle) to perform different types of analysis. For example if the user is interested in analyzing the byte code of each method [3], one simply creates a new particle and add it to the framework.

A well defined particle is composed of two very important operations: generate and compare. The generate operation is the method of taking a JAR file and producing a fingerprint according to the algorithm defined in the particle. The compare operation takes a previously generated fingerprint particle and compares it to another fingerprint particle of a different JAR file. In essence, a fingerprint is made up of concatenated fingerprint particles.

The modularity feature is based on the Observer design pattern [5]. Each particle represents an observer where the Fingerprint Generator is the subject. The Fingerprint Generator is a component of the framework that generates a unique fingerprint signature of a JAR file. It also compares with other fingerprints. When the fingerprint generator receives a request to generate a fingerprint, the generator will notify each subscribed observer to generate its own particle data. The particle receives the location of the JAR file, goes through a series of computational steps, and a particle fingerprint signature is generated and appended to the fingerprint. The data of the fingerprint is represented using XML format, for ease of parsing.

The fingerprint comparison operation is quite similar to the fingerprint generation process. The fingerprint generator receives a request to compare a fingerprint to a JAR file. Each particle calls its generate method to compile its own particle data. Afterwards the two particles are compared to produce a certainty percentage. The certainty percentages that are collected from the particles are then combined to form an overall certainty percentage average.

To show the modularity of the framework, three more simple particles have been included on top of the class dependency and class logic; to help increase the accuracy of the overall certainty percentage.

**3.1.1. Preparation Step.** Before each particle generates its own fingerprint, the framework will first unarchive the JAR file and save its contents in a specified temporary directory. Afterwards, the extracted classes are decompiled using an open source tool called JClazz [2]. The framework has the ability to incorporate other decompilers in case one decompiler

**Input**  
V: List of values  
C: List of classes  
c1,c2: Classes in C  
i, e, d: Symbol denotes that class1 implements, extends, depends class2 respectively.

**Output**  
V: List of values

```

GENERATE-CLASS-DEP-PARTICLE()
1 for each c1 in C[m]
2 do
3   for each c2 in C[n]
4   do
5     if c1 != c2
6   then if c1 is an interface in c2
7     then Add String(c2,i, c1) to V
8   else if c2 extends c1
9     then Add String(c2,e, c1) to V
10  else s<-source of c2
11    if FIND-CLASS-IN-FILE(s) = found
12      then Add String(c2,d, c1) to V
13  REMOVE-SMALL-GRAPHS(V)

```

**Figure 1:** An algorithm to generates a class dependency fingerprint.

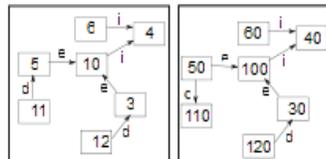
is not able to carry out the operation. The information of each decompiled class is stored in the ClassDataInfo object for later use by particles. The ClassDecompiler is implemented as a singleton pattern[6] where it holds a list of ClassDataInfo objects.

### 3.2. Fingerprint Generator

As mentioned in Section 3.1, the fingerprint generator generates a unique fingerprint for each JAR file. The fingerprint is composed of a list particles' data, generated from each of the registered particles. Even though we implemented Filename, Filesize and Image particles to increase the accuracy of the overall certainty percentage, this paper concentrates on the class dependency and class logic.

**3.2.1. Class Dependency Particle.** The first and the most important particle of all is the class dependency particle. This particle constructs a graph  $G = \langle V, E \rangle$ , where each node  $v \in V$  represents a class and each  $e \in E$  represents a dependency between two nodes. We determined that there are three types of dependencies needed for this analysis which are implements, extends, and references. See the algorithm in Figure 1 for generating a class dependency fingerprint. The algorithm ignores small graphs of size less than five nodes since small graphs could cause misleading results during the comparison operation.

**3.2.2. Other Particles.** Class logic particle represents the number of times a logical keyword appeared in a class file. Some of the keywords are  $A =$



**Figure 2:** An example of a class dependency graph of a very small JAR file.  $N1 = \{5, 10, 11, 3, 12, 4, 6\}$

*if, else, while, do, for, ...*. The results are stored in a list for later use in the comparison operation.

As we mentioned earlier, the image particle focuses on images found in a JAR file. For each image, grab the first 100 bytes and hash it using the MD5 algorithm. We decided to extract the first 100 bytes for simplicity and MD5 them. A list of hashed strings represents the image fingerprint particle.

The most simplistic of all are the filename and file-size particles. The filename particle forms the particle data from a list of filenames found in the JAR file. The same applies to the file size particle, except each file size is extracted.

### 3.3. Fingerprint Comparison

The framework is able to accept a generated fingerprint and a JAR file to perform the comparison operation. Each particle generates a fingerprint particle data out of the JAR file, and compare that with the corresponding fingerprint particle, inside of the inputted fingerprint. In the next section, we are going to explain in detail how each particle executes the compare operation.

**3.3.1. Class Dependency Particle.** The class dependency particle comparison is an approximation algorithm since the problem falls under the NP-Complete category [6]. First a graph  $G1$  is generated out of the inputted fingerprint particle data, and another graph  $G2$  is generated out of the JAR file using the generate operation of the particle. Figure 2 provides an example of a class dependency graph.

The graphs  $G1$  and  $G2$  are converted to Relationship Matrices  $M1$  and  $M2$ , as shown in (see Figure 3). Each row and column in the relationship matrix represents the nodes of a graph. Each entry in the matrix is assigned with one type of dependency edge:  $d, e, i$ . Edge  $d$  represents a class making a reference to another class. Edge  $e$  is a class that is extended by another class. Edge  $i$  is a class that implements an interface. Each dependency edge has a number 1 or 2 associated appended to the edge type letter. For

$$M1 = \begin{bmatrix} 0 & e1 & d2 & 0 & 0 & 0 & 0 \\ e2 & 0 & 0 & e2 & 0 & i1 & 0 \\ d1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & e1 & 0 & 0 & d2 & 0 & 0 \\ 0 & 0 & 0 & d1 & 0 & 0 & 0 \\ 0 & i2 & 0 & 0 & 0 & 0 & i2 \\ 0 & 0 & 0 & 0 & 0 & i1 & 0 \end{bmatrix}$$

**Figure 3:** An example of a relationship matrix.  $N1$  contains a list of class ID. Each row or column of  $M1$  represents the class ID  $N1[i]$ .

$$SM1 = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

**Figure 4:** An example of a Row Entry Summation matrix.

example,  $d1$  means that class  $X$  depends on class  $Y$ .  $d2$  represents that class  $Y$  is depended on class  $X$ . The same implies for  $e$  and  $i$ .

Once the relationship matrices have been generated for both particles, the next step is to create two small Row Entry Summation Matrices  $SM1$  and  $SM2$  that corresponds to  $M1$  and  $M2$  respectively (see Figure 4). The columns of  $SM1$  and  $SM2$  are the graphs edge types  $d1, d2, e1, e2, i1, i2$  where each entry in the row represents the number of the occurrence of a specific edge type in  $M1$  and  $M2$ . Here is a devised mathematical formula:

$$S = \{d1, d2, e1, e2, i1, i2\}$$

$$M : n \times n$$

$$SM : n \times 6$$

$$SM[i][r] = \sum_{j=0}^n m[i][j], \text{ if } m[i][j] = r, r \in S$$

After the  $SM$  matrices are created, the next step is to try to find identical rows in  $SM1$ , and then find the same identical rows in  $SM2$ . The matching algorithm is shown in Figure 5. For all matching rows, take a row in  $M1$  and a row in  $M2$  and walk in the two matrices; for as long as the nodes match. The walk algorithm is shown in Figures 6 and 7. Repeat this walk for every matching row in  $M2$  and save the longest sequence. This sequence represents the best matching nodes from  $G1$  and  $G2$ . The total number of best matches between  $M1$  and  $M2$  constitutes the certainty percentage of class dependency comparison between the two fingerprint particles.

**3.3.2. Other Particles.** The comparison process for Class Logic, File size, File name, and Image particles

#### Input

$MS1, MS2$ : Matrix that contains the sums of each edge type from  $M1$  and  $M2$  respectively  
 $V$ : Visited rows,  $v$ : Size of  $V$   
 $s$ : Number of rows in  $M1$   
 $B$ : Best nodes

#### Output

$nm$ : Number of matches

#### MATRIX-COMP()

```

1 while (true)
2   do incr m
3   if m not in V
4   then
5   if m > s
6   then exit
7 else
8   Q <- FIND-MATCHING-ROWS-SAME-MATRIX(MS1, m)
9   V <- add Q
10  S <- FIND-MATCHING-ROWS-OTHER-MATRIX(Q, MS2, m)
11  B <- FIND-BEST-MATCHES(Q, S)
12  for each entry in B
13  do
14    if v > VISITED_OFFSET_THRESHOLD
15    then incr matchingsOffset
16    else incr nm

```

**Figure 5:** The algorithm to find the matches between two matrices.

#### Input

$Q$ : Indices of rows that match each other in the first matrix  
 $S$ : Indices of rows that match the rows in the second matrix  
 $B$ : Best matches,  $V$ : Visited entries

#### Output

$B$ : Best matches,  $V$ : Visited entries

#### FIND-BEST-MATCHES(Q, S)

```

1 for each entry e in Q[i]
2 do
3   for each entry v in S[j]
4   do
5     V = BEST-MATCH(e, v, true)
6 n <- v/2;
7 Add largest of V to B

```

**Figure 6:** The algorithm to find the best matches in two graphs.

all follow the same logical comparison algorithm. The logical process is as follows: let's denote  $L1$  and  $L2$  to be the list of elements in each compared particle where an element could be a filename, filesize, MD5, or an entry of class logic (see Section 3.2.2). Initialize a matching counter  $c$  to zero. The particle will take an element from  $L1$  and check if there is the same element in  $L2$ . Successful matching will increment  $c$ . After processing all the elements of  $L1$ , the particle returns the certainty percentage of  $(c/size(L1)) * 100\%$ .

### 3.4. Decisions Made

During the design and development of the framework, several decisions were taken into account. The first issue was the type of format in which the fin-

### Input

M1, M2: First and Second matrices respectively  
R1, R2: Row or column in M1 and M2 respectively.  
i1, i2: Indices of a row or a column in R1 and R2 respectively.  
e: Entry from R1

### Output

V1: Visited entries

```
BEST-MATCH(i1, i2, isRow)
1 if isRow = true
2 then R1<-M1 at row i1
3 R2<-M2 at row i2
4 isRow<-false
5 else R1<-M1 at col i1
6     R2<-M2 at col i2
7     isRow<-true
8 i1 <- GET-FIRST-ENTRY(R1);
9 if i1 = notfound
10 then exit
11 e<-GET-ENTRY(R1, i1)
12 i2<-FIND-FIRST-ENTRY(R2, e)
13 if i2 = notfound
14 then exit
15 r <- GET-ROW(M1, e)
16 c <- GET-COL(M1, e)
17 SET-ENTRY-AS-VISITED(V1, M1, r, c)
18 SET-ENTRY-AS-VISITED(V1, M1, c, r)
19 SET-ENTRY-AS-VISITED(V2, M2, r, c)
20 SET-ENTRY-AS-VISITED(V2, M2, c, r)
21 BEST-MATCH(i1, i2, isRow)
22 if isRow = true
23 then isRow<-false
24     i1 <- GET-INDEX-OF-COL(M1, i1)
25     i2 <- GET-INDEX-OF-COL(M2, i2)
26 else
27     isRow = true
28     i1 <- GET-INDEX-OF-ROW(M1, i1)
29     i2 <- GET-INDEX-OF-ROW(M2, i2)
30 BEST-MATCH(i1, i2, isRow)
```

Figure 7: The algorithm to compute the best match.

gerprint data to be stored. We decided to output the fingerprint data as XML format for ease of parsing. We used an open source library named XStream [3] that helped importing the XML data into the fingerprint object.

The second issue was to derive a very unique pattern of every JAR file that is not easily changeable. We found that the relationship between classes can form a unique graph. In most cases, a copycat keeps the logic and the relationship between classes intact and changes the naming of classes, methods, variables, etc....

There is a possibility that there could be misleading results since two JAR files may have a small subset of the same class structure. After running several experiments, we found out in 20% of class dependency certainty percentage may be misleading result. Hence if the class dependency particle reports the certainty percentage as 100%, there is 80% chance that code piracy has occurred.

Several graphs generated from the JAR files are usually small. We made a decision to ignore any graph that contains less than 5 nodes. This helped to increase the accuracy and minimize the misleading results.

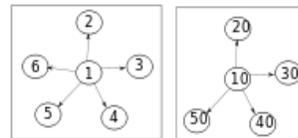


Figure 8: A star shaped graph.

The analysis of the class dependency showed that the comparison of two star shape graphs (see Figure 8) gave misleading results. The graphs traversal showed that a node in graph X could be identical to many nodes in graph Y. Therefore, it makes no difference which node is the start node in graph Y since all the traversal results would always produce the same number of visits.

We decided to develop an image particle since there is a high probability that a copier would not modify the images. For simplicity, we picked the first one hundred bytes of an image and MD5 that.

The class logic particle is included in the application to enhance the class dependency comparison results. In most cases, the copycat changes the naming of classes, methods, variables, etc... and do not change the logical keywords such as if, else, while, for, etc...

Lastly, the filename and filesize particles were incorporated into the program since they are the quickest and the simplest to analyze the difference between two JAR files.

## 4. Results

We have collected several results to confirm that the relationship between classes is indeed a unique approach to generate a fingerprint signature. We choose 10 JAR files from the Eclipse IDE tool to produce our results, as shown in Table 1. The analysis was conducted in the following environment: Windows XP Professional SP3, with Intel Dual Core CPU T7600 processing chip at 2.33 GHZ, 3.00 G.B. of Memory RAM, 667 MHZ Bus Speed, and 4 MB of L2 Cache.

For each entry in Table 2, the number of classes, images, and the total number of files found in each JAR file is listed in Table 1. Table 2 also includes the JAR filesize and the generated fingerprint size in bytes. The table shows that the fingerprint size is small compared to the JAR filesize.

Table 3 shows the total running time of each particle for generating a fingerprint.

Table 4 shows the certainty percentage of the class dependency devised from comparing a JAR file to itself, or two unrelated JAR files or by combining two

JAR #	JAR Filename
1	.uml2.codegen.ecore.ui_1.1.0.jar
2	.datatools.connectivity.db.generic.ui_1.0.1.v200906020900.jar
3	.ui.views_3.4.0.I20090525-2000.jar
4	.wst.web.ui_1.1.300.v200905242131.jar
5	.uml2.editor_1.1.0.jar
6	.wst.xml.xpath.ui_1.0.0.v200904240436.jar
7	.ui.navigator.resources_3.4.0.I20090525-2000.jar
8	.ui.views.log_1.0.100.v20090527.jar
9	.ui.console_3.4.0.v20090513.jar
10	.ui.browser_3.2.300.v20090526.jar

**Table 1:** An ID associated with every JAR file. Each filename starts with org.eclipse.

JAR #	Classes	Images	Total Files	Size in KB	Fingerprint Size in B
1	13	2	26	36	551
2	14	1	22	29	688
3	55	9	75	86	1541
4	39	6	57	76	1270
5	48	4	56	69	1160
6	50	14	75	98	1682
7	71	10	90	123	1924
8	78	31	118	133	2560
9	95	14	119	155	3056
10	108	38	156	183	3508

**Table 2:** Detail information of each JAR file listed in Table 1.

JAR files into a single file and comparing the results to one of the combined JAR files.

JAR #	Cl. Dep.	Cl. Logic	Img	Fl. size	Fl. name	Total
1	47	47	16	16	31	157
2	31	78	32	16	15	172
3	125	219	47	31	31	453
4	157	250	47	31	31	516
5	203	188	94	31	16	532
6	157	297	78	31	31	594
7	203	281	47	32	46	609
8	1844	344	94	31	31	2344
9	4281	406	62	32	31	4812
10	5312	360	78	78	47	5875

**Table 3:** Particle Running Time in Milliseconds.

Compared JARs	Class Dependency
1 and 1	100%
2 and 2	100%
3 and 3	100%
10 and 9	40%
10 and 8	17%
10 and 7	5%
1 and 1 combined with 2	100%
4 and 4 combined with 2	100%
7 and 7 combined with 10	100%

**Table 4:** The certainty percentage of comparing two JAR files. The first column represents the JAR IDs from Table 1.

## 5. Conclusion

In this paper we have presented a new approach to IP detection that depends on the class structure in a

JAR file. We are able to generate a unique fingerprint for each JAR file, where the size of the fingerprint is very small relative to the JAR file size. The class dependency and class logic approach provided crucial information extracted from the JAR file. This information helped to analyse and report an accurate certainty percentage of the comparison operation, after the misleading results were taken into account. The framework was designed in a specific way such that new particles can be added very easily and rapidly to increase accuracy. We believe our framework provides a good base for future particle additions by us and others.

## 6. Future Work

As we mentioned in Section 3.4, we encountered a difficult problem with the star shaped graphs. This is a high priority tasks to be completed and can be achieved by combining analysis with other particles such as class logic.

The current class dependency matching algorithms can definitely be enhanced to provide a better running time. There is a possibility that the approximation algorithm used can run in  $O(n^2)$  time.

Currently, the fingerprint is stored in a XML file. These fingerprints can be stored in a public central database and be retrieved based on analysis needs. This would help bring open source software communities together in fight for intellectual property piracy.

## References

- [1] Incidence matrix. Wikipedia the free encyclopedia, 14 March 2010. [http://en.wikipedia.org/wiki/Incidence\\_matrix](http://en.wikipedia.org/wiki/Incidence_matrix).
- [2] Jclazz. SourceForge, 14 March 2010. <http://jclazz.sourceforge.net/>.
- [3] Xstream, 14 March 2010. <http://xstream.codehaus.org/>.
- [4] Carson Brown, David Barrera, and Dwight Deugo. Figd: An open source intellectual property violation detector. *SEKE*, pages 536–541, 2009.
- [5] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns*. Addison Wesley, 1995.
- [6] V. Kann. *On the approximability of the maximum common subgraph problem*, volume 577, pages 375–388. Springer Berlin / Heidelberg, 1992.

# Detection of Malicious Software Engineer Intrusion

Michael E. Shin, Nipul Patel, Snehadeep Sethia

*Department of Computer Science*

*Texas Tech University*

*Box 43104, Lubbock, TX 79409*

*1-806-742-3527*

*{Michael.Shin, nipul.k.patel, Snehadeep.sethia}@ttu.edu*

## Abstract

*This paper describes an approach to detecting malicious software engineer intrusion attacks using the business processes (use cases) of applications. The approach detects malicious codes inserted by malicious software engineers to applications during software development or maintenance. The proposed approach protects application systems from malicious codes in terms of maliciously release (**confidentiality security requirement**) or change of sensitive information (**integrity security requirement**) encapsulated in application systems.*

## 1. Introduction

As security is becoming more important for business these days, secure application systems are required to support business security. In general, application systems are developed and maintained by multiple software engineers, such as program developers and maintenance software engineers. Though most of the software engineers are ethical, a few may be unethical and malicious. A malicious software engineer, who may not be satisfied with the organization, may add malicious code to application systems or change existing code maliciously such that the malicious code is activated either periodically or non-periodically for personal gain or disrupt service or destroy organization system and/or data. This is referred to as **malicious software engineer intrusion** in this paper.

A number of intrusion detection systems have been developed for improving computer system security, but application systems are still vulnerable to the attacks from insider intrusions as well as outsider intrusions of the systems. Most of the intrusion detection systems have been focused on outsider intrusion detection for operating systems, network systems, and database systems, but less attention has been paid to intrusion detection for application systems [Balzarotti07, Gómez07, Jones01, Masri08, Stillerman99], such as banking systems and finance systems [Randazzo04], which contain lots of sensitive business information. Most of the intrusion detection approaches for application systems focuses on

outsider intrusion, but not much is available for insider intrusion, such as malicious codes created by malicious software engineers.

While many organizations use mandatory code review for development of systems and follow stringent configuration processes for change control, several insiders such as software engineers were able to inject malicious code into new systems or existing systems [Cappelli08, Cappelli09]. This is because, although review of all the program codes in a system can detect malicious code or many different defects (used by insiders to insert malicious code to a system), line-by-line review of the entire code is not cost and time effective for projects that have a short timeline to be met. Also ineffective configuration or change control processes contributed to insiders' ability to insert malicious code to the systems as well. Practically, code review or change control process can be limited to only security-relevant programs.

This paper describes an on-going project for developing an intrusion detection approach that detects dynamically malicious code created by malicious software engineers. The intrusion detection approach proposed in this paper concentrates on malicious software engineer intrusion attacks at the application level. Using the business processes (use cases) of applications, the proposed approach detects malicious codes inserted by malicious software engineers to the system during the software development or maintenance phase.

### 1.1 Example of Malicious Software Engineering Intrusion

An Automated Teller Machine (ATM) system [Gomaa00], which is part of a banking system, may have several components composed of multiple objects. An ATM server component in an ATM system provides bank customers with several services (use cases), such as Withdraw Funds, Query Account, and Transfer Funds. The ATM server component can be composed of multiple objects so as to support these services. Figure 1 depicts the communication diagram in the Unified Modeling Language (UML) [Booch05, Rumbaugh05] for the Transfer Funds service in the ATM server component, which is supported by Bank Transaction Server, Transfer Transaction Manager, Checking Account, and Savings Account objects. The

UML communication model describes a dynamic view of the system. The messages T1 through T6 describe a successful message sequence for the Transfer Funds service (use case) in the ATM server component without considering exceptional branches. The ATM server component has more objects to implement the other services such as Withdraw Funds and Query Account, but it is simplified for this paper.

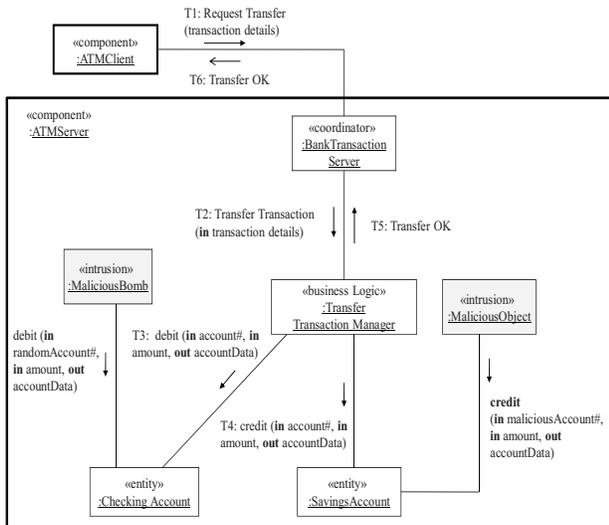


Figure 1. ATM Server component

A malicious ATM software developer or maintenance software engineer can add a Malicious Object (Figure 1) to the ATM Server component during development or maintenance phase such that the object accesses the Savings Account object to earn gain maliciously. The Malicious Object may be made either by changing an existing object or by adding a new object to the component. The Malicious Object is designed to invoke the credit() operation in the Savings Account object so as to credit some amount to a specific account periodically. **(This violates the integrity security requirement.)** The Malicious Object does not need any confidential information of a customer, such as account number and Personal Identification Number (PIN), to invoke the credit() operation in the Savings Account object.

In other case, a Malicious Bomb object (Figure 1) can be planted in the ATM Server component by a malicious software engineer who may be convinced of her/his lay-off. After the software engineer leaves, the Malicious (logic or time) Bomb object may be activated to repeatedly debit some amount from random customer checking accounts so that it disrupts system services or destroys account data. **(This is a case of violation of integrity security requirement.)**

A malicious ATM software developer or maintenance software engineer may add an extra object to the ATM Client component so that she/he accesses customer transaction information including card number maliciously. Figure 2 depicts the UML communication diagram for the Transfer Funds service in the ATM Client component, which is simplified for this paper. The message sequence K1 through K6 receives a customer input and requests transfer funds from the ATM Server component. A Malicious Object is designed to invoke the read() operation of ATM Transaction object that has the

customer transaction information. **(This is a case of violation of the confidentiality security requirement.)** The Malicious Object may request transfer funds maliciously from the ATM Server component **(This is also a case of violation of the integrity security requirement.)**

A malicious ATM software developer or maintenance software engineer may create a malicious object outside the ATM Server and ATM Client components, which invokes some operation of objects within the ATM Server or Client components maliciously. The malicious object may invoke the read() operation of ATM Transaction object in the ATM Client component or may invoke the credit() operation of Savings Account object in the ATM Server component.

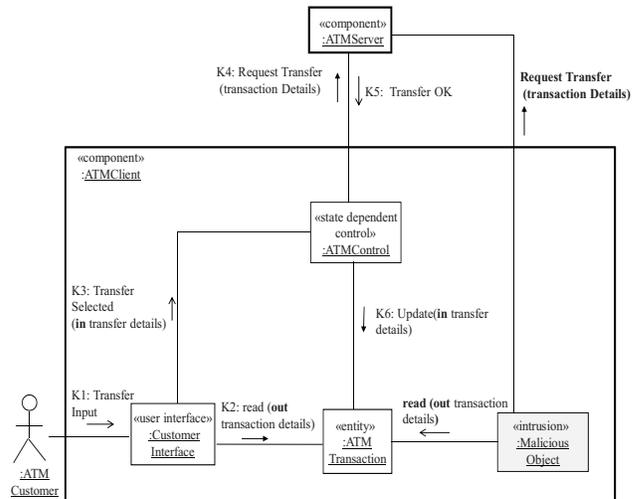


Figure 2. ATM Client component

## 2. Related Work

Research for insider intrusion is increasing gradually as insider attacks are severe and devastating than outsider attacks. Several techniques have been suggested in order to reduce this kind of insider attacks:

- Programming techniques can be used as an insider attack tool [Cappelli08]. Many intrusion cases related to programmers were analyzed to suggest some general practices (guidelines), which can be helpful to reduce malicious programmer intrusion in development of systems and during the maintenance phase. However, the practices do not provide the detailed technique for detecting malicious software engineer intrusion.
- An insider's accumulated knowledge about the system can be used to access sensitive data (objects) that should not be authorized to the insider. A knowledge-based model [Althebyan07] is developed for predicting insider threat using knowledge and object dependencies.
- A formal model [Chinchani05] is developed to detect insider threats and attacks. This approach calculates the cost of an attack using key challenge graph and metrics for each attack. The cost of an attack is used to predict insider threats.

- To protect documents from insider threat, an ontological approach [Aleman-Meza05] utilizes the notion of semantic associations and their discoveries among a collection of documents.
- Specification-based approach [Ko97] relies on program specifications that describe the behavior of privileged programs in Unix. Intrusion can be detected while the programs execute if there is a deviation of their behavior from the specifications. This approach is different from our approach that traces a deviation of component states.
- [Forrest96, Hofmeyr98] suggests a method for detecting intrusions at the level of privileged processes using system calls executed by the processes. Normality of a privileged process is defined in terms of short sequences of system calls executed by a process. This approach has been enhanced by the Fast Automaton-Based Intrusion Detection Method [Sekar01]. These approaches rely on system calls by processes, but our approach depends on accesses to security-relevant objects.

### 3. Overview of Our Approach

The underlying idea for detecting malicious software engineer intrusion is that a system state, composed of both the state of each security-relevant component and the state of interaction between (security-relevant or security-irrelevant) components, should not violate confidentiality and integrity security requirements for a secure application system. In this paper, a security-relevant component or object is defined to be a component or object that contains sensitive information, which should not be released to or changed by unauthorized parties.

The detection of malicious software engineer intrusion is performed at two levels – at the security-relevant component level and component interaction level. At the security-relevant component level, a security-relevant component dynamically detects a malicious code inserted by a malicious software engineer to the component when the malicious code is activated to invoke an operation provided by a security-relevant object of the component. At the component interaction level, intrusion is detected if a malicious code in a component invokes an operation of a security-relevant object in a different security-relevant component.

An extended state machine (statechart) is used to describe the security-relevant component states and the interaction states between components. Along with the UML communication model, a state machine model in the UML [Booch05, Rumbaugh05] describes the dynamic view of a system - the states of a system (or component) and their transitions in terms of events of application concern. In this paper, a state machine describing application concern of a system is extended to include security concern such that the states of a system consist of security states related to security concern as well as application states related to application concern. A secure state of a system or component is defined to be either a security state or application state. When a system or component, referred to as a secure system or component, is in a secure state, it means the system or component is not under a malicious software engineer intrusion attack. But a deviation from secure states indicates that the system or component is under an intrusion attack.

A secure application system detecting malicious code inserted by a software engineer is developed using secure components and interaction relationship between the components. A component in a system is either a state-dependent or non-state dependent component. In this paper, both security-relevant state-dependent components and security-relevant non-state dependent components are designed to be secure components that detect malicious code created by a software engineer. The interaction relationship between components is used to detect malicious software engineer intrusion when a malicious code inserted by a software engineer to a component requests a service from a different security-relevant component.

To achieve our approach, we assume that some objects are secure so that they are not compromised by a malicious software engineer. The objects should be reviewed and tested by multiple software engineers for **separation of duty**, being trusted for detecting malicious software engineer intrusion. The following objects are assumed to be secure:

- a) A security-relevant object that either contains or manipulates sensitive information in an application system.
- b) A state-dependent control object that encapsulates states of a state-dependent component and controls other objects in the component.
- c) An intrusion detector that detects malicious software engineer intrusion in a non-state dependent component.
- d) A system detector that detects malicious software engineer intrusion between components.
- e) An interface object that receives events to initiate a business use case in a security-relevant component.

## 4. Intrusion Detection for Malicious Software Engineer

A malicious software engineer intrusion is detected in security-relevant state-dependent components, security-relevant non-state dependent components, and in interaction between components.

### 4.1 Detection of Intrusion in Security-relevant State-Dependent Component

The state machine encapsulated in a state-dependent control object of a security-relevant state-dependent component is extended to include security aspects of the component. A state-dependent component [Gomaa00] contains one or more state-dependent control objects, each of which maintains the application aspects of the component using a state machine. The original state machine in a state-dependent control object of a state-dependent component contains only application states of the component, whereas the extended state machine for a security-relevant state-dependent component has additional security states as well as application states. When an operation of security-relevant object in a state-dependent component is invoked legitimately by some objects within the same component, the invocation causes a transition from one secure state to another secure state in the extended state machine.

A security-relevant state-dependent component detects malicious software engineer intrusion - maliciously release and/or change of sensitive information within the component - using the extended state machine of a state-dependent control object in the component. A message (i.e., an event) arriving at a state-dependent control object causes a transition in the extended state machine for the object. At a given state, a state-dependent control object expects to receive one of the messages that can cause a transition from one secure state to another secure state in the extended state machine. Whenever an operation of a security-relevant object is invoked by other objects, it notifies a state-dependent control object of the invocation. The notified message causes a transition in the extended state machine for the state-dependent control object if the message is one of expected messages. However, when a state-dependent control object receives an unexpected notification message from a security-relevant object, it detects that the sensitive information is released or changed maliciously in the security-relevant object.

Figure 3 depicts the revised secure ATM Client component for transferring funds in Figure 2. The ATM Control object is state-dependent, encapsulating an extended state machine (Figure 4), which has security states - *Requesting Transaction Details* and *Updating Transaction Details* - and application states - *Waiting for Customer Choice* and *Processing Transfer*. The ATM Transaction object is a security-relevant object that contains the customer card number. When the read() operation of the ATM Transaction object is invoked, the extended state machine in the ATM Control object should be the *Requesting Transaction Details* security state. Whenever the read() operation of the ATM Transaction object is invoked, the ATM Transaction object sends the Transaction Details Requested message to the ATM Control object. The ATM Control object detects an intrusion attack at runtime if it receives the Transaction Details Requested message from the ATM Transaction object when the ATM Control object is not in the *Requesting Transaction Details* security state. This means some malicious code in a Malicious Object (Figure 2) tries to read the customer card number maliciously. Similarly, the *Updating Transaction Details* security state is used to detect malicious software engineer intrusion attacks.

#### 4.2 Detection of Intrusion in Security-relevant Non-State Dependent Component

An intrusion detector object is designed for a security-relevant non-state dependent component to detect malicious software engineer intrusion at runtime. Unlike a state-dependent component, a non-state dependent component is a stateless one, which does not need to maintain the component state information so as to provide services. Thus a non-state dependent component does not contain a state-dependent control object in the component. In our approach, an "intrusion detector" object is designed and added to each security-relevant non-state dependent component. An intrusion detector object encapsulates a state machine describing the states of security-relevant objects in a security-relevant non-state dependent component.

A security-relevant non-state dependent component detects malicious software engineer intrusion at runtime using the state

machine encapsulated in the intrusion detector object. Whenever a security-relevant object in a security-relevant non-state dependent component is invoked by other objects to read or change sensitive information, it notifies the intrusion detector object. The notification message causes a transition in the state machine of the intrusion detector object. If an intrusion detector object receives an unexpected message from a security-relevant object at a given state, it detects that a security-relevant object is invoked by some malicious object.

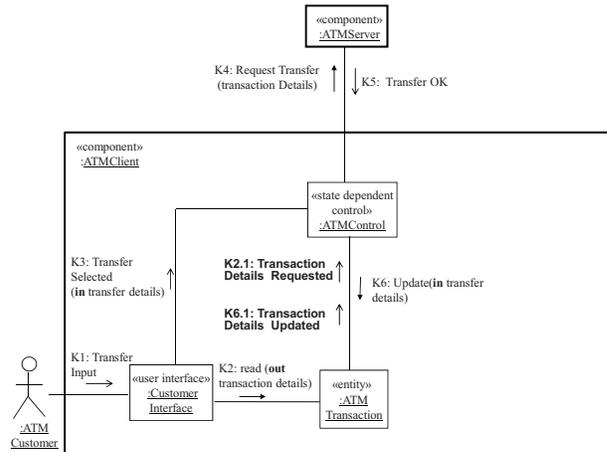


Figure 3. Security-relevant State-dependent ATM Client component

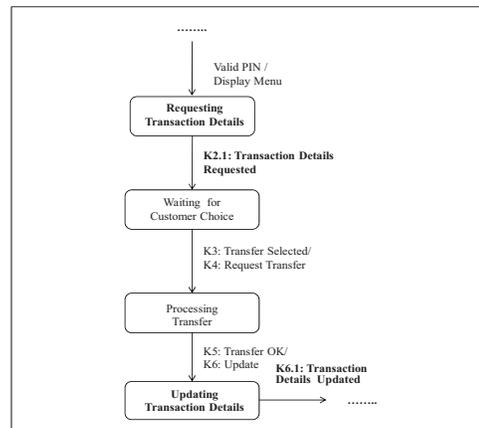
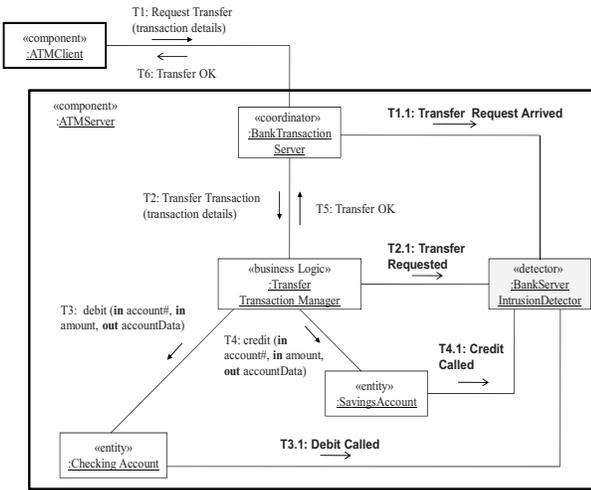


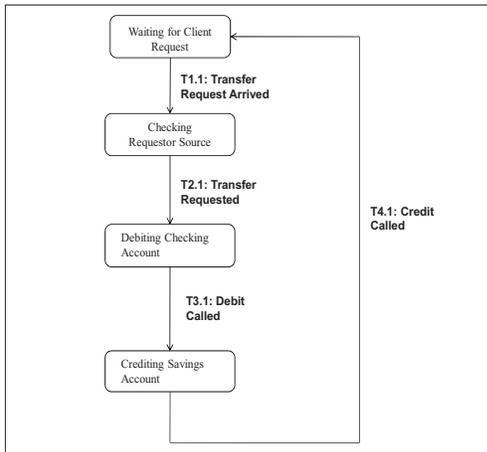
Figure 4. State Machine for Security-relevant State-dependent ATM Client component

Figure 5 depicts the revised secure ATM Server component for transferring funds in Figure 1. The Bank Server Intrusion Detector object encapsulates a state machine (Figure 6), which has the *Waiting for Client Request*, *Checking Requestor Source*, *Debiting Checking Account*, and *Crediting Savings Account* states. The Savings Account is a security-relevant object that contains the sensitive information of savings accounts. When the credit() operation of the Savings Account object is invoked, the Bank Server Intrusion Detector object should be in the *Crediting Savings Account* security state. Whenever the credit() operation is invoked, the Savings Account object sends a "Credit Called" notification message to the Bank Server Intrusion Detector object. The Bank Server Intrusion Detector

object detects a malicious software engineer intrusion attack (by the Malicious Object in Figure 1) if it receives the notification message when the detector is not in the Crediting Savings Account security state.



**Figure 5. Security-relevant Non-state Dependent ATM Server Component**



**Figure 6. State Machine for Security-relevant Non-state Dependent ATM Server Component**

### 4.3 Detection of Intrusion between Components

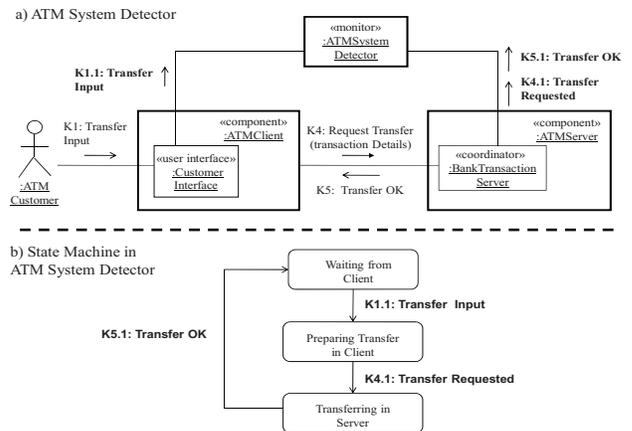
The intrusion detection methods for security-relevant (state-dependent or non-state dependent) components described in sections 4.1 and 4.2 can detect only malicious software engineer intrusion when a malicious code in a component invokes some operation of a security-relevant object within the same component. However, some malicious code in a component can be designed to avoid this intrusion detection in such a way that it requests a service maliciously from a different security-relevant component. An example is that the Malicious Object in the ATM Client component in Figure 2 requests a Transfer Funds service from the security-relevant non-state dependent ATM Server component. In this case, the malicious transfer

funds service request is processed by the security-relevant ATM Server component without detecting the intrusion.

A system detector is designed to detect malicious software engineer intrusion between different components. A system detector monitors the interaction between different components, detecting a malicious intrusion when a component requests a service maliciously from a different, security-relevant component. For this, the system detector contains the interaction relationship between components, which can be described using a state machine. When a service (that is, a use case) provided by a component is triggered by an internal or external event, the component notifies the system monitor that traces the use case sequence. This use case sequence between components is described in a sequential or concurrent state machine. The system detector detects a malicious intrusion between components if a service request from a component does not take place as specified in the state machine.

Figure 7 depicts an ATM System detector between the ATM Client and Server components, which encapsulates a state machine describing interaction between the components. When the Transfer Funds service (use case) is triggered by the ATM Customer, the Customer Interface notifies the ATM system detector of the *Transfer Input* ((a) in Figure 7), which causes a transition from the *Waiting from Client* state to the *Preparing Transfer in Client* state ((b) in Figure 7). The Bank Transaction Server object in the ATM Server component also notifies the ATM System detector whenever it receives the *Request Transfer* message. The notification message causes a transition from the *Preparing Transfer in Client* state to the *Transferring in Server* state in the state machine of the detector. The Customer Interface and Bank Transaction Server are assumed secure in section 3.1. The Transfer Funds should be requested by the ATM Client component, and the request should take place when the ATM System detector is in the *Preparing Transfer in Client* state. Otherwise, The ATM System detector detects malicious software engineer intrusion. This is the case where the Malicious Object (Figure 2) requests transfer funds maliciously from the ATM Server component.

The messages between an interface object and a system detector is authenticated. When a message has arrived at the system detector, the detector needs to check who sent the message. This is to prevent a malicious object from sending a fake message to the system detector.



**Figure 7. System Detector between ATM Client and Server Components**

## 5. Conclusions

This paper has described an on-going project for developing an intrusion detection approach that detects dynamically malicious code created by malicious software engineers. The detection of malicious software engineer intrusion is performed at two levels – security-relevant component level and component interaction level. For this, a state machine (statechart) or an extended state machine is used to capture the security-relevant component states and the interaction states between components. A component in a system is either state-dependent or non-state dependent. Both security-relevant state-dependent components and security-relevant non-state dependent components are designed to be secure components, whereas the interaction relationship between components is used to detect malicious software engineer intrusion.

This paper can be extended for further research. In some cases, malicious code inserted by software engineer may invoke a security-relevant object just before the object should be invoked by expected other objects. Although this case is not detected real-timely by a state-dependent control object or intrusion detector, this is detected by the security-relevant object's next notification message. However, we will investigate about how this case is detected in a real-time manner. We also plan to investigate that the formal rules about how an extended state machine can be created for a security-relevant state-dependent component and how a state machine in an intrusion detector can be created for a security-relevant non-state dependent component. These state machines can be formalized for given security-relevant state or non-state dependent components. In addition, we plan to investigate the way of co-operation between system detectors in distributed application systems. Security-relevant components can be distributed in multiple nodes (computers), each of which may have a distributed system detector monitoring the components in a node. The distributed system detectors need to co-operate with each other in order to trace the interaction between components in different nodes.

## 6. References

[Althebyan07] Q. Althebyan and B. Panda, "A Knowledge-Base Model for Insider Threat Prediction," Workshop on Information Workshop on Information Assurance, United States Military Academy, June 2007.

[Aleman-Meza05] B. Aleman-Meza, P. Burns, M. Eavenson, D. Palaniswami, and A. Sheth, "An Ontological Approach to the Document Access Problem of Insider Threat," In Proceedings of the IEEE International Conference on Intelligence and Security Information, ISI 2005, Atlanta, Georgia, USA, May 19-20, pages 486-491, 2005.

[Balzarotti07] D. Balzarotti, M. Cova, V. V. Felmetzger, and G. Vigna, "Multi-module vulnerability analysis of web-based applications," Proceedings of the 14th ACM Conference on Computer and Communications Security, Alexandria, Virginia, USA, October 28 - 31, 2007.

[Booch05] G. Booch, J. Rumbaugh, and I. Jacobson, "The Unified Modeling Language User Guide", Second Edition, Addison Wesley, Reading MA, 2005.

[Cappelli08] D. M. Cappelli, T. Caron, R. F. Trzeciak, and A. P. Moore, "Spotlight On: Programming Techniques Used as an Insider Attack Tool", CERT, Software Engineering Institute, and CyLab of Carnegie Mellon, December 2008.

[Cappelli09] D. M. Cappelli, A. P. Moore, R. F. Trzeciak, and T. J. Shimeall, "Common Sense Guide to Prevention and Detection of Insider Threats," CERT, Software Engineering Institute, and CyLab of Carnegie Mellon, January 2009.

[Chinchani05] R. Chinchani, A. Iyer, H Ngo, and S. Upadhyaya, "A Target-Centric Formal Model for Insider Threat and More," In Proceedings of the International Conference on Dependable Systems and Networks, pages 108-117, October 2005.

[Forrest96] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A sense of self for Unix processes," In Proceedings of the IEEE Symposium on Computer Security and Privacy, Oakland, Calif., IEEE Press, 1996.

[Gomaa00] H. Gomaa, "Designing Concurrent, Distributed, and Real-Time Applications with UML," Addison-Wesley, 2000.

[Gómez07] J. M. Gómez and J. Lichtenberg, "Intrusion Detection Management System for eCommerce Security," Journal of Information Privacy & Security, vol. 3, no. 4, pp. 19-31, 2007.

[Hofmeyr98] S. A. Hofmeyr, S. Forrest, and A. Somayaji "Intrusion Detection using Sequences of System Calls," Journal of Computer Security, vol. 6, pp. 151-180, 1998.

[Jones01] A. K. Jones and Y. Lin, "Application Intrusion Detection using Language Library Calls," 17th Annual Computer Security Applications Conference (ACSAC'01), 2001.

[Ko97] C. Ko, M. Ruschitzka, and K. Levitt, "Execution Monitoring of Security-Critical Programs in Distributed Systems: A Specification Based Approach," In Proceedings of the IEEE Symposium on Security and Privacy, 1997.

[Masri08] W. Masri and A. Podgurski, "Application-based anomaly intrusion detection with dynamic information flow analysis", Computers & Security, vol. 27, no.5-6, October 2008.

[Randazzo04] M. R. Randazzo, M. Keeney, E. Kowalski, D. Cappelli, and A. Moore, "Insider Threat Study: Illicit Cyber Activity in the Banking and Finance Sector," Software Engineering Institute/ Carnegie Mellon, August 2004.

[Rumbaugh05] J. Rumbaugh, G. Booch, and I. Jacobson, "The Unified Modeling Language Reference Manual," Second Edition, Addison Wesley, Reading MA, 2005.

[Sekar01] R. Sekar, M. Bendre, D. Dhurjati, and P. Bollineni, "A Fast Automaton-Based Method for Detecting Anomalous Program Behaviors," In Proceeding of the 2001 IEEE Symposium on Security and Privacy, 2001.

[Stillerman99] M. Stillerman, C. Marceau and M. Stillman, "Intrusion detection for distributed applications", Communications of the ACM, vol. 42, no. 7, pp. 62-69, July 1999.

# Developing Precise Misuse Cases with Security Robustness Analysis

Mohamed El-Attar

Information and Computer Science Department  
King Fahd University of Petroleum and Minerals  
P.O. Box 5066, Al Dhahran, 31261  
Kingdom of Saudi Arabia  
melattar@kfupm.edu.sa

## Abstract

*In recent years, misuse modeling has emerged as a promising technique to elicit, model and communicate security requirements. Currently, misuse case authors have no guidance towards developing precise misuse case descriptions that accurately represent potential security threats and protection measures. Moreover, there remains a gap between the analysis and design phases of security aspects, meaning that misuse case models are solely and directly used to drive the development of detailed design artifacts that satisfy its underlying security requirements. Such large gap can lead to the development of an end system that does not satisfy its security requirements. This paper presents an approach that will guide misuse case modelers towards authoring precise misuse case descriptions. The approach is based on performing robustness analysis on use case descriptions with the intent of determining workflows of security attacks that may occur during the execution of a use case. The robustness analysis performed will also allow security analysts to precisely determine how mitigation measures against the misuse scenarios will be performed. The proposed approach can be used to bridge the gap between the analysis and design phases. The proposed approach is demonstrated through an industry strength case study of a restaurant mapping system.*

**Keywords** Misuse cases • Use cases • Robustness diagrams • Secure software development

## 1. Introduction

Nowadays, most applications that are not engineered to operate securely are deemed unusable as they are vulnerable to attacks from outsiders and insiders. Traditionally, security requirements were addressed towards the end of a development process. Software systems would be supplied with defensive mechanisms

such as firewalls, IDS (Intrusion Detection Systems) and cryptographic components. However, this traditional approach has proven to be insufficient, potentially leading to costly reworks [8]. The software development community has become increasingly aware of the need to consider security requirements at the requirements engineering phase [4]. Security measures need to be built as an integral part of a system's structure and machination, rather than being "patched-on" just before deployment.

The growing success of misuse case modeling can be attributed to the popularity of the use case modeling technique [9, 14]. A misuse case model is an augmentation to a use case model. The additional components introduced by a misuse case model describe misuse scenarios (also referred to as "Negative" scenarios). Misuse case models introduce two entities which are described in [15] as follows:

Misuse Case: "A Sequence of actions, including variants, that a system or other entity can perform, interacting with misusers of the entity and causing harm to some stakeholder if the sequence is allowed to complete". [15]

Misuser: "An actor that initiates misuse cases, either intentionally or inadvertently". [15]

Currently, misuse case modeling suffers from two major drawbacks:

- (1) Similar to use cases, the main component of misuse cases lays in their textual descriptions. While the literature have provided a number of templates to describe misuse case models, there lacks any guidance towards authoring precise misuse case descriptions.
- (2) In a use case-driven approach, misuse cases are expected to be utilized to drive the development of design artifacts which will satisfy the underlying security requirements. However, this leaves a large gap between the analysis and design phases. As

such, designers will be highly vulnerable to misinterpret the security requirements and ultimately delivering an insecure end system. Moreover, such an approach may limit the designers' ability to consider various design strategies as they become prematurely concerned with the details of their design artifacts.

This paper presents an approach that attempts to remedy the abovementioned drawbacks. The contribution of the approach presented in this paper is thus two-fold. Firstly, the approach guides misuse case authors towards developing precise misuse case descriptions. Secondly, the approach also helps bridge the gap between the analysis and design phases using security robustness analysis.

The remainder of this paper is organized as follows: Section 2 presents a brief background on misuse case modeling and robustness analysis as well as related works in both areas. Section 2 also describes the proposed approach. In Section 3, a use case from the RestoMapper case study is presented to demonstrate the feasibility of the proposed approach. Section 4 concludes and describes future work.

## 2. Related Work on Misuse Case Modeling and Robustness Analysis

The literature shows that misuse cases can be described using two methods: (a) a *lightweight* approach whereby the misuse case description is embedded within the corresponding use case description in a *Threats* field, and (b) an *extensive* approach where the misuse case is described in a separate template on equal terms with regular use case templates. Examples of both description methods are presented in [15]. During the requirements gathering process, it is suggested to use the *lightweight* authoring approach. When requirements become more stable, misuse case descriptions can be transformed into the *extensive* format. The approach proposed in this paper is concerned with early definition of security requirements and therefore it is based on misuse cases described using the *lightweight* approach. The notation of misuse case models is similar to use case models, but with their entities' colors inverted as a symbol of their negation property.

Misuse case modeling remains a relatively new technique and hence most of the research work conducted in this area has been directed towards its validation. To date, several industrial and research projects have validated the effectiveness of the technique, such as [1-3, 2, 6, 15]. This early work

resulted in a number of refinements and extensions to the modeling notation [5, 6, 10, 13]. In [15], a five step process to elicit security requirements with misuse cases was outlined. However, no research work was conducted towards authoring precise misuse case descriptions.

Robustness analysis is a technique that was introduced in [11]. The technique is concerned with analyzing a use case description sentence-by-sentence to develop a robustness diagram. A robustness diagram is in essence a simplified version of a collaboration diagram which allows analysts and designers to take a first-cut guess at the objects required to realize scenarios described in use cases. The advantage of robustness diagrams over other modeling techniques is that it contains a small notational set and syntax rules. This allows its users to consider various design strategies without committing to any particular design prematurely and becoming overly concerned with design details and conforming to a wide variety of syntax rules. Robustness analysis helps provide a seamless transition from requirements to design. A robustness diagram depicts several concepts shown in Table 1:

**Table 1** Robustness diagram objects

Entity	Symbol	Concept
Actors		Similar concept to an actor in use case diagrams.
Boundary		Actors communicate with the system using boundary objects.
Entity		Similar concept to an entity in a conceptual model.
Control		Undertakes logical tasks using boundary and entity objects.

Robustness analysis has thus far only been applied to regular use cases. This paper utilizes robustness analysis to model the security aspects of a misuse case description. Thus, the version of the technique featured in this paper is referred as *security robustness analysis*. Security robustness diagrams introduce two sets of objects. Firstly, objects that realize misuse actions, including misusers, boundary, control and entity objects. These objects have similar notation as that shown in Table 1 but are colored black to symbolize their negation property (in-line with misuse case models). Secondly, objects that realize mitigation actions, including actors, boundary, control and entity objects. These objects have similar notation as that shown in Table 1 but are colored green.

The workflow of the proposed approach is as follows:

- Step 1.** Perform security robustness analysis on a use case description to identify security threats.
- Step 2.** Update the original robustness diagram with objects that realize misuse actions.
- Step 3.** Populate the *Threats* field of a use case description based on the updated robustness diagram (from step 2) as well as the original use case description.
- Step 4.** Perform further robustness analysis using the current robustness diagram (from step 2) and use case description (from step 3) to identify mitigation measures.
- Step 5.** Update the robustness diagram with objects that realize the mitigation measures.
- Step 6.** Populate the *Mitigation Points* field of a use case description based on the updated robustness diagram (from step 5) and use case description (from step 3).

The final version of the robustness diagram can be used to drive the development of more detailed design artifacts, such as in [16], which utilizes the concept of mal-activity diagrams to model security attacks.

### 3. The RestoMapper Case Study

This section presents a case study to demonstrate the application of the proposed approach. The case study features a misuse case description that is based on a simplified version of an industry strength use case description and its associated robustness diagram. The misuse case is concerned with a Mapplet system called RestoMapper<sup>1</sup>, which is featured and discussed in great detail in the book *Agile Development with ICONIX Process: People, Process, and Pragmatism* authored by Rosenberg et al. [12]. RestoMapper is a mapping application that allows its user to locate restaurants in a given city based on their preferences. The use case featured in this paper is called “Filter Restaurants” and its description is presented in Table 2. The robustness diagram corresponding to the “Filter Restaurants” use case is presented in Figure 1.

**Table 2** Textual description of the “Filter Restaurants” use case

---

**Use Case: Filter Restaurants**

**Basic Flow - Filter by Features:**

- bf1.** A list of features is displayed for the User.
- bf2.** The User can select one or more features such as the availability of valet parking, live music and a

---

smoking section.

- bf3.** A RestaurantFilter is then created by the MapViewer based on the selected features.
- bf4.** The system queries the restaurants for the given city.
- bf5.** The system filters the retrieved cities according to the RestaurantFilter.
- bf6.** The map is then refreshed to display the filtered restaurants.

**Alternative Flow - Filter by Restaurant Chain:**

- af1-1.** A list of restaurant chains is displayed for the User.
- af1-2.** The User then selects a particular restaurant chain from the list.
- af1-3.** A RestaurantFilter is then created by the MapViewer based on the selected chain.
- af1-4.** The system queries the restaurants for the given city.
- af1-5.** The system filters the retrieved cities according to the RestaurantFilter.
- af1-6.** The map is then refreshed to display the filtered restaurants.

**Alternative Flow: No restaurants matching criterion**

- af2-1.** After **bf6** or **af1-6**, if no restaurants matched the filter criterion, the following popup message is displayed to the User: “No restaurants meet filter criterion. Please expand your search”.
- 

In summary, the intent of the use case is to allow users to filter a map containing restaurant icons based on a preferred restaurant chain selection, or based on a set of desired features. Such features include live music, valet parking and a smoking section. An alternative usage scenario to the “Filter Restaurants” use case may occur in the case that the filtering process results in no matches. In such case, a popup is generated to inform the user.

The remainder of this section is structured according to the six-step workflow of the proposed approach which is outlined in Section 2.

**Steps 1 & 2:**

The success of the process of identifying security threats depends on the experience and skill of the security analysts. This process can be aided by a repository of a reusable security threats [15]. For demonstrative purposes, it can be assumed that the general goal of a misuser of the “Filter Restaurants” use case is to manipulate the filtering process initiated by a user to illegally promote their restaurants while shielding away

---

<sup>1</sup> The RestoMapper system has been modified to prevent any copyright infringements.

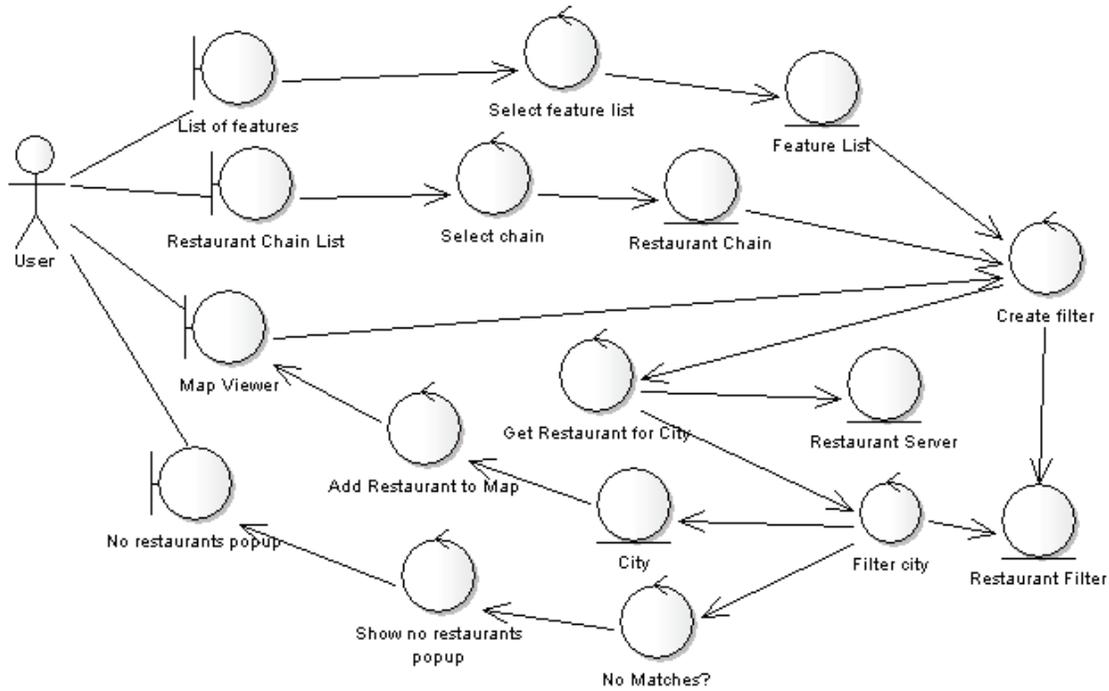


Fig. 1 Robustness diagram for the “Filter Restaurants” use case.

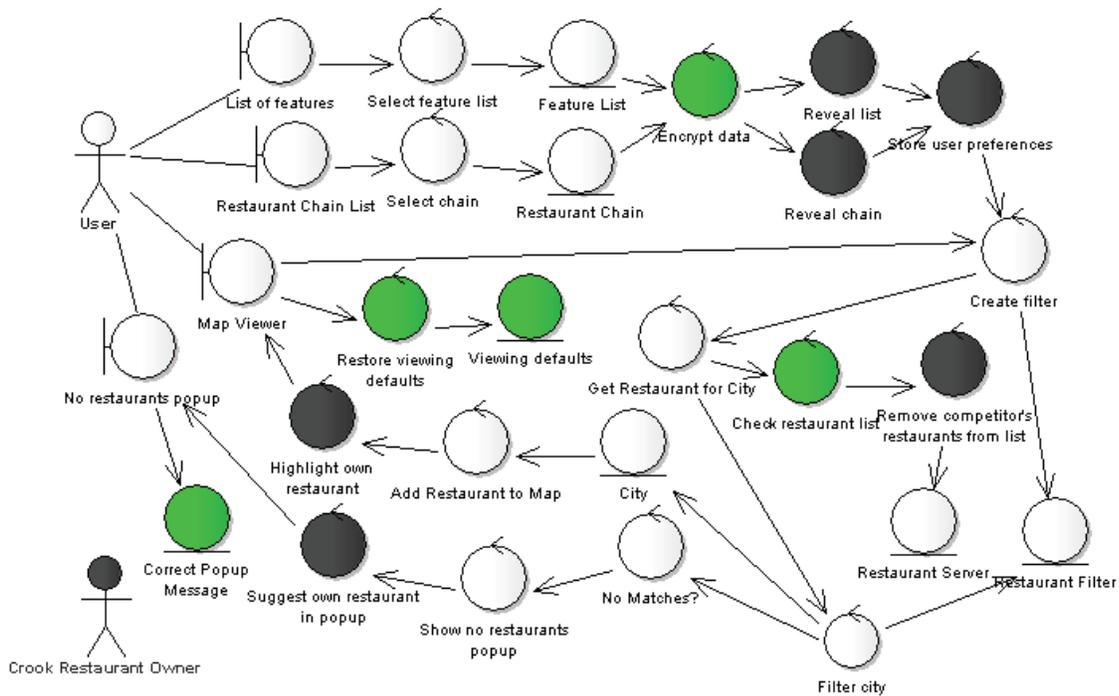


Fig. 2 Robustness diagram for the “Filter Restaurants” use case after perform security robustness analysis.

competitor’s restaurants from the user. This leads to the identification of a misuser which shall be named Crook Restaurant Owner. Based on the general harmful goals of the goals of the misuser, the original robustness diagram will be used to determine the “injection” points within the workflow of the use case where the security attacks can happen. The robustness diagram will then be modified to include the misuse actions which represent the various scenarios of security attacks (Step 2). The misuse actions are realized using misuse (black) objects. Due to space limitations, this intermediate version of the robustness diagram is not presented<sup>2</sup>.

**Step 3:**

Based on the updated robustness diagram, the security threats identified may now be documented in the *Threats* field of the use case description (see Table 3). The original use case description is augmented with the *Threats* field. Due to space limitations, only the *Threats* field is shown in Table 3 excluding the *Basic* and *Alternative flows* shown in Table 2.

**Table 3** *Threats* field of the “Filter Restaurants” use case description.

---

**Threats:**

- t1. After bf2**, the list of features selected by the User can be intercepted and revealed by the Crook Restaurant Owner.
- t2. After af1-2**, the User’s preferred restaurant chain selected can be intercepted and revealed by the Crook Restaurant Owner.
- t3. After t2**, the User’s preferences are stored by the Crook Restaurant Owner.
- t4. After bf4 or af1-4**, The Crook Restaurant Owner can remove the competitors’ restaurants from the list retrieved by the system from the restaurant server.
- t5. At bf6 or af1-6**, The Crook Restaurant Owner can highlight their favored restaurant to catch the User’s attention.
- t6. At af2-1**, The Crook Restaurant Owner can intercept and manipulate the message in the popup generated when no matches result from the filtering process. The modified message may contain a suggestion of the restaurant favored by the Crook Restaurant Owner.

---

<sup>2</sup> This intermediate robustness diagram version can be deduced by eliminating the mitigation (green) objects from Figure 2. The creation of mitigation objects occurs downstream and is explained at a later stage.

As shown in Table 3, when referring to the updated robustness diagram, the threats identified are described relative to the steps described in the *Basic* and *Alternative flows*.

**Steps 4 & 5:**

The updated robustness diagram is then used to determine the mitigation points where the misuse can be mitigated. The robustness diagram is then modified to include the mitigation (green) objects that will realize mitigation measures against the threats previously stated in the *Threats* field (see Figure 2).

**Step 6:**

Based on the updated robustness diagram, the mitigation points may now be documented in the *Mitigation Points* field. The *Mitigation Points* field is augmented to the current use case description (Table 3). Due to space limitations, only the *Mitigation Points* field is shown in Table 4 excluding the *Basic* and *Alternative flows* (Table 2), and the *threats* field (Table 3).

**Table 4** The *Mitigation Points* field of the “Filter Restaurants” use case description.

---

**Mitigation Points:**

- mp1. After bf2 and af1-2**, the information created is encrypted before being transmitted.
- mp2. Before bf5 and sf1-5**, the list of restaurants retrieved from the restaurant server for a given city must be verified against statistical information already stored in the system.
- mp3. At bf6 and af1-6**, viewing defaults are restored at the client machine to ensure that no restaurants are unduly highlighted.
- mp4. At af2-1**, the content of the popup message is retrieved from the application layer (the client machine) instead of being transmitted from the server side.

---

As shown in Table 4, when referring to the updated robustness diagram, the mitigation points identified are described relative to the steps described in the *Basic* and *Alternative flows*. However, it would be appropriate, if the need arises, to describe mitigation points relative to specific threats.

Note that the Crook Restaurant Owner misuser shown in Figure 2 should be connected to the misuse objects. However, the misuser was not connected to the misuse objects to prevent cluttering of the diagram. Also note that the security robustness analysis performed for this particular misuse case resulted in creating only control objects. However, this is not necessarily always

the case. Security robustness analysis can result in the creation of entity and interface objects as well.

#### 4. Conclusion and Future Work

The approach presented in this paper is a step towards developing higher quality misuse case models by offering guidance to modelers on how they can author their misuse case descriptions. The approach presented in this paper is also a step towards bridging the gap between the analysis and design phases in secure software engineering development, which will potentially lead to developing systems that properly address the security needs of a system.

The application of the proposed approach was demonstrated using an industry strength case study on a use case from a restaurant mapping system. The application of the proposed approach has shown how the *Threats* and *Mitigation Points* fields can be populated by precisely identifying how they relate to the steps described in the *Basic* and *Alternative flows*. The application of the proposed approach yielded robustness diagrams that can be readily used to spur the development of UML design artifacts, in particular, activity diagrams.

Future work can be directed towards utilizing the developed robustness diagrams as a source for the creation of acceptance tests for early validation systems. Formal approaches can also be devised towards using robustness diagrams to create more detailed design artifacts such as sequence diagrams and statecharts.

**Acknowledgments** The author would like acknowledge the support of King Fahd University of Petroleum and Minerals in the development of this work, which funded this work through project ID# IN100016.

#### References

1. Alexander IF (2002) Initial industrial experience of misuse cases in trade-off analysis. In: Proceedings of the 10<sup>th</sup> anniversary IEEE international requirements engineering conference (RE'02), Essen, Germany
2. Breivik GF (2002) Abstract misuse patterns—a new approach to security requirements. Masters thesis, Department of Information Science, University of Bergen
3. den Braber F et al (2002) Model-based risk management using UML and UP. In: Proceedings of the 13th IRMA international conference: issues and trends of information technology management in contemporary organizations (IRMA'2002), Seattle, Washington
4. Dubois, E. and S. Wu (1996). A framework for dealing with and specifying security requirements in information systems. Information Systems Security, Facing the information society of the 21st Century (SEC'96), Greece, Chapman&Hall.
5. Dwaikat, Z. and F. Parisi-Presicce (2004). From Misuse Cases to Collaboration Diagrams in UML. 3<sup>rd</sup> International Workshop on Critical Systems Development with UML, Lisbon, Portugal.
6. Herrmann, A. and B. Paech (2005). Quality Misuse. REFSQ'05, Porto, Portugal.
7. Houmb S-H et al (2002) Towards a UML profile for modelbased risk assessment. In: Proceedings of the UML'2002 satellite workshop on critical systems development.
8. Jürjens, J. (2004). Secure Systems Development with UML. Berlin, Springer.
9. D. Kulak and E. Guiney, *Use Cases: Requirements in Context*. Addison-Wesley, 2000.
10. Pauli, J. J. and D. Xu (2005). Misuse case-based design and analysis of secure software architecture. International Conference on Information Technology: Coding and Computing (ITCC 2005), Las Vegas, IEEE.
11. D. Rosenberg and Scott, K, *Use Case Driven Object Modeling with UML*. Addison-Wesley, 1999.
12. Rosenberg D, Stephens M, Collins-Cope M (2005) Agile development with ICONIX process: people process and pragmatism. Apress, Berkely.
13. Røstad, L. (2006). An extended misuse case notation, including vulnerabilities and the insider threat. REFSQ'06, Luxembourg.
14. Rumbaugh J (1994) Getting started: using use cases to capture requirements. Journal of Object Oriented Programming 7(5):8-23.
15. Sindre, G. and A. L. Opdahl (2000). Eliciting Security Requirements by Misuse Cases. TOOLS Pacific 2000, Sydney, IEEE CS Press.
16. Sindre, G. (2007). Mal-Activity Diagrams for Capturing Attacks on Business Processes. International Working Conference on Requirements Engineering: Foundations for Software Quality (REFSQ'07), Trondheim, Norway, Springer.

# Developing configurable extensible code generators for model-driven development approach

Souvik Barat

Tata Consultancy Services  
54-B, Industrial Estate  
Hadapsar, Pune, INDIA  
+91 20 66086261

souvik.barat@tcs.com

Vinay Kulkarni

Tata Consultancy Services  
54-B, Industrial Estate  
Hadapsar, Pune, INDIA  
+91 20 66086301

vinay.vkulkarni@tcs.com

## ABSTRACT

The need for agility and adaptiveness of business applications is on the rise with continued increase in business dynamics. Code-centric techniques show unacceptable responsiveness in this dynamic context as business applications are subjected to changes along multiple dimensions that evolve independently. Use of model driven techniques for developing such business applications is argued as a preferable option as platform independent specification can be retargeted to technology platform of choice through a code generation process. Recent literature suggests the use of product line architectures to increase agility and adaptability by capturing commonality and variability, and configuring them appropriately. By visualizing a business application along five dimensions, namely, Functionality (F), Business process (P), Design decisions (D), Architecture (A) and Technology platform (T), the use of models are largely limited to F and P dimensions in commonly used model-driven development techniques and thus use of product line concept is also limited to these two dimensions. In this paper we argue the use of product line concept, i.e. variability and configurability, in these two dimensions is not sufficient to achieve the desired agility and adaptability. It is as well critical to use product line concept for code generators that addresses D, A and T dimensions. This paper evaluates existing approaches for code generator product line, introduces a new approach, and discusses early experiences with the presented approach.

## Keywords

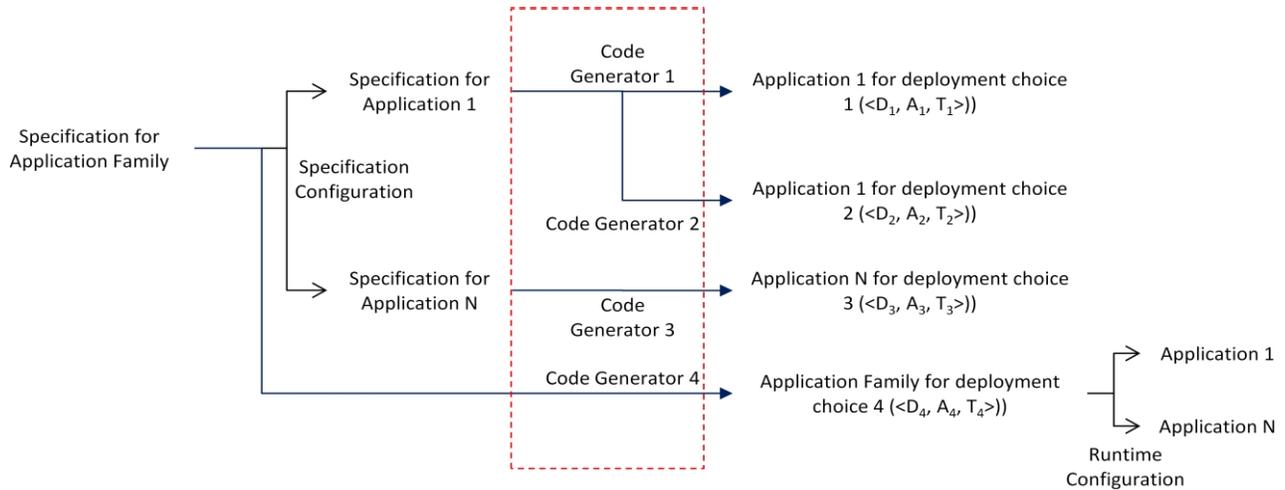
Configurability, Extensibility, Adaptiveness, Model-driven development, Code Generation product line

## 1. Introduction

Rapid evolutions of technology platforms and business demands have contributed to significant increase in business dynamics in recent years. The increased dynamics puts new requirement on businesses while opening up new opportunities that need to be addressed in an ever-shrinking time window. Stability and robustness seem to be giving way to agility and adaptiveness. This calls for a whole new perspective for designing (and implementing) IT systems so as to impart these critical properties. Traditional business applications typically end up hard-coding the operating context in their implementation. As a result, adaptation to a change in its operating environment leads to opening up of application implementation resulting in unacceptable responsiveness.

Typical database-intensive enterprise applications are realized conforming to distributed architecture paradigm that requires diverse set of technology platforms to implement. Such applications can be visualized along five dimensions, namely, Functionality (F), Business process (P), Design decisions (D), Architecture (A) and Technology platform (T). A purpose-specific implementation makes a set of choices along these dimensions, and encodes these choices within application implementation in a scattered and tangled manner [13]. This scattering and tangling is the principal obstacle in agile adaptation of existing implementation for the desired change. Large size of an enterprise application further exacerbates this problem. This is an expensive and error prone process demanding large teams with broad-ranging expertise in business domain, architecture and technology platforms. Model-driven development alleviates this problem to an extent by automatically deriving an implementation from its high-level specification using set of code generators.

In last decade we have developed several business-critical enterprise solutions for a variety of business verticals like banking, financial services and insurance [13]. In our experience no two business applications are exactly alike. For identical business intent, different enterprises, even from the same business domain, may have different requirements along these five dimensions with a significant overlap. Being ignorant of these similarities and differences would mean rework, and result in specification and implementation redundancy which will create maintenance and evolution problems later. For example, the functionality of a banking application can vary if it is targeting different geographies, e.g. USA and India. Similarly the same application can be targeted for delivery into different architecture and technology platform based on specific business requirements. Thus, it is important to capture commonality while highlighting the variability, i.e. product line architecture [5], to avoid multiple copies of same application with different choices of F, P, D, A and T dimensions. We are working towards a framework for developing adaptive applications such that variations (changes) along any of these dimensions can be incorporated easily within the desired time window. Our definition of adaptiveness subsumes configurability (i.e. selecting one of the many available variants) and extensibility (i.e. adding a new variant). We use model driven development approach for developing business application wherein a business application is specified in terms of platform independent models (conforming to a meta model) and high level specifications; and transformed into a deployable implementation using code generators. Each code generator translates a specific view i.e. generating a specific architectural layer e.g. presentation layer, business process layer, data access layer etc. We extend the architectural layer specific meta models to support family concept [4] through modeling of commonality and variability. We use



**Fig 1: Approach for developing extensible & configurable application**

feature model [6] as a mechanism to select the desired member from the family i.e. the desired variant. Thus, we are able to model F and P dimensions of a business application product line [12]. We support extensibility along F and P dimensions by introducing placeholder model elements in the meta model which can be plugged in later with suitable models as new extensions. We have built upon aspect oriented techniques and applied them to models in order to realize our code generator product line [14]. Having tested this approach in practice with varying degree of success we observe this to be an effective approach for supporting business application product line (i.e. developing multiple applications with similar intent for a specific business domain for multiple situations i.e. users). However, we have identified some opportunities for improvement in the current implementation. In this paper, we present another approach to implement code generator product line on the lines of adaptiveness defined here.

The rest of the paper is organized as follows: we present a brief overview of developing adaptive applications using model driven development approach in section 2. Meta model for developing configurable and extensible code generator is presented in section 3. Section 4 presents an example. We discuss some of the related work in section 5. We conclude with a brief summary of early use of the proposed approach in section 6.

## 2. Model driven development of adaptive applications

In model-driven development approach, the various architectural layers of a typical business application are specified in terms of a set of models – each being an instance of a projection or a view of a unified meta model [13]. A, say, architectural layer specific code generator transforms the layer-specific model into the desired implementation on a chosen technology platform e.g. graphical user interface layer in JSP. Models, being at a higher level of abstraction, are easier to specify and are amenable for analysis leading to early detection of errors and prevention of some errors. Also, models can be kept independent of implementation technology platforms with model-based code generators that incorporate proven design and architectural

patterns, deliver increased productivity, uniformly high quality, and retargetability.

Such specification-driven approach imparts adaptiveness to a certain extent as concerns related to D, A and T dimensions are separated from those related to F and P dimensions. For example, the same application specification, i.e. business functionality and business process concerns, can be delivered into multiple implementations that differ in terms of design decisions, architecture and technology platform of choices using different code generators. Thus adaptation related to D, A and T dimensions completely rely on adaptiveness of code generation process.

Fig 1 describes the process for deriving a purpose specific application from an application family specification. A deployable application for deployment choice  $\langle D_1, A_1, T_1 \rangle$  can be derived from application family specification by configuring application family to the desired application model (i.e. specification for application 1) and then transforming it into a deployable implementation using the appropriate code generator (i.e. code generator 1). The same application can be retargeted to a different deployment choice (i.e.  $\langle D_2, A_2, T_2 \rangle$ ) by using different code generator (i.e. code generator 2). Thus, veracity of technology platforms leads to development of multiple code generators even for the same application specification. A code generator product line [14] addresses this problem as follows:

- A code generator is a set of building blocks where each building block is a model to text transformer,
- Model to text transformation process generates fragments of code corresponding to application models, and also the weaving specification to compose the fragments, and finally
- Weave the generated code fragment as per the weaving specification.

We found, using this technique in MDD code generation raises several issues. For example - a) Lack of traceability from aspect to implementation affects readability, debugging and testing of the generated code as aspects are woven together in byte code, b) other issues related to (a), e.g. decomposition of system test cases and aspect level testing, computation of change impact of an application level change in aspect and c) tooling issues for using weaving tools for industrial-strength applications [14].

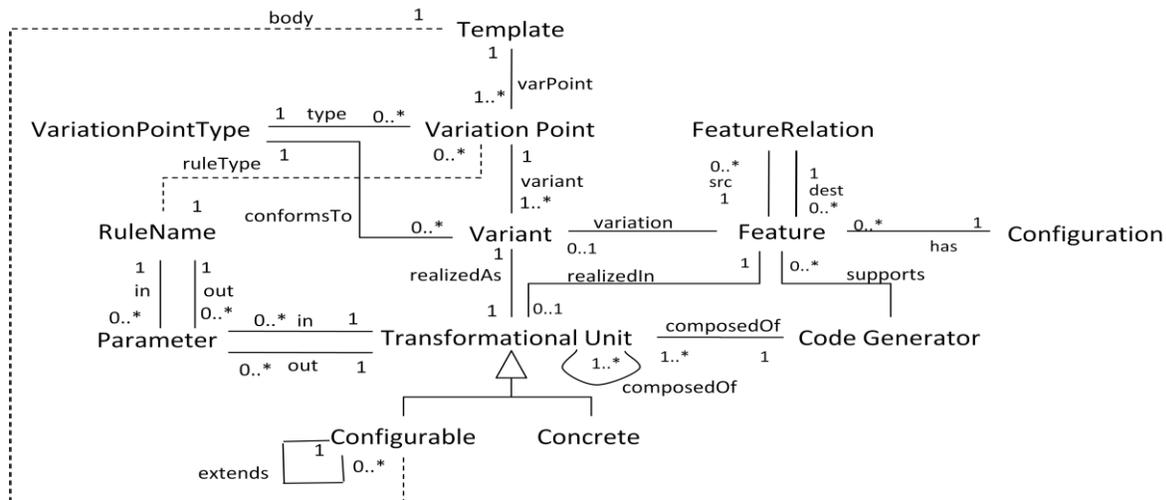


Fig 2: Meta model of extensible & configurable code generator

### 3. Extensible and configurable MDD code generator

A product line groups together a set of products (or product variants) which share a set of features (i.e. common part) and have distinguishing features (i.e. variations) [4,6]. Producing a purpose specific product can be seen as selecting from the available variants at each variation point. We model a family of code generators with a set of core and variable features. We capture a) feature commonalities with placeholders for extensions and variations, b) variability - a specific pattern for variations that can be plugged in at appropriate variation points, and c) extensibilities - a pattern for extending features as new variants of a code generator (code generator family). Fig. 2 shows the proposed code generator product line meta model describing commonality, variability and extensibility patterns for our code generator family. A brief description of the meta model follows:

- A code generator is composed of several composable transformational units. Each transformational unit encodes model-to-text (MTT) transformation rules.
- Transformational unit can be of two types - concrete and configurable.
- Concrete transformational unit encodes simple MTT rules without any placeholder for extension and variation.
- Configurable transformational unit is essentially MTT rules with placeholder for extension and variation. Meta model is extended to specify the extensions and variations as follows:
  - o Body of a configurable transformational unit is specified using *Template*.
  - o Each template can have multiple variation points (placeholder for variations).
  - o A set of variants, which are realized as transformational unit, can fit into each variation point.
  - o Having atleast one variant for each variation point is necessary for defining well-formed transformational unit.
  - o The meta model elements VariationPointType and RuleName (associated with parameters) is

introduced to verify the semantic and syntactic correctness of the fitment between VariationPoint and Variant.

- A code generator product line can declaratively specified in terms of a set of features where each feature is associated with a transformational unit through realizedIn association.
- Configuration operation is equivalent to selecting one amongst the available variants for each variation points, i.e. selecting features conforming to FeatureRelations. FeatureRelation describes all possible feature relations [6], i.e. exclusion, inclusion, optionality, dependency, etc. The selected set of features, also called as feature configuration, must be internally consistent - whatever the definition of consistency may.
- Extension means adding a new variant for an existing variation points (i.e. adding variant of an existing feature) or adding a new Template by extending a transformational unit (i.e. adding new feature).

Essentially a code generator product line is a composite template whose behavior (i.e. transformation capability) is determined by a specific configuration which is conforming to feature relationships. We use an imperative model-to-text transformation specification language (OMGen [11]) for specifying transformation rules of a transformational unit as in our experience programmers are more comfortable with imperative languages than declarative languages. However declaration language, such as [8], can be used as specification language of transformational units. OMGen language provides for model navigation, model manipulation and text writing through familiar constructs such as IF, FOR EACH, PRINT etc. It also provides a syntactical short cut for PRINT construct through ':' operator. The ':' Operator is used for specifying model navigations and '\$' to distinguish model elements from normal text. It supports the notion of MODULE to organize transformation specifications into a set of reusable units called FUNCTIONS. We use Functions for implementing transformational unit, and extend OMGen language with a special keyword VARIATIONPOINT to denote variation point for a Function.

Earlier AOP based approach for realizing code generator product line used declarative model to text transformation specification

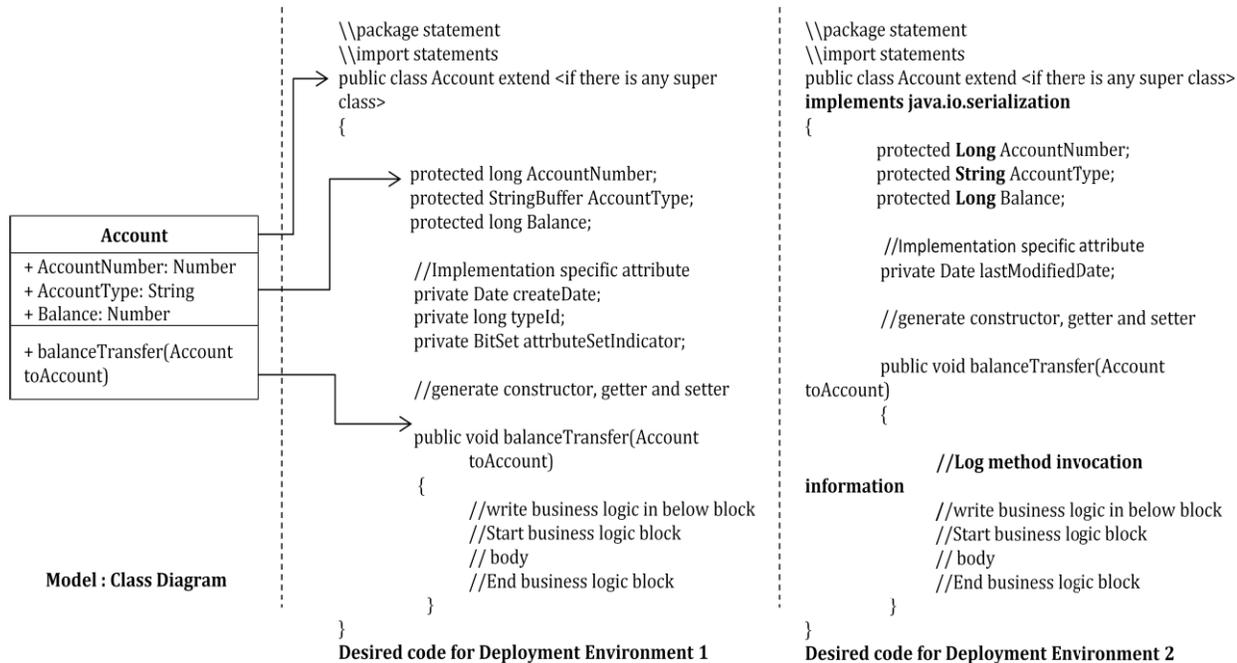


Fig 3. Example of variation in code generation - A class diagram to Java class

language, and was, in essence, a composition mechanism that relied on OO techniques like overriding to implement variability and inheritance to implement extensibility. The new approach uses an imperative OO language to specify model to text transformations and special syntax to denote placeholders i.e. variation points and extension points. This is identical to specifying variability and extensibility in business functionality and business processes. Imperative nature of specification language, and uniformity of mechanism for specifying variability and extensibility along all dimensions is the salient feature of the new approach.

#### 4. Example

In this section, we illustrate our approach using an example - a simple code generator for generating java code from class model. Fig 3 shows a class diagram of a single class, *Account*, and desired code for two deployment scenarios, i.e. deployment environment 1 and deployment environment 2. As shown in the figure, the desired code patterns are similar for both scenario, i.e. an implementation of java classes with package statement, import statement, class header, modeled attribute, implementation specific additional attributes, constructor, getter and setter, method signatures with placeholder for method bodies. However, different deployment environments desire a little different code, e.g. environment 1 desires primitive data types for each modeled attributes (of primitive data types), and a specific set of implementation specific attributes (i.e. *createDate*, *typeId*, *attributeSetIndicator*), whereas environment 2 desires java objects for all modeled attributes and different set of implementation specific attributes. In addition, environment 2 desires code for logging method invocation information for each method. As requirements are different for these two environments, different code generator could be a possible solution for supporting these

environments. However one can think of parameterized code generator with all possible options encoded within code generator implementation. But any extensions to such parameterized code generator might result into opening up entire code generator implementation and configuration could be unfeasible unless they are decided at design time. This limits the extensibility and configurability of a code generator to a large extent.

We specify a code generator family as an instance of proposed meta model. Fig 4 depicts the model of a code generator family that supports code generation for environment 1 and environment 2 of the example described in Fig.3. As shown in the figure, code generator family, *Class2Java*, is composed of a configurable transformation unit, *class2JavaGen*. *Class2JavaGen* is further composed of concrete transformation units - *printSuperClasses*, *importStatementGen*, *packageStatementGen*. Essentially, these transformation units form the core (fixed part) of *Class2Java* code generator. *Class2JavaGen* has four variation points - *printAttributeType*, *printInputParameters*, *printImplementationSpecificAttribute* and *printLogInformation* to specify desired variations. The set of transformational units, shown in transformational units 2 of figure 4, are the available variants of supported variation points. For example - *printAttributePrimitiveType* and *printAttributeObjectType* are possible variants of *printAttributeType* variation point. Model describes the supported features of *Class2Java* code generators and feature relationships. The feature model described in Fig 4 specifies the following relationships - a) *primitiveType* and *objectType* are the alternate feature of *AttributeType*, similarly implementation choice 1 and implementation choice 2 are the available options of *ImplementationOptions* feature. b) *MethodLevelLogging* is an optional feature of *Class2Java* code generator, c) selection of implementation choice 1 is mandatory if *primitiveType* of *AttributeType* is selected. As shown in the figure, model describes the associations between supported feature and transformational unit that encodes the feature using

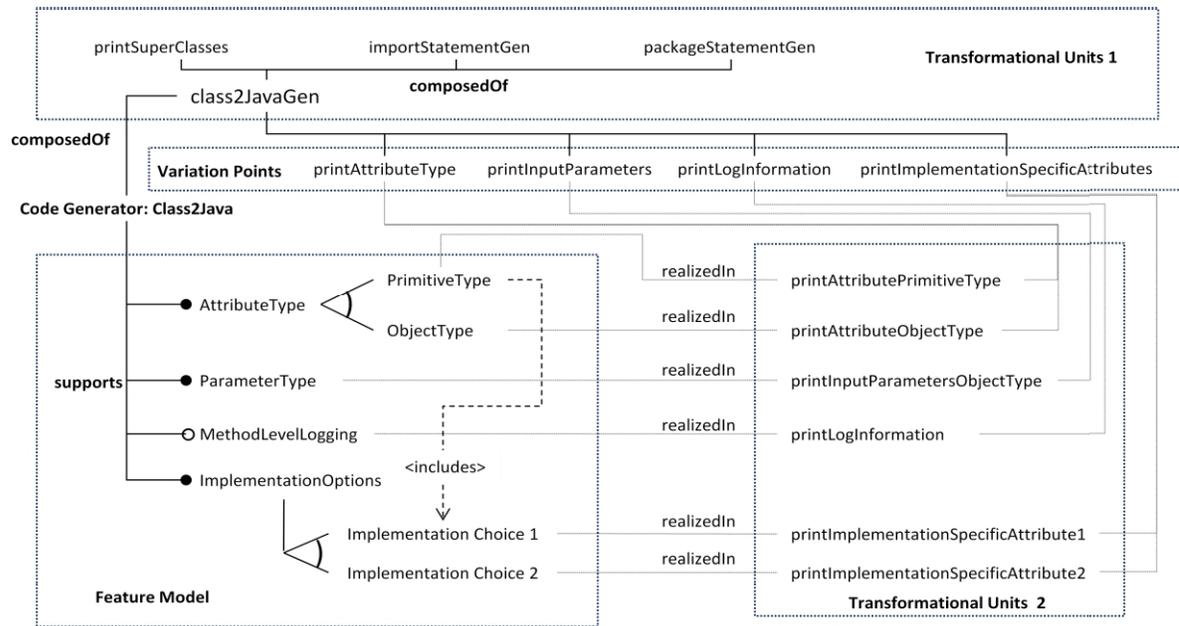


Fig. 4 Model of an extensible & configurable code generator

realizedIn associations. The OMT specification of `class2JavaGen` is depicted in Fig 5. Fixed `composedOf` associations are encoded as `IMPORT MODULE` and `EXTERN FUNCTIONS`. The extended keyword `VARIATIONPOINT` is used for specifying all variation points of the function (`Class2JavaGen`).

A model of code generator family and organized transformation rules (encapsulated within transformational unit) improve extensibility and configurability capabilities into manifolds. For example, a specific configuration can serve code generator for a purpose specific solution; e.g. a configuration selecting `PrimitiveType` and `Implementation Choice 1` of `Class2Java` code generator can produce the desired code for environment 1, and configuration selecting `ObjectType`, `MethodLevelLogging` and `Implementation Choice 2` features can produce desired code for environment 2. Similarly the code generator model can be extended by i) introducing new variants of an existing variation point by adding new transformational unit – a mixed data type for modeled attribute, ii) replacing fixed `composedOf` association by variation point and providing possible variants – defining `importStatementGen` as variation point and providing variants, and/or iii) associating new template with different set of variation points for a transformational unit – a new template `ClassToJavaGen` or `Implementation Choice 1`, etc.

## 5. Related Work

Several approaches for managing variability in software product lines have been proposed in literature. A proposal for modeling variability in software families with UML using the standardized extension-mechanisms of UML is presented in [7]. A variation point model that allows user or application engineer to extend components at pre-specified variation points is proposed in [3]. A conceptual model for capturing variability in a software product line is presented in [1]. These three support the notion of variation

point only and that too at modeling level. A general template-based approach for mapping feature models to concise representations of variability in different kinds of other models is presented in [5]. However it can address agility and adaptability of F and P dimensions only. The same concept is used in generation of model-based code generators from their high-level specifications [14].

Use of product line concept in model-based code generator is not new. It has been reported in several MDD literature [4, 7, 9, 15] to address the agility and adaptability of D, A and T dimensions. These approaches are based on aspect-oriented technique [2] and implemented using declarative language for model-to-text transformation [8]. The essential limitation of AOP based approach is that it works with the languages that support aspect weaving as this technique generates code fragments that need to be weaved for deriving a deployable artifacts whereas, our solution is based on OO techniques, uses an imperative language and feature model notation, and does not require special composition machinery. It uses parameterization and composition techniques to derive a consistent code generator similar to traditional MDD code generators that generate deployable code.

## 6. Conclusion

We are working toward a framework for developing adaptive business application product line using model driven development techniques. Use of model driven techniques for developing such business applications is argued as a preferable option as platform independent specification can be retargeted to technology platform of choice through a code generation process. In our experience, use of product line architectures enables agility and adaptability in business functionality and business processes, i.e. F and P dimension of the business application for typical model-driven approaches. In this paper we argued the use of product line concept in F and P dimensions is not sufficient to achieve the

```

MODULE class2JavaGen;
IMPORT MODULE importStatementGen;
IMPORT MODULE packageStatementGen;
IMPORT MODULE printSuperClasses;

EXTERN FUNCTION importStatementGen (Class);
EXTERN FUNCTION packageStatementGen(Class);
EXTERN FUNCTION printSuperClasses (Class);

VARIATION POINT printAttributeType(Type);
VARIATION POINT printInputParameters (Operation);
VARIATION POINT printImplementationSpecificAttributes();
VARIATION POINT printLogInformation() [OPTIONAL];

DEFN FUNCTION public ClassToJavaGen(cls: Class)
{
    packageStatementGen($cls);
    importStatementGen();
    : public class $cls.Name extend
    printSuperClasses (cls)
    :{
        FOREACH attribute IN cls.attribute
        {
            :protected \c
            printAttributeType(attribute.type) \c
            : $attribute.Name;
        }
        printImplementationSpecificAttributes();
        FOREACH operation IN cls.operation
        {
            :public \c
            printAttributeType(operation.type) \c
            : $oper.name ( \c
            printInputParameters(operation)
            : ) {
                printLogInformation();
                ://Start business logic block
                : //body
                ://End business logic block
            :}
        }
    :}
}

```

**Fig 5: Specification of configurable transformational Unit**

desired agility and adaptability - it is as well critical to addresses D, A and T dimensions. We presented an approach for developing configurable and extensible code generator product line. A meta model for specifying the product line and suitable extension to MTT specification for describing transformation rules in modularized form are presented. We demonstrated how a model (instance of code generator meta model) and transformational unit specification can be configured to derive a purpose specific code generator through an illustrative example.

We have used this approach to specify our MDD code generator product line resulting in time and effort savings [10]. Supporting a new deployment environment is now mostly a configuration activity rather than implementing a new code generator. Supporting new features in code generator family is much simplified process as one has to define transformational unit for the new feature with suitable associations with existing features and transformational units. Early observations of testing the approach on real life industrial strength problems are encouraging but there is still some way to go.

## 7. Reference

- [1] Felix Bachmann, Michael Goedicke, Julio Leite, Robert Nord, Klaus Pohl, Balasubramaniam Ramesh and Alexander Vilbig. A Meta-model for Representing Variability in Product Family Development. Software Product Family Engineering, volume 3014 of LNCS, pages 66-80, Springer, 2004.
- [2] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Videira Lopes, Jean-Marc Longtier and John Irwin. Aspect oriented programming. ECOOP'97 LNCS 1241, pp 220-242. Springer-Verlag. June 1997.
- [3] Hasan Gomaa, Diana L Webber. Modeling Adaptive and Evolvable Software Product Lines Using the Variation Point Model. 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Volume 9. Page: 90268.3
- [4] K Czarnecki and U Eisenecker, Generative programming methods, tools and applications, Addison-Wesley, 2000.
- [5] K. Czarnecki and M. Antkiewicz. Mapping features to models: A template approach based on superimposed variants. Generative Programming and Component Engineering, Volume 3676 of LNCS, pages 422-437. Springer, 2005.
- [6] K Kang, S Kohen, J Hess, W Novak and A Peterson, Feature-orientation domain analysis feasibility study, Technical Report, CMU/SEI-90TR-21, November 1990.
- [7] M Clauß, I Jena. Modeling variability with UML. GCSE 2001 Young Researchers Workshop, 2001.
- [8] MOF Model to Text Transformation Language (MOFM2T), 1.0. January 2008, [www.omg.org/spec/MOFM2T/1.0/](http://www.omg.org/spec/MOFM2T/1.0/)
- [9] Markus Voelter, Iris Groher. Handling Variability in Model Transformations and Generators. 7th OOPSLA Workshop on Domain-Specific Modeling, OOPSLA, Canada, 2007
- [10] MasterCraft – Component-based Development Environment. Technical Documents. Tata Research Development and Design Centre. <http://www.tata-mastercraft.com>, <http://www.tcs.com/SiteCollectionDocuments/Brochures/TC-S%20Code%20Generator%20Framework.pdf>
- [11] OMGen Reference manual, version 1.5, Technical Document, Tata Consultancy Services, May, 2008
- [12] Souvik Barat and Vinay Kulkarni: Supporting Agile Adaptive Business Services Using Model-based Techniques, 3rd International Workshop on Service Oriented Computing collocated with the 16th IEEE International Conference on High Performance Computing (HiPC 2009), December, 2009
- [13] Vinay Kulkarni, Sreedhar Reddy: Model-Driven Development of Enterprise Applications. UML Satellite Activities 2004: 118-128
- [14] Vinay Kulkarni, Sreedhar Reddy, An abstraction for reusable MDD components: model-based generation of model-based code generators. GPCE 2008: 181-184
- [15] X. Liu , Y. Feng , J. Kerridge, "Achieving Dependable Component-Based Systems Through Generative Aspect Oriented Component Adaptation", In Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC'06), Chicago, USA, September 2006

# Feature based Structuring and Composing of SDLC Artifacts

Nishigandha Hirve

TRDDC

Pune 411 013, India

nishigandha.hirve@tcs.com

Tukaram Muske

TRDDC

Pune 411 013, India

t.muske@tcs.com

Ulka Shrotri

TRDDC

Pune 411 013, India

ulka.s@tcs.com

R. Venkatesh

TRDDC

Pune 411 013, India

r.venky@tcs.com

**Abstract**— Product organizations often need to develop variants of the same basic product. All the product development life-cycle artifacts from requirements documents to testing artifacts have to be developed for each variant. Storing these artifacts as core assets and structuring them as units that can be composed to get the final artifacts can greatly reduce the cost and improve the quality of the resultant product. This paper proposes a structural alignment of core assets corresponding to the various phases of software development to features of the product family and aspect weaving as a core assets composition operator. Aspect Oriented Programming (AOP) is used for code and an aspect oriented extension to XML is used for other artifacts. These ideas are validated through a prototype toolset that captures the complete Software Development Life Cycle (SDLC) process and allows the user to model the commonalities and variations as features, associate them to core assets and build the product automatically. This paper presents our approach and the details of the toolset along with a case study of a Library System.

## I. INTRODUCTION

Planned software reuse as recommended by Software Product Line Engineering (SPLE) [1] can considerably improve product development cost and time. SPLE and Feature Modeling [2] identify variations and commonalities between products in a product line and model them as features [3]. This paper proposes the structuring of the different software development artifacts so as to align them with the structure of the feature model. Once this is achieved, composition similar to weaving in aspect oriented programming can be used to derive artifacts at the product level from artifacts at the feature level.

Development of each product from the feature model follows the entire SDLC where the resultant requirements document has to be reviewed and approved by relevant stakeholders and the final product has to be tested for this specific combination of features. Therefore, just producing the final product by composition will not suffice and there is a need to generate all artifacts required by SDLC using similar composition. This paper adapts the idea of Aspect Oriented Programming (AOP) applied to product development, to other non-graphical SDLC artifacts like documents and test harness. Documents are composed using an aspect oriented extension to XML and test harness is composed by writing the test harness in a programming language and using AOP.

The prototype toolset presented here captures complete SDLC process from product requirements gathering to its release. This toolset mainly consists of two parts, core assets repository and product generator. Creation of core assets repository involves identifying features, modeling them using ASADAL [4] and identifying, developing, and maintaining feature-based core assets. ASADAL allows selection of features required for product development. Product generator identifies and composes the required core assets together into a final product.

We demonstrate the toolset and its benefits using library domain case study. This case study illustrates the identification of variations and commonalities in a library product family and their modeling as a feature model. It also illustrates the structuring of other core assets consisting of various development artifacts so that they are aligned with the feature model. AspectJ – AOP implementation for Java [5] is used as the core assets composition operator to handle these variations between product variants for the artifacts implemented in Java. AOP has been chosen as it provides a very general and flexible set of composition operators. A XML weaving tool is used to implement a composition operator for the document artifacts.

**Related work:** Various other toolsets are available for generating products from a product family, customizing products [7], and configuring product line features [8]. Kyo C. Kang has combined feature oriented analysis with AOP [9]. Studies have shown that feature models can be used for product derivation [6]. Also, feature modeling is supported by several tools [10],[11].

Compared to the tools listed above, our prototype toolset provides support for different artifacts like documents and code, required in a typical SDLC. It mainly investigates AOP based techniques to compose both code and documents.

## II. TOOLSET

Figure 1 shows the detailed toolset architecture. The toolset organization is mainly divided into Core assets repository and Product generator.

### A. Core Asset Repository

The core asset repository stores all the assets including feature models and associated documents and code.

### 1) Feature Modeling in ASADAL.

We use ASADAL tool for feature modeling that supports FODA notations and allows selection of features for product development. We associate the feature-specific core assets information to the description field in ASADAL using a set of keywords. The keyword set includes keywords such as SOURCE, CONFIG, XML, and TESTCASES.

### 2) Structuring and Implementation of Core Assets

The required set of core assets is identified from feature model and mandatory, optional and alternative features are implemented separately. The core assets for optional features are implemented as aspects in order to manage variations found in the product family. Whenever an optional or alternative feature is selected, its corresponding aspect gets included in the source list. These aspects modify the functionality of the mandatory feature to provide the product behavior with respect to the selected feature. This type of implementation also provides traceability from features to their implementations.

**Code Composition:** The composition of code related core assets is similar to the other AOP based applications.

**Test Harness Composition:** As the test harness is written in a programming language using AOP, its composition is exactly similar to the code composition.

**Documents Composition:** AOP extensions to XML help to manage documentation at the feature level. This requires structuring of the feature based documents using XML and additional AOP tags. The organizational alignment between feature model and document states that document is either a base document (associated with mandatory feature) or an aspect document (associated with optional, alternate and or feature).

The base document has a standard XML structure and consists of tags, attributes with values and content. Each path from the root to any tag in the document specifies a join point. Aspect documents also have standard XML structure specifying advices to modify the contents at a join point in the base document. An advice is also an XML path from the root to an advice tag and optionally includes the child. The advice tags can be either “after”, “before”, “replace” or “delete”.

The document composition algorithm is:

Let  $p = r, t_1 \dots t_n, a, c$  be a path in the aspect document where

- $r$  is the root node,
- $a$  is the advice node having tag “after”, “before”, “replace” or “delete”
- $t_1 \dots t_n$  are nodes in the path from  $r$  to  $a$  in the aspect document
- $c$  is the child tag of  $a$  and is of the form  $\langle ctag, id=val \rangle$

For each such path  $p$  in the aspect document, it matches a corresponding path  $p1 = r1, t1 \dots tn$ , in the base document such that the tags of  $t1 \dots tn$  are identical in both  $p$  and  $p1$  and if any of  $t1 \dots tn$  has an attribute  $id$ , then the value of the attribute also should be the same. For each such match if the tag of  $a$  is ‘Before/After’ then  $c$  is added as the first/last child of  $tn$  respectively in the base document. In the case of Replace/Delete tag, the child of  $tn$  with the same tag and value of attribute  $id$  as  $c$  is replaced/deleted in the base document

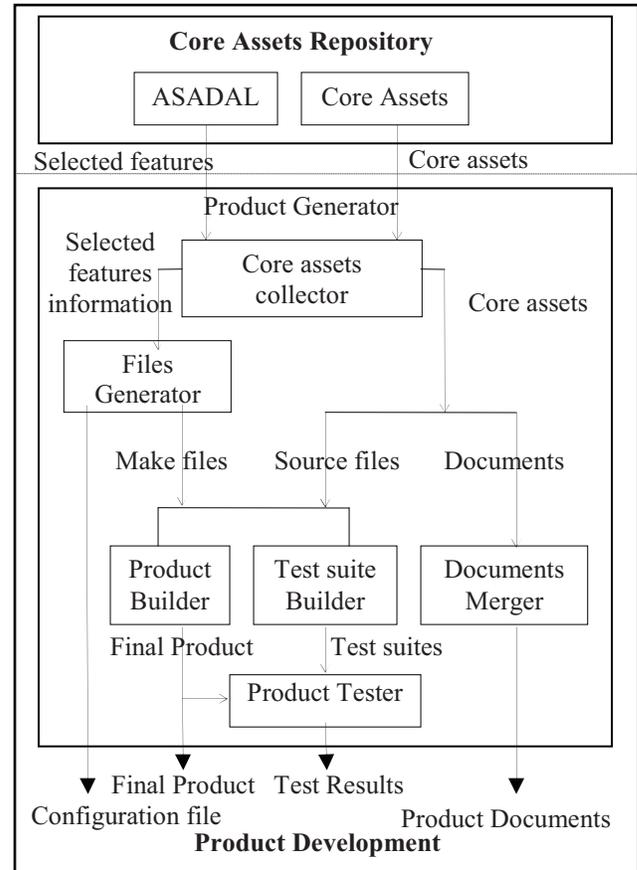


Figure 1. Toolset architecture

The values of attributes other than id are not considered for the match.

### B. Product Generator

When a new product is to be built, required features are selected from feature model and exported as an ascii file. The exported file also includes associations between the selected features and corresponding requirements, test case, implementation and test harness aspect file names. The *core assets collector* extracts the required files from the source code and document repository. Once all the files have been extracted the *File Generator* generates a makefile that invokes aspectJava and the java compiler to build the application as well as the test harness. The *product* and *test suite* builder then executes these makefiles. The test harness is then run on the application and the results logged.

Orthogonally the *Document merger* invokes the XML weaver on all the test case and other base and aspect documents extracted by the *core assets collector* to generate requirements and test case documents for the final product.

## III. LIBRARY SYSTEM CASE STUDY

This section illustrates the features of the toolset using a Library System product family case study, where a particular set of requirements is considered.

### A. Library System Domain

Library systems maintain books and members and offer services to members to reserve, borrow and return books. There are two types of members (Ordinary and Privileged) and three types of books (Journal, Technical book and Magazine). The number of books to borrow and days allowed are determined by the member and book types. Library system may allow members to put claims on the books and the number of claims varies as per member type. There may be a late return fine. Although we have implemented several features as part of our case study, only few of them are described in detail here in the interest of brevity.

### B. Modeling and Implementation of Library Systems

Using the concepts of SPLE and feature modeling, we study the Library system domain and systematically elicit the variations – Member type, book type, book claim, number of books and days allowed, Late return fine etc.

#### 1) Feature Modeling

The features identified from variations and commonalities are modeled in the feature model, using FODA notations, as shown in Figure 2. We have shown only a small set of selectable features in Capability and Implementation Layer for clear representation. Mandatory features such as Member and Book Maintenance are omitted from the feature model.

#### 2) Library System Core assets

The various core assets for Library system can be categorized into – primary assets (feature model, requirements, class diagrams), components (code implementations such as core service component, claim processing component), test harness assets (test cases’ implementations, test data, test results), various tools used (Tomcat, ASADAL, product generation tools), configuration files, etc.

#### 3) Library System Implementation

We use client-server architecture to implement the library system with Java (JDK1.5.0), JSP, AspectJ (release 1.5) [12] and Tomcat [13]. These implementation details will affect the feature representations. A feature will have its corresponding functionality component represented in implementation layer. For example, the core services are implemented by core service component which is further linked to Java implementation to specify its source files. The Java implementation feature is further divided into client and server components.

### C. Core Asset Composition Technique

The source implementation of library functionality has been done in AspectJ with variants of each feature implemented as aspects. AspectJ has also been used to implement the test harness. Since the implementation of functionality is similar to other aspect oriented applications, in this paper we only describe the test harness implementation in detail.

#### 1) Test Harness Composition

The core assets corresponding to test harness are implemented similar to other basic functionality components. The variable part in the test cases is put into the aspects which modifies the behavior of the mandatory test cases.

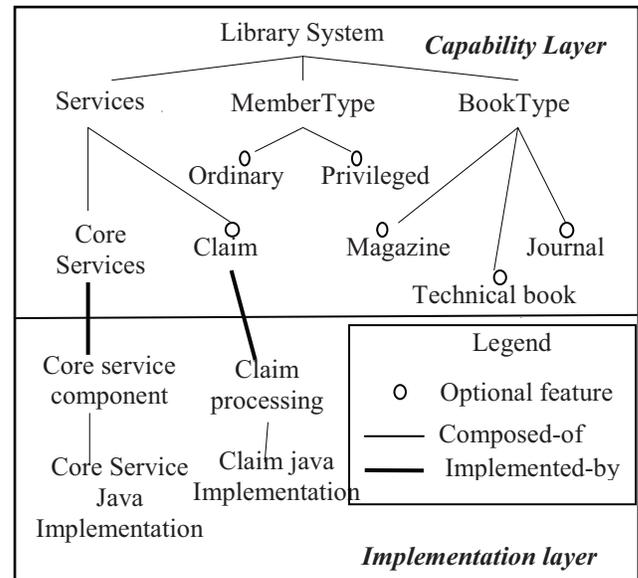


Figure 2. Library system feature model

**Example:** The test cases for the borrow operation are implemented in BorrowTests class. We consider following test case, in pseudo code, for book unavailable scenario.

```
class BorrowTests{
.....
Function chkPostCondition () {
    Check_for_msg "The requested book is not
    available in the Library";
    return true if successful, else false;
} }

```

In the presence of claim, if the requested book to borrow is unavailable, it gets reserved for the member. Thus, when claim is selected, the borrow operation behaves differently and above test case needs modification to check for claim object. The required modification to borrow test case, due to claim, is achieved using below piece of aspect where method chkPostCondition is overridden by the below aspect.

```
around():BorrowTest.chkPostCondition();{
    check_for_msg "The requested book is not
    available in the Library. The same book is
    reserved for you."
    check_for claim_object;
    return true if successful, else false;
}

```

#### 2) Documents Composition

Various documents are structured with respect to features and formatted as per AOP concepts. For example, for better reuse, test case documents are maintained at feature level for Core services and optional Claim feature separately. These documents will be meaningful only if the Claim feature is selected. This necessitates their formatting in a particular manner to manage variability which includes specifying the required and related join points, pointcuts and advices in an AOP way. A join point from SimpleBorrow.xml, the test case document for Core services feature is shown below.

```
<TestCase id=2> <CheckResult>
<Check id="a"> Check for return message: "The
requested book is not available in the
Library".</Check>
</CheckResult> </TestCase>
```

The elements `TestCase` and `CheckResult` specify the join point. In presence of `Claim`, the information in check results needs to be modified as the test case behaviour is changed. Additional checks for claim message and claim object are added using following advice from `BorrowClaim.xml`.

```
<TestCase id="2"> <CheckResult> <After>
<Check id="b">Check for return message: "The
same book is reserved for you"</Check>
<Check id="c">Check For loan object for Title1
with member 13 and bookcopy of Title1</Check>
</After> <CheckResult> </Case>
```

Here, the ‘After’ element specifies the After advice to add the extra checks in the target test case. The test case mapping is achieved using common value for “id” attribute. Thus the information with elements and attribute values specifies the pointcut. When `Claim` feature is selected, the XML documents composition occurs and the resulting test document for the above mentioned scenario appears as shown below.

```
<TestCase id=2> <CheckResult>
<Check id="a">Check for return message: "The
requested book is not available in the
Library."</Check>
<Check id="b">Check for return message: "The
same book is reserved for you"</Check>
<Check id="c">Check For loan object for Title1
with member 13 and bookcopy of Title1</Check>
</CheckResult> </TestCase>
```

Requirements and design documents are similarly structured and XML weaving is used for composition.

#### D. Product Development

As the entire required core assets base is in-place in repository, product development simply becomes the process of picking-up and combining the related components. We can easily develop Library systems with the required features and release the tested library system to the market.

## IV. CONCLUSION

We have demonstrated the feasibility of aligning the structure of all software development assets with feature

models and the use of an aspect composition operator to derive a specific artifact corresponding to a specific product. Successful application of the proposed approach requires the documents and code to be structured in a way that is amenable to aspect composition. In particular requirements and design documents should be structured as a tree with the leaves being simple elements.

We believe the ideas presented can be extended to larger projects.

## ACKNOWLEDGEMENT

We sincerely thank Suparna Soman for contributing through multiple reviews to make this paper better.

## REFERENCES

- [1] Clements, P., Northrop, L., “Software Product Lines: Practices and Patterns”, 2002
- [2] Kwanwoo Lee, Kyo C. Kang, and Jaejoon Lee., “Concepts and Guidelines for Feature Modelling for Product Line Software Engineering, Software Reuse: Method, Techniques and Tools”, ICSR-7, April 2002
- [3] Jaejoon Lee, Dirk Muthig, “Feature-oriented variability management in product line engineering”, Communications of the ACM December 2006/Vol. 49, No. 12.
- [4] Kyungseok Kim, Hyejung Kim, Kyo C. Kang, “ASADAL - a tool system for co-development of software and test environment based on product line engineering” ICSE '06
- [5] Ramnivas Laddad, “AspectJ in Action: Practical aspect-oriented programming”, 2003
- [6] Spinczyk, O., Papajewski, H., “Using Feature Models for Product Derivation”, SPLC 2006
- [7] Mazen Saleh, Hassan Gomaa, “Separation of Concerns in Software Product Line Engineering”, ICSE 2005
- [8] Andreas Hein, John MacGregor, Steffen Thiel, “Configuring Software Product Line Features” ECOOP 2001
- [9] Kwanwoo Lee, Kyo C. Kang, Minseong Kim, “Combining Feature-Oriented Analysis and Aspect-Oriented Programming for Product Line Asset Development”, SPLC 2006
- [10] Alain Forget, Dave Arnold, Sonia Chiasson “CASE-FX: feature modeling support in an OO Case tool”, OOPSLA 2007
- [11] Thomas Leich, Sven Apel, Laura Marnitz, Gunter Saake, “Tool support for feature-oriented software development: featureIDE: an Eclipse-based approach”, OOPSLA 2005
- [12] AspectJ (release 1.5), available at as on March 09, 2010, <http://www.eclipse.org/aspectj/index.php>
- [13] Apache Tomcat URL as on March 09, 2010, <http://tomcat.apache.org/>

# Distributed and Adaptive Execution of Condor DAGMan Workflows

Selim Kalayci<sup>1</sup>, Gargi Dasgupta<sup>2</sup>, Liana Fong<sup>3</sup>, Onyeka Ezenwoye<sup>4</sup>, S. Masoud Sadjadi<sup>1</sup>

<sup>1</sup>Florida International University, Miami, FL, USA, {skala001,sadjadi}@cs.fiu.edu

<sup>2</sup>IBM India Research Lab, New Delhi, India, gdasgupta@in.ibm.com

<sup>3</sup>IBM Watson Research Center, Hawthorne, NY, USA, llfong@us.ibm.com

<sup>4</sup>South Dakota State University, Brookings, SD, USA, onyeka.ezenwoye@sdstate.edu

**Abstract**— This paper presents a decentralized execution approach to large-scale workflows on multiple resource domains. This approach includes a low overhead, decentralized runtime adaptation mechanism that improves the performance of the system. A prototype implementation based on standard Condor DAGMan workflow execution engine, does not require any modifications to Condor or its underlying system.

**Keywords:** Decentralized adaptive workflow execution.

## I. INTRODUCTION

Large-scale applications, in the form of workflows, may require a coordinated usage of resources spreading across multiple administrative domains [3, 4]. Current systems mostly provide a centralized approach to the execution of such workflows across resource domains. This may induce scalability problems for large workflows executing in multiple domains. In our previous study [5], we provided a solution to decentralize the execution of workflows to address these issues. Another desired capability with the execution of large-scale workflows in dynamic distributed environments is the runtime steering of the workflow in order to meet the desired Quality of Service objectives. Typically, scientific workflows are composed as abstract workflows, in which tasks and data are specified without bindings to the actual physical resources. Workflow management systems [6, 11] perform the execution of abstract workflows in two basic steps. First, tasks and data comprising the workflow are mapped on to the available physical resources [1, 9]. This results in a concrete workflow, which can then be executed. The dynamic nature of the resources, and the unpredictable behavior of the workflow tasks, may require variations in the original workflow execution plan to meet the desired Quality of Service objectives.

Most existing proposals [8, 7, 2] tackle this problem by re-mapping the rest of the workflow, halting the execution of the original plan and enacting a new execution plan. This costly approach requires the interruption of the whole workflow execution even if variation of the workflow execution plan does not require changes to the execution plan of all tasks. In this paper, we propose a new adaptation approach that is low-overhead, because it operates at the sub-workflow level and requires only the involvement of the affected tasks. Our prototype implementation focuses on adapting an existing

workflow management system to make it more resilient to changes in its execution support. Hence, in this paper, we present an adaptation mechanism to a widely-used centralized workflow management system called Condor DAGMan [12]. The adaptation mechanism we provide is transparent, which means that it does not require modifications to the Condor DAGMan infrastructure.

The technical contributions detailed in this paper include:

- The novel concept of *adaptive sub-workflow* during runtime of workflow execution in peer domains.
- The design of *adaptive sub-workflow* run-time support with event notification across peer domains.
- The prototype implementation of *adaptive sub-workflow* as a transparent enhancement to a conventional workflow execution engine.

The rest of the paper is organized as follows. In Section 2, a background on workflows and our decentralization approach is provided. In Section 3, we present our adaptive workflow execution approach. In Section 4, we provide implementation-level details. In Section 5, the related work is reviewed. In Section 6, the paper is concluded with some future work.

## II. BACKGROUND

In this section, we briefly introduce the concept of scientific workflow management. We provide an overview of our decentralized design of workflow management, upon which our adaption of workflow execution is designed.

### A. Application workflow and execution

Fig. 1 (a) shows a very simple workflow represented as a **directed acyclic graph (DAG)**  $G = (V, E)$ , where  $V = \{V_1, V_2, \dots, V_8\}$  is the set of vertices that correspond to the tasks and  $E = \{E_1, E_2, \dots, E_{10}\}$  is the set of edges that correspond to the dependencies between the tasks. Workflow managers execute the tasks in  $V$  in the order specified by the dependencies between them. For example, if there is an edge in the graph from vertex  $V_x$  to vertex  $V_y$ ,  $V_y$  can start its execution only after  $V_x$  has finished its execution.

### B. Decentralized workflow execution

Based on our approach for decentralized workflow execution [5], the home workflow manager (HWM) first

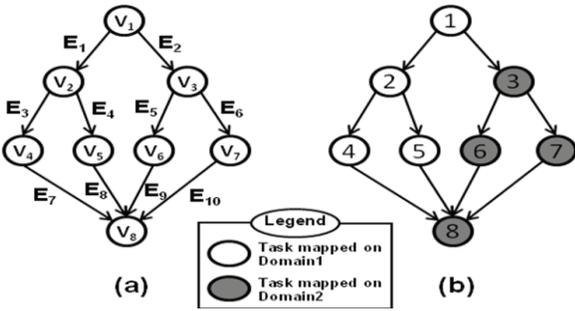


Figure 1. (a) A Sample workflow specification represented as a DAG. (b) Workflow mapped on two domains. For simplicity, the labels for edges are dropped and vertices are merely numbered.

decides on a mapping of workflow tasks to its peer domains. Based on the workflow in Fig. 1 (a), a sample mapping for this workflow is shown in Fig. 1 (b). Tasks are first assigned to domains, and then actual binding of tasks to specific resources is made autonomously by each domain.

Each workflow manager executes only the tasks that have been assigned to it. After a workflow manager executes a task, it checks whether the task has any dependent task(s). If a dependent task is assigned to a different peer, the workflow manager sends a trigger message to the responsible peer to start execution of the dependent task(s).

Fig. 2 illustrates the execution control and peer interactions of the sample workflow, based on the mapping given in Fig. 1 (b). Workflow Manager 1 (WFM1) is the HWM, and WFM2 is the peer workflow manager.

### III. ADAPTIVE EXECUTION

Our adaptive workflow execution approach builds upon the distributed execution mechanism that is briefly explained in Section 2. At runtime, the original workflow execution plan may need to be changed in order to meet Quality of Service objectives. Reasons for change may vary from inaccurate runtime predictions made during the mapping process to dynamic changes in computing environments due to resources being overloaded.

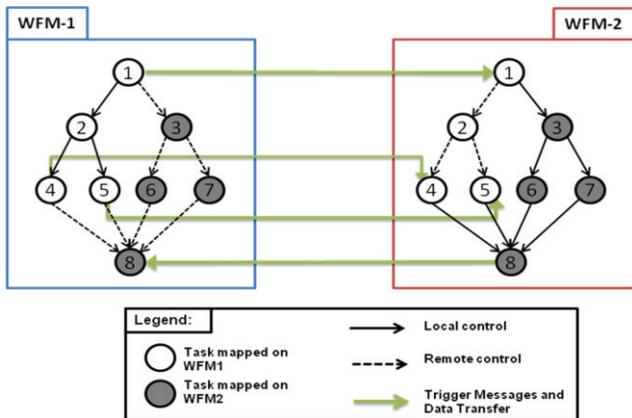


Figure 2. Execution controls and peer interactions during the execution of the sample workflow.

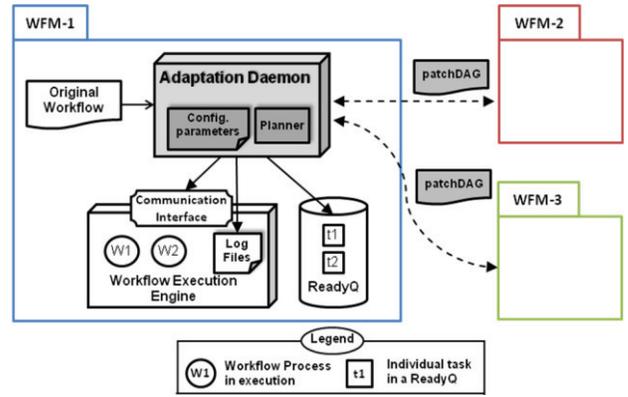


Figure 3. Architecture of the distributed adaptive workflow execution system.

Fig. 3 shows the architectural diagram of our adaptation approach. Gray-colored components and artifacts denote those introduced by us. As our approach is based on peer-to-peer communication and collaboration of workflow managers rather than coming up with a whole new mapping for the entire workflow, we let individual workflow managers detect any problems in their own domain that may prevent QoS objectives from being met. In such a case, tasks may then be migrated to other available domains for execution. The execution of the rest of the workflow is not affected by this adaptation.

#### A. Monitoring and Detection

Each workflow manager independently monitors its resource queues for normal operations, as well as detection of problems and troubleshooting. A backed-up resource queue is almost always an indication of some underlying problem. In this work, we consider *readyQ* that queues all the tasks that are *ready* for execution at the WFM. Under normal operating conditions, tasks from the *readyQ* are steadily dispatched for execution to the underlying scheduler resources. A steadily building up *readyQ* signifies that there is some problem with the underlying infrastructure resources (e.g., resource outages, surge in background traffic, etc). We use *readyQ*-length as the indicator for the onset of congestion and proactively make runtime reconfigurations to deal with this.

*Detecting a congested queue:* Each WFM continually monitors its *readyQ*-length. For every *readyQ*, we assign two thresholds *low* and *high*. The average queue length is calculated, using an exponential weighted moving average, given by:

$$avg_t = (1 - w) * avg_{t-1} + w * qlen_t,$$

where  $qlen_t$  is the length of *readyQ* at time  $t$ , and  $w$  takes values between  $[0,1]$ .

At time  $t$ , the following decisions are taken:

- if the queue length is below *low*, no corrective action is taken.
- if the queue length is between *low* and *high*, we probabilistically pick *some* tasks for remapping. Due to space considerations, we will not explain this topic in this paper.
- if the queue length is above *high*, we pick *all* tasks for remapping.

B. *Planning and Execution*

As mentioned earlier, our adaptation mechanism is based on a peer-to-peer push-based mechanism, without interrupting the execution of the rest of the workflow. When a set of tasks need to be moved from one domain to another, we capture those tasks and the associated trigger messages and create a new workflow, which we call a *patchDAG*. The patchDAG is a fragment of the whole that encapsulates the business logic of the tasks that are selected for migration. This patchDAG is transferred to the proper domain and executed independent of the original workflow. Even though it executes separately from the original workflow, by means of the trigger messages embedded within it, the patchDAG achieves the necessary interactions with the original workflow. This way, execution of the original workflow is sustained without making any changes to the rest of the workflow.

IV. IMPLEMENTATION

In this section, first we will introduce the implementation of decentralizing a centralized workflow specification. Then, we will explain how we achieve adaptation over decentralized execution. Our implementation is based on Condor DAGMan workflow execution engine.

A. *Decentralization*

Listing 1 shows a sample standard Condor DAGMan specification. In a traditional setup, this workflow is orchestrated by a centralized Condor DAGMan instance (in this case, WFM-1).

Job	A	A.submit				
Job	B	B.submit				
Job	C	C.submit				
Job	D	D.submit				
Job	E	E.submit				
Job	F	F.submit				
PARENT	A	CHILD	B	C	D E	
PARENT	B	C	D	E	CHILD	F

Listing 1. Original Condor DAGMan specification for a simple DAG.

The decentralization of an original DAG specification for our Condor DAGMan-based prototype occurs in two stages. In the first stage, HWM (WFM-1) aggregates the original DAG specification with mapping and site-specific contact information. Mapping information denotes the site that is responsible for each task’s execution. Site-specific contact information includes the GRAM (Grid Resource Allocation and Management) [3] end-point reference to facilitate trigger message communication, and the GridFTP [10] contact information to facilitate the transfer of data items among sites. HWM distributes this aggregated DAG specification to all the peer WFMs involved in the execution of the workflow.

The second stage of decentralization process occurs at individual sites upon receiving the aggregated DAG specification. Each WFM modifies its own copy of the DAG specification based on the information received from the HWM. Listing 2 and Listing 3 displays the DAG specifications at WFM-1 and WFM-2 respectively, after the modification. There are 3 basic functionalities involved within this modification process:

1. If a task is mapped on a different domain, this task is labeled as *DONE* in the DAG specification.

2. If a task has child task(s) that is mapped on a different domain, insert a *Post Script* in the DAG specification to synchronize with the child task(s).

3. If a task has parent task(s) that is mapped on a different domain, insert a *Pre Script* in the DAG specification for the reception of a matching synchronization message.

Our Post Script prototype simply generates a file that is specifically named (e.g., file: A\_D to trigger task D at WFM-2) and puts it in the shared folder of the receiving end. The Pre Script simply checks the existence of the specifically named file in the shared folder.

Job	A	A.submit			
Job	B	B.submit			
Job	C	C.submit			
Job	D	D.submit	DONE		
Job	E	E.submit	DONE		
Job	F	F.submit	DONE		
SCRIPT	POST	A	trigger.sh	A_D	A_E
SCRIPT	POST	B	trigger.sh	B_F	
SCRIPT	POST	C	trigger.sh	C_F	
...					

Listing 2. Modified DAG specification for WFM-1.

Job	A	A.submit	DONE		
Job	B	B.submit	DONE		
Job	C	C.submit	DONE		
Job	D	D.submit			
Job	E	E.submit			
Job	F	F.submit			
SCRIPT	PRE	D	synch.sh	A_D	
SCRIPT	PRE	E	synch.sh	A_E	
SCRIPT	PRE	F	synch.sh	B_F	C_F
...					

Listing 3. Modified DAG specification for WFM-2.

Hence, the final set of DAG specifications at each site differs, but the collaborative execution of these DAGs results in the complete and exact execution of the original DAG specification.

B. *Run-time Adaptation*

We provide run-time adaptation through an adaptation daemon running at each site. The adaptation daemon constantly monitors the execution progress of the workflow and the length of ready queue. It also has a thread that listens for requests that might come from other peers.

The specific functionality of an adaptation daemon in case of an adaptation process varies depending on its role. The algorithm employed at the adaptation daemon for the pushing role is shown in Listing 4. Listing 5 shows the algorithm employed at the adaptation daemon for the receiving role. Construction and execution of a carefully planned patchDAG plays a crucial role in the success of our collaborative system.

1. Select the task(s) to push
2. Pick a site among candidates
  - a. if negotiation is successful, move to step 3
  - b. if negotiation is unsuccessful, pick another site, repeat step 2
3. Construct the patchDAG and send it to the receiver site
4. Remove pushed tasks from the local queue (*condor\_rm taskID*)
5. Wait for a rescue DAG to be created by Condor DAGMan
6. Modify the rescue DAG
  - a. label the pushed tasks as DONE
  - b. insert a Pre Script for the task(s) which is mapped locally and is a child of a pushed task
7. Submit the modified rescue DAG to Condor DAGMan.

Listing 4. Pushing algorithm employed within the Adaptation Daemon.

1. Receive the request to push tasks
2. Check the status of the local queue and respond to the request
  - a. if response is negative, go back to step 1
  - b. if response is positive, go to step 3
3. Receive the patchDAG
4. Submit the patchDAG to the local Condor DAGMan

**Listing 5.** Receiving algorithm employed within the Adaptation Daemon.

Listing 6 shows the sample patchDAG specification assuming that task B and C needs to be migrated from WFM-1 to WFM-2. Here, a transferData Pre Script before both task B and task C, performs the transfer of any input data required for the execution of these tasks at its new location.

```

Job      B      B.submit
Job      C      C.submit
SCRIPT  PRE  B      transferData.sh
SCRIPT  PRE  C      transferData.sh
SCRIPT  POST B      trigger.sh   B_F
SCRIPT  POST C      trigger.sh   C_F

```

**Listing 6.** Sample PatchDAG specification.

## V. RELATED WORK

Pegasus [6] and ASKALON [11] are two significant workflow management systems for the grid environment. Pegasus uses the Condor DAGMan [12] as the workflow execution engine. A single central DAGMan instance works as a meta-scheduler to submit and monitor jobs on local and remote domains of a grid. ASKALON has its own design of the workflow engines. For each workflow, the execution enactment consists of a master and many slave engine for sub-workflows. In other words, ASKALON supports a distributed model of workflow execution. Unlike Pegasus and ASKALON, the workflow system we propose for a grid environment consists of one or more peer domain workflow managers [5].

Many studies examine the efficiency and effectiveness of run-time workflow rescheduling algorithms. The studies presented in [7, 2] propose modifications to the initial workflow schedule at run-time via some modifications to existing workflow scheduling algorithms. The closest study [8] to the work presented in this paper is provided as an adaptation mechanism on top of Pegasus workflow planner [6]. In this study, the execution of a workflow is monitored periodically, and if there is a substantial increase or decrease in queue wait times per site, Pegasus workflow planner tools are called to perform a new mapping. Then, execution of the current workflow is stopped, and the workflow is deployed again according to the new mapping. Adaptation process in this study is performed in a centralized manner, which is different from our approach. Also, our approach does not necessitate the halting of the execution of the workflow; hence it is expected to be lower-cost and less intrusive under normal conditions.

## VI. CONCLUSION

In this paper, we addressed the scalability and adaptation problems of large-scale workflows in distributed environments. Contrary to many existing solutions, our decentralized approach has a low performance overhead and is transparent.

Our devised patchDAG-based solution does not require the halting and re-deployment of a workflow at runtime, but rather, acts as a bridge between the remapped portions of the workflow. Our prototype implementation on Condor DAGMan shows the feasibility and transparency of our approach. Our future studies include the enhancement of the adaptation with a *pull-based* mechanism. In such a mechanism, a lightly-loaded workflow manager can opportunistically pull some of the tasks from relatively heavy-loaded sites. We also would like to explore the application of our approach to other existing workflow management systems.

## ACKNOWLEDGMENT

This work was supported in part by the NSF (grants OISE-0730065, OCI-0636031, and HRD-0833093), and in part by IBM.

## REFERENCES

- [1] Topcuoglu, H., Hariri, S., and Wu, M. 2002. Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing. *IEEE Trans. Parallel Distrib. Syst.* 13, 3 (Mar. 2002), 260-274.
- [2] Sakellariou, R. and Zhao, H. 2004. A low-cost rescheduling policy for efficient mapping of workflows on grid systems. *Sci. Program.* 12, 4 (Dec. 2004), 253-262.
- [3] *Globus Toolkit*. [Online] <http://www.globus.org/toolkit/>.
- [4] *EGEE*. [Online] <http://www.eu-egce.org/>.
- [5] Kalayci, S., et al., 2010. A Peer-to-Peer Workflow Mapping and Execution Framework for Grid Environments. Technical Report FIU-SCIS-2010-01-01, Florida International University.
- [6] Deelman, E., et al., 2005. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Sci. Program.* 13, 3 (Jul. 2005), 219-237.
- [7] Yu Z, Shi W. "An adaptive rescheduling strategy for grid workflow applications." *IPDPS*, IEEE Press, 2007; 1-8.
- [8] Lee, K., Paton, N. W., Sakellariou, R., Deelman, E., Fernandes, A. A., and Mehta, G. 2008. Adaptive Workflow Processing and Execution in Pegasus. In *Proceedings of the 2008 the 3rd international Conference on Grid and Pervasive Computing - Workshops*. IEEE Computer Society, Washington, DC, 99-106.
- [9] Yu, J and Buyya, R., 2007. Workflow Scheduling Algorithms for Grid Computing. Grid Computing and Distributed Systems Laboratory, The University of Melbourne. s.l. : GRIDS-TR-2007-10.
- [10] Allcock, B., et al., 2001. Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing. In *Proceedings of the Eighteenth IEEE Symposium on Mass Storage Systems and Technologies*. MSS. IEEE Computer Society, Washington, DC, 13.
- [11] Wiczorek, M., Prodan, R., and Fahringer, T. 2005. Scheduling of scientific workflows in the ASKALON grid environment. *SIGMOD Rec.* 34, 3 (Sep. 2005), 56-62.
- [12] Condor team, "The directed acyclic graph manager", [www.cs.wisc.edu/condor/dagman/](http://www.cs.wisc.edu/condor/dagman/), 2002.

# Towards Automated Synthesis of Executable Eclipse Tutorials

Nuyun ZHANG, Gang HUANG\*, Ying ZHANG, Ning JIANG, Hong MEI

*Key Laboratory of High Confidence Software Technologies (Ministry of Education), School of Electronics Engineering and Computer Science, Peking University, Beijing, 100871, China*

\*corresponding to huanggang@sei.pku.edu.cn

## Abstract

Eclipse tutorials guide users step by step to perform programming tasks. However, as current executable tutorials can only guide users through one way, users' specific requirements cannot be satisfied. In this paper, we propose an automated approach to synthesizing different tutorials to generate a tutorial that covers many ways. We use record and replay techniques to generate the original tutorials and synthesize them by mining experts' comments and actions. First, we find the same sub-tasks of different tutorials according to their titles and synthesize the actions of a sub-task by building a hidden Markov model. Then with the synthesized tutorial, steps of a task are recommended gradually during the programming process and executed automatically by replaying some actions. In addition to presenting this approach, we finally present a tool to implement the approach and evaluate its feasibility by an experiment.

**Keywords:** Eclipse, tutorial, user action.

## 1. Introduction

Eclipse based Java programming is complex. As a result there are many tutorials to help users. A tutorial is a step by step guide to perform a task [3]. It is usually written in natural language for users easy to learn. However, users may find it inconvenient to use the text tutorials. Therefore, there are some executable tutorials that are able to automatically execute some steps of a task, such as Cheat Sheet [12], JTutor[4] and SmartTutor[1]. A Cheat Sheet is a text based tutorial with some buttons added in the text. The button can trigger the execution of some work described in the text. JTutor and SmartTutor create a tutorial by recording and replaying experts' programming actions, with critical information highlighted and commented. Since executable Eclipse tutorials are vivid and efficient, they become popular recently.

A tutorial always has a focused mentoring-goal. People can be taught about how to achieve the goal in

various ways. However, due to the reason that executable tutorials are difficult to create, an author often present only one way of achieving that goal in the corresponding tutorial. This practice may leave the novices totally unknown to the other processing ways, other than the given and fixed way in the tutorial. The other ways may be more suitable for his/her programming context. We argue that let novices know the various ways/or the best-suitable way of achieving the goal is important. Therefore, we propose an executable tutorial with more than one way of performing a task. However, knowing the requirements of different ways is challenging. Fortunately, there are many experts who perform a task in different ways in programming practice. They make different tutorials for a task. We can summarize them to create the tutorial.

In this paper, we propose an approach to automatically synthesizing executable tutorials. Usually, each task has some sub-tasks. The ways of performing a task are different from the ways of performing the sub-tasks. First, we find out the same sub-tasks of each tutorial by matching key word of their titles. Then, we compare the performing ways of the same sub-tasks and try to find a complete set of performing ways. We do this by building a math model that takes each step sequence as input and a complete step sequence as output. We build the model by an algorithm proposed by Jonathan [15] to discover process from event data.

After observing programmers' programming processes, we find out that programmers' specific requirements are exposed gradually along with the development of the programming process. Our tutorial recommends the steps with the development of the programming process.

We implement our approach as an extension of our previous work SmartTutor. It works as follows: at the beginning of a task, the system shows users the sub-tasks of performing the task which is an overview, and with the development of task performing, the system recommends the detailed steps on users' demand. The input of the system is a user's finished

actions and her/his requirements about the next step. The output is a set of advice on the next steps for users to choose along with the actions that compose the step.

## 2. An illustrative example

To illustrate our approach, we take the “Create a Hello World application” cheat sheet of Eclipse 3.4 in Figure 1 as an example. In this cheat sheet, there are 5 steps (sub-tasks): 1) “open the Java perspective”, 2) “create a Java project”, 3) “create your hello world class”, 4) “add a print statement”, 5) “run your Java application”. Bob, a novice, opens the cheat sheet to create an application with existing source. He cannot achieve his goal with this only cheat sheet in Eclipse since step 2 does not contain how to include existing sources.

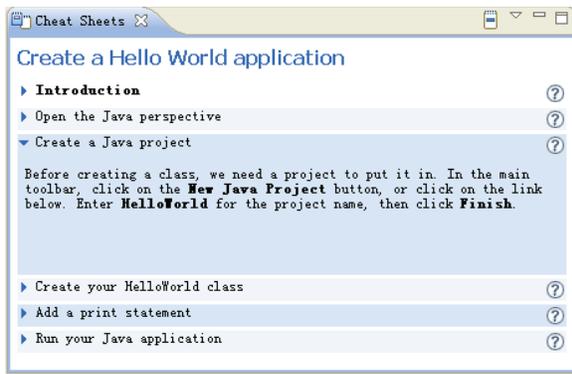


Figure 1. “Create a Hello World application” cheat sheet of Eclipse 3.4

Our tutorial can help him in the following ways. First, a tutorial is generated as follows. In *original tutorial* generation process, we ask experts to perform the task and comment on their actions. The comments include three parts as shown in Figure 2(a). The first one is “Stop line” that divides actions into steps (sub-tasks). The second one is “Description”, which explains each step. The third one is “Title” to entitle each step. The “preview” button is for generating the cheat-sheet-like tutorial. The generated cheat-sheet-like tutorial is shown in Figure 2 (b). The architecture of steps (sub-tasks) can be viewed in tutorial editor in Figure 2(a). The detailed actions are shown in action viewer.

Second, Bob’s task can be guided as follows.

- a) Bob selects the tutorial “Create a Hello World application”. The tool shows him the tutorial in Figure 2(b).
- b) He clicks to begin and the tool opens eclipse “New Project Wizard” in the first step as shown in Figure 2(c). This current step is high-lighted in tutorial editor and step viewer in Figure 2(a). He switches

to our tutorial shown in Figure 2(b) to see the instruction to fill in the project name. He fills in a project name in a pop-up window controlled by our tool as shown in Figure 2(d).

- c) He clicks “finish” to notify our tool that he has finished the name. Then it pops out a menu for him to select the next step, as shown in Figure 2(e). There are 6 options: “Create project form existing source”, “Use a project specific JRE”, “Add Library”, “Create a new class”, “Go to the default next step”, and “I will do it myself”. Bob chooses “Create project from existing source” as he wishes and clicks “finish”. The tool automatically clicks “Create project from existing source” item in “New Project Wizard” window, and clicks “Browse” button to wait for him to choose source files from the file system, as shown in Figure 2(f). The corresponding step is always high-lighted in tutorial editor and step viewer in Figure 2(a).
- d) He clicks “finish”. The tool pops out a window to recommend the next steps as shown Figure 2(g). Bob chooses “Add Library” this time. The tool clicks “Next” button of “New Project Wizard” window to go to “Java setting” page and clicks “Add Library” button of “Libraries” tab automatically, only letting him to add his favorite class files.
- e) Finally, Bob creates his application within Eclipse. It takes him less than 20 minutes.

## 3. Approach overview

The whole approach works as shown in Figure 3. We capture experts’ actions for a task on Eclipse and their comments which describe the meaning of the actions. After a period of time, we can get many different action sequences from different experts for a same task. We mine the sequences to generate a more complete tutorial that covers many users’ requirements of a task. The generation of such tutorial includes not only steps but also descriptions of each step. Then, we can guide a user through the task. By monitoring user actions, the system recommends the subsequent steps with the development of programming process. According to the user’s choice, the system automates the steps by replaying the actions.

The work is based on our previous work SmartTutor, which generates tutorials by recording and replaying an expert’s actions with comments. The record and replay console is called SmartReplayer. By our approach, SmartTutor adds tutorial synthesis and action recommendation function to it.

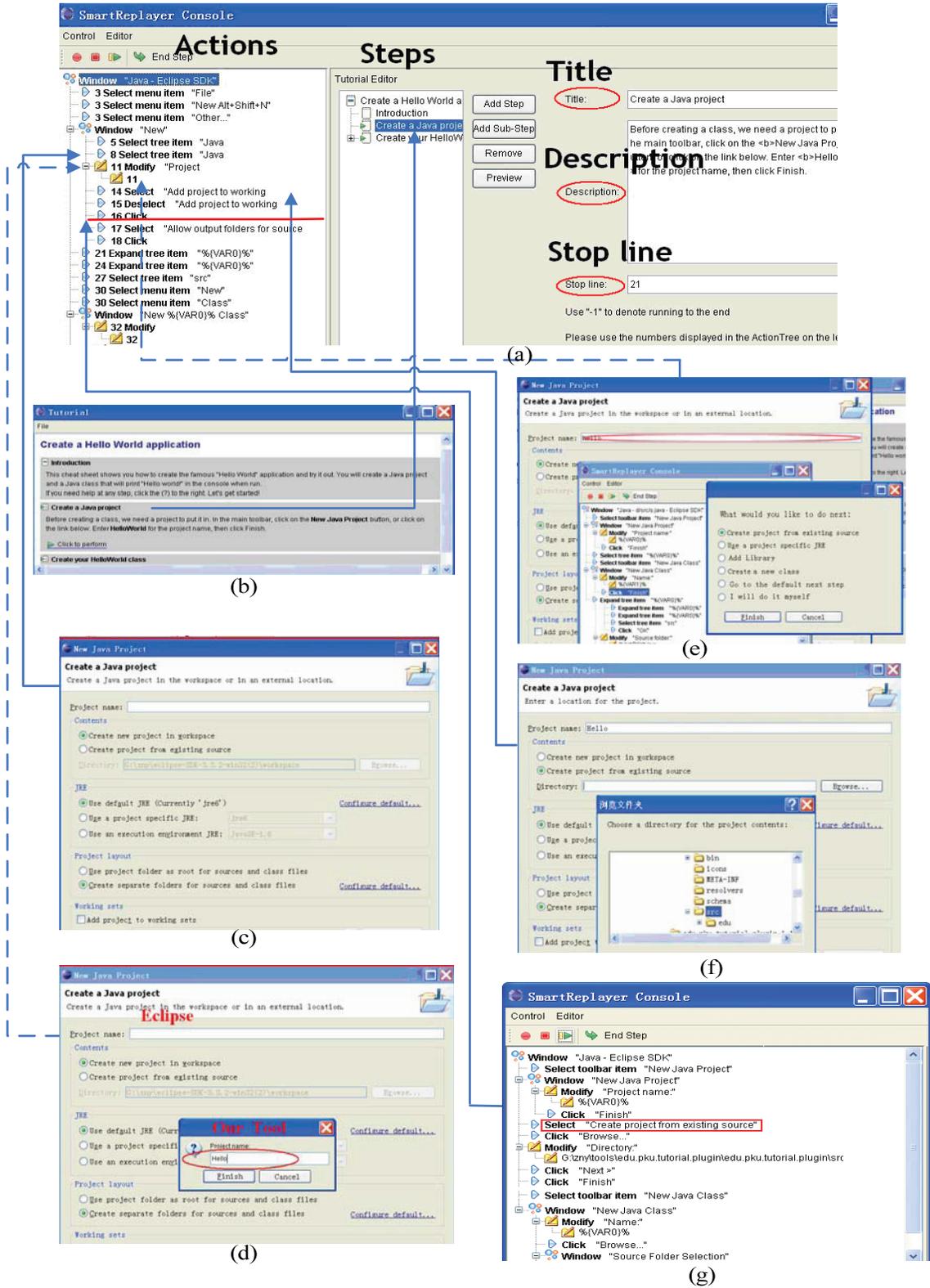


Figure 2. An illustrate example of creating a hello world Java application

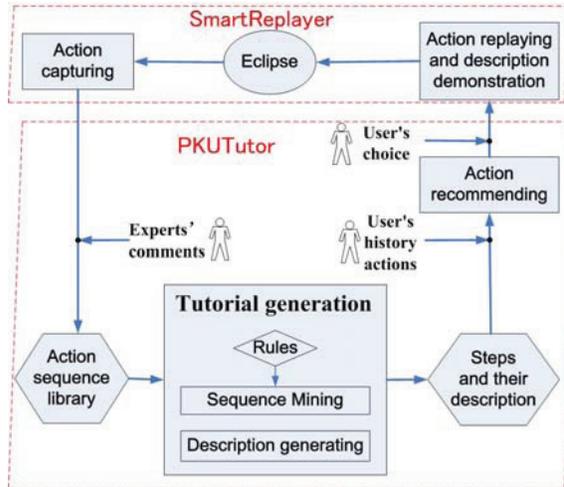


Figure 3. Approach overview

#### 4. Action capture and replay

First, we describe SmartTutor and action capture/replay as the foundation of our approach. A task is executed as a user performs a sequence of actions on Eclipse. A user action is an operation a user performs via input devices such as the mouse and keyboard to operate Eclipse. We capture the low level events of Eclipse GUI system by adding a listener at the base class of all GUI objects, and turn the events into high level actions.

If an action will be used for replaying, three kinds of information should be recorded: action type, action target and action content. Action type means the operation type of using the input devices such as right click, left click, double click, text input, etc. Action target is the GUI object that the action performs on, such as a node in a project tree and a button in a ‘New project’ window, etc. Action content is useful when the target of an action is of some kinds. For example, to a check box, the contents are ‘selected or not’, and to a text box, the contents are the texts in it. In Figure 4, a user inputs some letters in a text field. The action type is ‘text input’; the action target is ‘Project name text field and the action content is ‘Hello’.

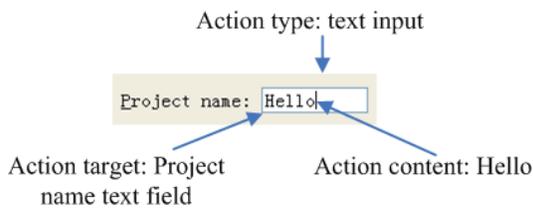


Figure 4. An action recorded in a replayable form

When replaying, we first transform a user action back to event, then dispatch the event to Eclipse which will perform the corresponding operation after receiving an event. The implementation details can be found in work [2] which includes some algorithms for UI matching and action editing.

### 5. Tutorial generation

#### 5.1 Tutorial definition

First, we give a definition of tutorial. By observing the existing tutorials, we define a tutorial containing sequential steps as a line, as shown in Figure 5. A node represents a step (i.e. operations to finish a sub-task) with its description and a directed edge represents a transition between two steps with possible users’ input about the parameters. This kind of tutorial has only one possible next step. Since it can be generated by direct capturing and replaying, we call it *original tutorial* (OT) in this paper.

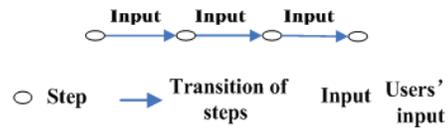


Figure 5. An original tutorial

When multiple next steps are available, it is the exact tutorial that can adapt to more applications. Since this kind of tutorial can be generated by synergizing OTs, it is called synergy tutorial. We call synergy tutorials as tutorial for short later in this paper. We define a tutorial as a directed graph  $G(V,E)$  as shown in Figure 6. Each node  $V_i$  represents a step of the task. Each edge  $E_{ij}$  between two nodes  $V_i$  and  $V_j$  represents the transition between the two steps. A user makes his/her decision by his/her input  $A_i$ .

The execution process of a tutorial is as follows. (1) At each step, there are one or more next steps. The transition ways between two steps can be multiple either. The tool recommends the possible next steps to the user according to his/her current step and the past actions. (2) On receiving some user input, the tool automatically executes and moves to some next step. It recommends and waits again.

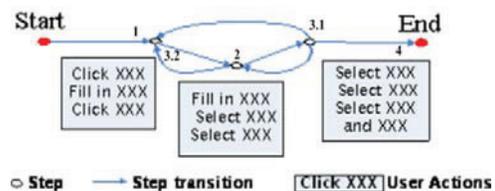


Figure 6. A synergy tutorial defined as a directed graph

## 5.2 What is tutorial synthesizing

The tutorial generation problem is to mine many OTs of a same task to get the common steps to generate the graph that covers all the OTs. It can be described in Figure 7. We call it sequence alignment to find out the common steps. For example, in Figure 7 the first few actions in Alice’s sequence are: Click ‘File’ Menu → Click ‘New’ submenu → Click ‘Java project’ menu item, while in bob’s are: Right click on blank package explore → Click ‘New’ menu item → Focus on project → Click Project → Click Java project. Though these action sequences are different, both of them are for opening a “New Java Project Wizard” and can be aligned into one step. We assume it as the first step in Figure 7. For the third step, Alice performs some actions to add source for the project while bob configures the library. Since the works they do are different, their actions should be aligned onto different nodes. For the last step, they perform a same set of actions and their actions can be aligned onto the same last node.

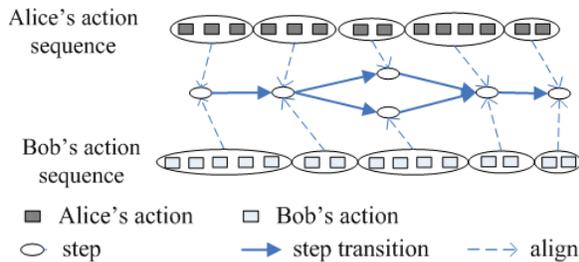


Figure 7. Aligning different sequences onto a graph

## 5.3. Alignment approach

We have surveyed all the tutorials in Eclipse 3.3.1 Help system. The 76 tutorials have 276 sub-tasks in total. Their titles are almost (270 of 276) composed of a verb phrase and a noun phrase. Only a few (6) titles have only a noun phrase. The most used 18 verbs and their frequencies are shown in Table 1.

Table 1. The most used verbs of sub-task titles of Eclipse tutorials

None	project	file	plug-in	editor	code	class
Frequency	23	15	13	12	11	11
None	elements	application	expression	programs	environment	breakpoint
Frequency	11	6	6	6	6	6
None	search	model	memory	manifest	category	feature
Frequency	4	4	4	3	3	3

By observation, the frequencies of these verbs add up to 228, which is 83% of all verbs. The verb phrases are usually a simple word and they repeat very frequently. In addition, according to research in [6],

tags are almost the same. Title is a short description like a tag. The titles are almost the same for the same step. Then, we can align the sub-tasks by their title. We propose an approach to recommending to experts the titles that have been used for similar actions. We use the dynamic time warping algorithm [5] to judge the sequence similarity.

The noun phrases are more complex. There are 1035 words in the noun parts of 276 titles. The most used nouns are listed in Table 2. We find the nouns are more diverse than verbs. The frequencies of the most used nouns add up to 147, which is 14% of all nouns.

Table 2. The most used nouns in steps' titles of Eclipse tutorials

Verb	Create	Open	Viewing	Edit	Define	Run
Frequency	41	20	20	16	16	16
Verb	Add	Use	Configure	Set	Test	Layout
Frequency	15	13	9	9	8	6
Verb	Search	Step	Import	Debug	Evaluate	Move
Frequency	6	6	6	5	5	4

According to the above analysis, we give a word table of verbs from the survey result that has 74 words, a table of noun that has 698 words. The title should contain the two parts and the verb part is allowed to be neglected.

The sub-tasks may have dependency on each other. Their order is usually fixed. As a result, we can align the sub-tasks simply by title and do not have to reorganize their sequence. By observing the 76 tutorials, we find that all experts perform the sub-tasks of a same task in the same order, which proves the reasonability of our alignment approach.

After the alignment of sub-tasks, we compose them together on to one map. We adopt the composition algorithm proposed by Jonathan [15]. The Jonathan algorithm discovers process from event data. It takes the event data as input and output a graph as the process containing all the data. Since combining the event into a graph may produce more paths than actual process, the algorithm use graph transition to avoid redundant paths. It splits a step node into two or more nodes. Then it converts the graph into its dual graph where two or more nodes of a step are integrated into a single node again. The sub-tasks in our approach are considered as the events in Jonathan algorithm. Then we combine them to generate a graph of all the sub-tasks with the Jonathan algorithm.

## 5.4 Description generation

For a sub-step, there may be several titles and descriptions. We choose the title that is used most as the title for the sub-step and the longest corresponding description as the description for the sub-step.

### 5.5 Path validation and action recommending

Since the synergy algorithm may bring some new execution path of a task, its validity should be checked. We do this by replaying them all. If the system throws exceptions, the current execution path is wrong and we eliminate it from the tutorial.

At the end of each step, if there is more than one next step available, the system pops out a window and recommends the steps by listing their titles. Some user may not know what to do at a branch point or have preferences in task performing paths. We offer an automatic approach to selecting path. For users who do not know what to do next, we recommend the default parameters and path which is the shortest path for finishing a task. For users who have special requirements, we can mine the path that meets their requirements.

The path selecting depends on a set of user-specific criterions. For example, if users prefer automated generation of the most complete code, productive paths are selected. If they prefer succinct tutorial, the shortest paths in the graph is mined and selected. If their preferences are an integration of several goals, the multi-objective programming is able to solve it.

Consider the example in Figure 9. There are 5 sequences for creating a Java Project. The dotted one has the least steps. The software artifacts they bring are the same. So, the dotted one can be customized as the most succinct tutorial. However, some people prefer the most used one. Then, the fourth (top-down) sequence in Figure 9 will be chosen. The reason is that according to our statistic, 6 people in 11 choose this way to create a project.

### 6. Case Study

In this section, we show how to create a Java HelloWorld application by our approach. The 11 volunteers as experts are Java programmers who use Eclipse every day in the past two years from computer science department of Peking University. We asked them to build the tutorial by SmartTutor. Surprisingly, none of them made the same tutorial. Parts of the sequences are shown in Table 4. For the convenience of representation, the actions are denoted by numbers. The mapping of number and actual actions can be found in <http://code.google.com/p/smarttutor/>.

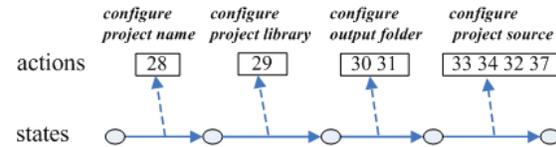
The 11 experts name the sub-tasks all the same according to our word table of the sub-task titles. They perform the sub-tasks in an exactly same sequence that can be simply stringed together. The task is divided into 9 sub-tasks at most: 1) open the “New Java Project Wizard”, 2) create a new project, 3) configure the

project, 4) open the “New Package Wizard”, 5) create a new package, 6) open the “New Java Class Wizard”, 7) create a new class, 8) configure the class, 9) write code. The 4<sup>th</sup> and 5<sup>th</sup> sub-tasks appear in sequence No.10. The 3<sup>rd</sup> sub-task appears in sequences No.4, 5, 6, 9, 10, and 11. The rest sub-tasks appear in all sequences.

**Table 4.** Action sequences to create a Java HelloWorld application

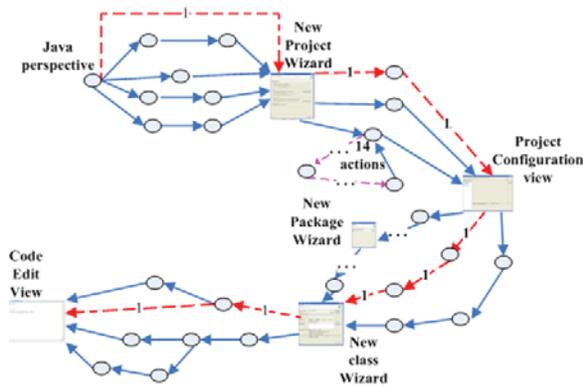
No.	Action Sequences
1	1 2 3 4 5 --6 7-- 8 1 2 9-- 10 18 12- 22
2	13-- 6 7-- 14 15 16 17-- 18 19 10 12- 22
3	1 2 23 --6 7 --14 15 16 1 2 9--10 12- 22
4	1 2 23-- 6 24 7 --14 15 16 1 2 9-- 18 10 12- 22
5	1 2 23-- 6 24 7 --14 15 16 1 2 9-- 10 18 12- 22
6	1 2 3 4 5-- 6 24 7-- 14 15 16 1 2 9-- 10 12- 22
7	1 2 23-- 6 7 --14 15 16 1 2 9-- 10 18 19 12- 22
8	25 26 27-- 6 7 --14 15 16 1 2 9-- 10 18 12- 22
9	25 26 27-- 6 24 7 --16 1 2 9-- 10 12- 22
10	1 2 23--6 24 28 29 30 31 32 29 31 33 34 32 7 --14 15 16 1 2 35 36 37 --14 15 16 381 2 9-- 10 18 19 11 12- 22
11	25 26 27 --6 24 7 --14 15 16 1 2 9 10 18 12- 22

Each sub-task can be aligned. We take the “3) configure the project” sub-task as an example to illustrate the algorithm. The sub-task is divided into 5 sub-tasks by the 11 experts. The 10<sup>th</sup> expert gives the most complete partition: 1) configure project name, 2) configure project library, 3) configure output folder, 4) configure project source. Using the *Jonathan algorithm*, the generated graph of this sub-task is shown in Figure 8. The whole tutorial graph is shown in Figure 9.



**Figure 8.** “Configure the project” sub-task generated by our approach

We assume that some programmers would like a tutorial with the least actions. Among the paths, we use Dijkstra algorithm to find the shortest path which is numbered as “1” in the graph. It has only 8 actions. For a step following which there will be more than one ways to perform, we generate a selection window as shown in Figure 2(e) in the illustrative example section. The window recommends the next steps. In Figure 9, we can see that following “New Project Wizard” there are 3 steps. As a result, there are some options based on these steps shown in Figure 2(e). The complete generated tutorial is shown in Figure 2 in the illustrative example section.



**Figure 9.** The graph of “create a Java hello world application” tutorial

## 7. Discussions

There are some conflicts in task automation and programming skill teaching. For full automation, we should offer users as little information as possible. For better teaching, we should offer users as much information as possible. What we do to compromise them is to customize the execution and exhibition of tutorials according to users’ requirement.

First, we discuss the limitations of our approach.

1. It depends on many various OTs made by experts. If there are few OTs, our approach cannot generate a complete tutorial. On the other side, the more programmers’ using our approach to perform a task, the better our tutorial will be.

2. Though we can build user-specific tutorials, too much user participation may cause some other problems, such as privacy, which is an open issue.

3. We ask the experts to use our word table to entitle the sub-tasks. It may not as convenient as the unlimited natural language. What is more, we conclude the word table from the current Eclipse Help system which may not be complete enough. The word matching problem is beyond the scope of this paper.

Second, we discuss the efficiency of our approach. It is mentioned in the Eclipse Cheat sheet developers’ guide [7] that “In a tutorial the goal is to learn how to perform a specific task. Cheat sheets will usually contain up to 10 steps and can be completed in a half an hour or less. For larger tasks consider using a composite cheat sheet.” A sub-task usually has less than ten steps so that our synergy can be efficient enough.

## 8. Related work

The first area of related works is on software example recommendation systems [11][13][14]. It mines all kinds of software repositories to learn from users’ usage history and give advice to them. The purpose and method of this field is mainly for API learning.

The second area of related work is on action capture and replay. Related works are numerous including SCRAPE [8], Eclipse TPTP [9]. But we replay the actions in a different way, which offers more paths than the traditional replay technique.

The third area of related work is on executable tutorial generation. As mentioned in the introduction section, Cheat Sheet uses hard coding while JTutor and SmartTutor use action record and replay. JTutor is mainly used for demonstration like a video while SmartTutor allows users to modify the recorded steps to adapt to some little change in the programming task. However, these approaches cannot automatically generate tutorials with many paths.

The fourth area of related work is on task automation, including Sheepdog [10]. It extracts a procedure from experts’ action sequences for technical support such as network configuration. The task is highly automated without human invention. Sheepdog adopts the IOHMM algorithm to train the whole process while we adopt the graph transition algorithm to train the steps of a sub-task. We have tried to use HMM training approach. It turns out that the generated models are difficult for human to understand and the results of sub-task alignment are especially terrible. But the Jonathan algorithm performs well.

## 9. Conclusion

Existing Eclipse tutorials usually provide one way of performing a task that cannot meet users’ specific requirements, mainly because the generation of a synthesized tutorial is very challenging. In this paper, we propose an automated approach for tutorial synthesizing so that Eclipse tutorials can guide users in many ways. We generate the OTs by action capture and replay. We synthesize them by title matching and action alignment. We can mine the many paths to find a path that has some special quality or suits users’ special requirements.

The contributions of our work are mainly two folds. First, we propose a definition for tutorial and divide them into two categories by their ways of performing a task. Second, we propose an approach to automatically synthesizing OTs, which frees human from building a complete tutorial.

## 10. Acknowledgement

This work is supported by the National Basic Research Program of China (973) under Grant No. 2009CB320703; the National Natural Science Foundation of China under Grant No. 60821003, 60933003; the National S&T Major Project under Grant No. 2009ZX01043-002-002; and the New Century Excellent Talent Program of MOE.

## 11. References

- [1] Y. Zhang, G. Huang, N. Zhang, H. Mei. "SmartTutor: Creating IDE-based interactive tutorials via editable replay". ICSE'09, pp 559-562.
- [2] Y. Zhang, G. Huang, N. Zhang, H. Mei. "Editable Replay of IDE-based Repetitive Tasks". COMPSAC'08, pp559-562.
- [3] D. C. Dryer. "Wizards, Guides, and Beyond: Rational and Empirical Methods for Selecting Optimal Intelligent User Interface Agents". IUI'97, pp265-268.
- [4] C. Kojouharov, A. Solodovnik, G. Naumovich. "JTutor: an Eclipse plug-in suite for creation and replay of code-based tutorials". OOPSLA'04 ETX, pp24-28.
- [5] L. Rabiner, A. Rosenberg, S. Levinson. "Considerations in Dynamic Time Warping Algorithms for Discrete Word Recognition". IEEE Trans. on ASSP. Vol 26, Issue 6, pp575-582,1978.
- [6] V. Loreto, L. Pietronero, "Collaborative Tagging and Semiotic Dynamics", CoRR abs/cs/0605015: 2006, arxiv.org.
- [7] Eclipse org, "Cheat sheet authoring guidelines". Platform Plug-in Developer Guide > Programmer's Guide > User assistance support > Cheat sheets in Eclipse 3.3.1. 2008
- [8] A. Orso, B. Kennedy. "Selective Capture and Replay of Program Executions". SIGSOFT Softw. Eng. Notes 30, 4 (Jul. 2005), pp1-7.
- [9] Eclipse org, TPTP, <http://www.eclipse.org/tptp/>.2006
- [10] T. Lau, L. Bergman, V. Castelli, D. Oblinger. "Sheepdog: Learning Procedures for Technical Support". IUI'04, pp109-116.
- [11] S. Thummalapenta, T. Xie. "PARSEWeb: A Programmer Assistant for Reusing Open Source Code on the Web". ASE'07, pp204-213.
- [12] "Building cheat sheets in Eclipse V3.2." <http://www-128.ibm.com/developerworks/library/os-ecl-cheatsheets/>
- [13] A. Michail. "Data mining library reuse patterns using generalized association rules". ICSE'00, pp167-176.
- [14] Y. Ye and G. Fischer. "Supporting reuse by delivering task-relevant and personalized information". ICSE'02, pp513-523.
- [15] Jonathan. E. Cook, A. L. Wolf. "Automating Process Discovery through Event-Data Analysis". ICSE'95, pp373-38.

# AN EXAMINATION OF A RULE-BASED EXPERT SYSTEM TO AID IN THE IMPLEMENTATION OF THE CMMI FRAMEWORK

Tessa Adderley, Sheryl Duggins, and Frank Tsui  
Southern Polytechnic State University

**Abstract** -- Implementing the Capability Maturity Model Integration (CMMI) framework in an organization can prove costly and daunting to an organization seeking to implement this framework for the first time. Complaints from organizations attempting this rigorous improvement program address the large overhead, implementation risks and lack of quantitative value compared to their present processes. However, using a simple rule-based expert system may address the implementation difficulties by suggesting a path of improvement incorporating the framework. This paper will examine the effects of applying an expert system as a process improvement management tool in the implementation of the CMMI model.

## 1. INTRODUCTION

The cost associated with implementing the Capability Maturity Model Integration (or CMMI) framework in an organization is significant, and without proper guidance and training, can prove to be daunting to an organization with process improvement goals [1]. Costs are associated with a number of activities including proper training of those involved in the process improvement effort, pre-assessments to determine the current state or condition of the organization, the actual implementation of the model itself, and appraisals which uncover the organization's strengths and weaknesses after implementation of the framework. Both assessment and appraisals can be very costly, and this cost varies based on the size of the organization and the model level being obtained. For smaller software companies (which make up a large percentage of the industry), the large cost of implementing the CMMI models can prove too much and most companies are either reluctant to use it, or do not understand the value-added. Because the value of having effective processes and best practices within organizations has shown to be important, an alternative and possibly more cost-effective way of implementing the CMMI framework is needed. Additionally, because the model has proven itself as an asset in improving quality and processes within organizations, as well as streamlining businesses, it would be of interest to explore a possible software solution that could be used as a management tool. This tool would assist in model implementation, and

would also help in keeping the organization on the right path to its process improvement goals.

Although expert systems have been widely used in many areas, little work has been done in the possible automation or expert system implementation of the CMMI model, an area that could possibly benefit by the utilization of an expert system. This paper will explore using a simple rule-based expert system to address the CMMI implementation difficulties by suggesting a path of improvement incorporating the framework. By applying an expert system to an existing software engineering process improvement model, this paper examines the use of an expert system as a process improvement management tool in the implementation of the CMMI model.

## 2. WHAT IS CMMI?

A process is defined as a particular course or set of actions that produce a result. The CMMI along with its predecessor CMM is a process improvement framework that seeks to provide guidance in the implementation and continuous improvement of an organization's processes, as well as the management activities associated with the development process of an organization [2].

The goal of the SW-CMM was to help software development companies become more mature such that in terms of projects and processes, predictability will be high, while risk is low. A fully mature organization is characterized by development processes that are disciplined, predictable, and most importantly repeatable [3].

### *A. The Structure of the CMMI Framework*

In using the CMMI model for process improvement, an organization must identify and clearly define their process improvement goals. There are three key elements that need to be chosen: the part of the organization to be used in process improvement, the model, and the representation. The CMMI model has two representations, the staged and continuous models, and although the goal of the models is the same, their approach to process improvement is different. Process area capability is the primary focus of the continuous representation while organizational maturity is the focus of the staged representation. The purpose of both representations is to take processes that are ill-defined or not defined at all, and turn them

into useful, quantifiable and well managed processes enabling the organization to reach their business objectives [4].

After implementation of either representation or successful process improvement, many organizations want to measure their progress and conduct an appraisal. Appraisals can be done for one or more reasons including informing external customers and suppliers of how well the organization measures up to the CMMI best practices, identifying areas where improvements can be made, and meeting contract requirements of one or more customers [4].

### *B. Model Implementation*

The actual implementation and use of the CMMI framework into mainstream businesses has seen positive results in these organizations. While the precise results of CMMI model implementation vary, organizations that have implemented CMMI initiatives have seen results including: reduction in general costs and delivery associated with their products, improved budget estimation accuracy, reduction in variation in schedule, improved software production, and improvement investments [5].

In looking at the software industry today with respect to the CMMI framework, the software products available for use as management tools for the CMMI framework are limited [6,7,8]. Most products are geared toward individual process areas such as but not limited to: Measurement and Analysis, Project Planning, and Organizational Training. There are also a few software tools that cover limited parts of the CMMI models with the use of what is referred to as a Process Asset Library (PAL); these typically focus just on level three of the staged representation.

## **3. THE INTELLIGENT ASSISTANT**

After careful analysis of the CMMI framework structure and its benefits, it is noted that there is a need for a management tool that would assist organizations in using the CMMI framework for their process improvement goals. After examining the different aspects of the CMMI framework and the different approaches that can be used for implementing the different models in an organization, it became clear that there is no simple algorithm that could utilize conventional programming to build a management tool that would serve as a “diagnostic intelligent assistant” for organizations seeking to use the CMMI framework for their process improvement goals. Therefore, we decided to use a simple rule-based expert system to implement the intelligent assistant. But to do this, we first had to decide what our tool needed to do.

In order to successfully “guide” an organization in its process improvement efforts with respect to the CMMI framework, the tool would need to complete several main activities. First the tool would need to effectively collect information about the organization. This organizational data should include: improvement goals, organizational goals and objectives, and any problems or potential problems (risks). This information would then need to be represented as relationships between important elements such as goals and problems, and the specifications of these elements must be detailed and prioritized. This specification when used with the inference rules in the inference engine would allow for association of these goals and/or problems to a particular process area, while also pinpointing the processes that either need to be improved or established within the organization. Further analysis of the process specifications would determine which process areas address the needs of the goals and problems, as well as determine which specific goals and practices within the process areas would address the needs of the process. The resulting output would provide a suggested path for improvement with completed steps outlined, and incomplete steps detailing proven practices that would directly address the improvement effort. Lastly the tool would have to be able to track the improvements that have been made.

The next section discusses the results of a survey indicating the need for such a tool and the implementation of our intelligent assistant.

### *A. Survey Results*

As part of this study, 50 software engineering companies in the US were asked to participate in a telephone survey to gauge the use of the CMMI framework. Since we called them directly, we had 100% compliance; nevertheless, the results were disheartening. Forty-two percent of the respondents thought that the CMMI framework was too big and complicated, while others (14%) complained of the mass overhead associated with implementing such a hefty process improvement program. Another 14% stated that the use of the framework was just too costly and that it was not worth the time it would take to implement it. Eighteen percent of the respondents were agile-based, and believed that the framework was counteractive to their agile processes, while 8% knew nothing of the model at all. Only 4% used some version of CMMI: one company successfully implemented the CMMI framework at level three, and another was currently using the SW-CMM.

After speaking directly with a number of CMMI consultants [9,10], research revealed that the problem that most organizations face with the implementation

of this model despite its high overhead is in the implementation itself. Often times the drive behind implementing the CMMI framework is based on client requests, or organizations focusing more on the level to be achieved than the benefit the improvement program can bring to the organization. Other reasons that organizations fail are that the process improvement program does not directly address the goals or needs of the organization; the program is implemented in its entirety instead of in smaller manageable pieces; and people within the organization are resistant to change. Additionally, others reported a lack of understanding of the model being implemented, and how the model should be implemented to have an effective process improvement program. Ultimately the scope of the improvement program should be driven by the organization's specific needs, and should be defined by the organization's goals and problems.

In order to be considered a successful management tool, the intelligent assistant would need to deduce from the information collected (i.e., organizational goals and problems, risks, funding, and company culture) an appropriate improvement path, or action plan, that would link the organization's problems to the appropriate goals, and map CMMI generic practices to the goals as possible solutions. The tool would also allow for actual process improvement to be tracked enabling the organization to see that the improvement plan is working. This would allow management to not only map the CMMI practices to their own which is what is required of the improvement program, but also would provide proven solutions to existing problems.

### B. Expert System Construction

With the use of an expert system shell, an intelligent assistant that could serve as a guide for those seeking to implement the CMMI framework for process improvement was developed. The expert system shell being utilized was Jess, a rule engine written entirely in Java [11]. The structure of an expert system consists of the knowledge base, the working memory, and the inference engine that controls the system's inner workings. Jess works by "matching" the facts found in the working memory to the rules found in the knowledge base through the use of the inference engine. This inference engine matches through the use of a pattern matcher that decides what rules are on the agenda. This schedules the activated rules that will fire. The "firing" of the rules is done by the execution engine. Jess uses a Rete algorithm for the pattern matching by building a network of nodes, each of which represents one or more tests found on a rule's "IF" side [11].

Summarizing, the engine matches the facts and rules, prioritizes them and then applies those rules based on the facts stored in the working memory. The information stored in the working memory has a representation similar to that of frames. The information is stored as facts, and every fact has a template. The templates provide the name and list of slots to the fact. The slots hold data about the fact, and a template can have one slot or multiple slots.

### C. Implementation

The goal of our intelligent assistant is to provide the user with a suggested path for process improvement that incorporates the proven practices within the CMMI framework. The assistant allows the user to implement improvement in smaller, more manageable pieces. The assistant receives company profile information from the user including the organization's various business goals and objectives, problems keeping the organization from reaching those goals, company culture, project implementation styles and risks, as well as CMMI criteria (model, representation, capability or maturity level, and additional appraisal goals). Our tool's output is a suggested process improvement path that maps the problems to related goals, prioritizes them based on importance, and allows the user to define metrics; it also maps the CMMI process areas (their specific goals and practices) to the problems and their goals, and tracks the progress of each area.

For the sake of simplicity, information gathered from the GUI was done through the use of radio buttons, Yes or No answers, and check boxes. This was done to avoid parsing large volumes of text that could be entered by the user, and also to avoid the errors that could arise with having to match so many different keywords. The templates stored in the working memory hold information on every single element: *framework* holds multiple slot information about the CMMI constellation, model, representation and level; *level* has multiple slot information about process areas; *process areas* hold information about the specific goals; and *company templates* hold information about goals, objectives and additional information.

The GUI for the assistant allows the user to create a new user profile or open an existing profile. The user is then guided through a series of questions specific to the user's organization. As the user completes each section of the diagnostic tool beginning with profile information, the information gathered is asserted in the corresponding fact template. At the end of the diagnostic, the rule engine fires all the rules that have been satisfied and presents the findings. The user is given a suggested improvement

path; shown the groupings of problems to their related goals; presented the appropriate parts of the maturity level process areas that can provide solutions to their problems; and extensions to new windows that will help define metrics and store data for appropriate tracking of improvement. The improvement plan will be based on the user's stated priority of the goals and problems, and pressing issues are suggested first.

In order to know whether or not the prototype would be successful, the prototype was used by the Chief Operating Officer (COO) of Organization X. Based on previous attempts at process improvement, the COO logged his results in a short survey created along with the prototype. The results obtained from the prototype are found in Figures 1 and 2.

Organization X is an agile-based software engineering firm with approximately 30 employees, whose primary business is software solutions. A new contract requires them to be rated at least at level two of the SEI's CMMI framework. Funding for the process improvement effort has been secured, and the appropriate team leaders have been trained by SEI. The company's main goal is to meet all scheduled commitments, with intermediate goals being improvement of software quality and increase in profits. The problems that Organization X faces are that employees feel that documentation takes too much time, and would rather just code the project; deadlines are often missed because of lack of a proper project management plan causing unforeseen project escalation, and because there is no real time allotted for proper design activities, customer requirements are sometimes missed, and the lack of testing tools forces more rework, risking budget excess.

While we are aware that having our prototype beta-tested by only one company is definitely too small of a sample size to claim success, we simply wanted to get some real feedback from a client experienced with the CMMI framework. Based on his answers, he felt that the tool was a helpful starting point for those seeking to implement the CMMI framework. He found that the question and answer format forced him to sit down with management and determine each individual goal and objective, as well as possible problems that prevent the achievement of these goals. This encouraged management and employees alike to find a purpose for the improvement effort and not to just implement the framework because the client requested it. Additionally the COO felt that the proven processes mapped to his existing problems were good starting points and considered them "plausible solutions" to some of the existing problems that the organization was challenged with. On the negative side, he felt that the tool was very limited in addressing the needs of all companies considering that

individual goals and problems had to be chosen from predetermined lists.

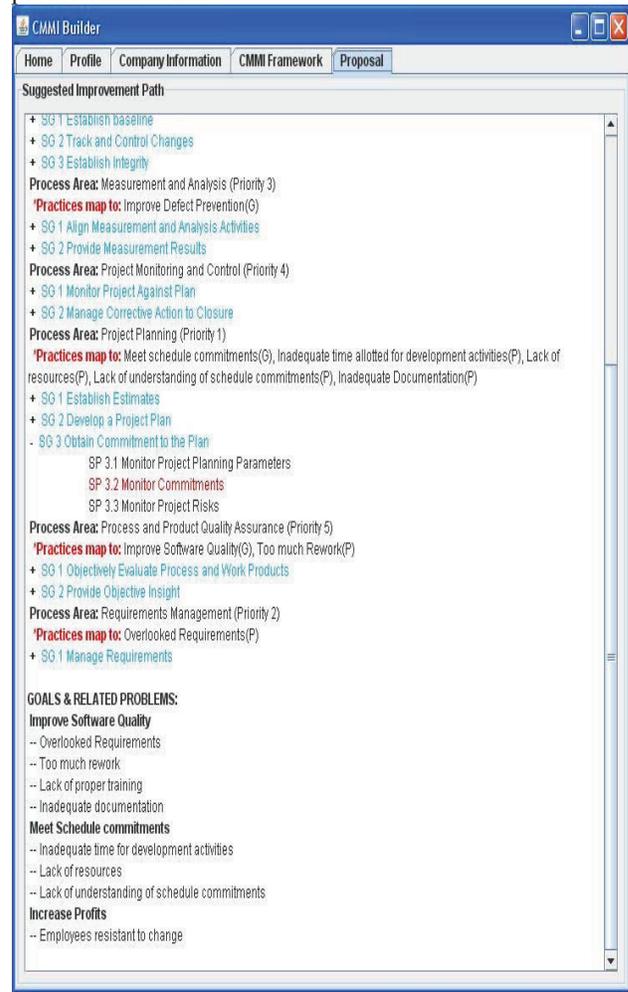


Figure 1: Suggested Improvement Path Screen 1

#### 4. SUMMARY AND RESULTS

This study revealed that issues such as implementation methods play a large role in the success of a process improvement effort. A prototype was built using an expert system shell to help companies trying to implement the CMMI model. The tool provides help in the definition and documentation of a company's process goals and objectives; and also provides some guidance on the path towards a rational approach to attain a specific CMMI level.

Our limited results show that the intelligent assistant proves helpful because during the question and answer methods, the organization is forced to specify and adhere to their set goals and objectives. This makes the process improvement effort reflect the needs of the organization, and employees are more willing to accept a process improvement effort that has a compelling purpose [10]. The results also show that

the CMMI framework can not only be used for improvement of process, but also to identify plausible solutions to existing problems. Because the framework consists of a group of proven processes, knowing that these processes can work to solve a company's problem is half the battle. The other half lies in the implementation and tracking of the process.

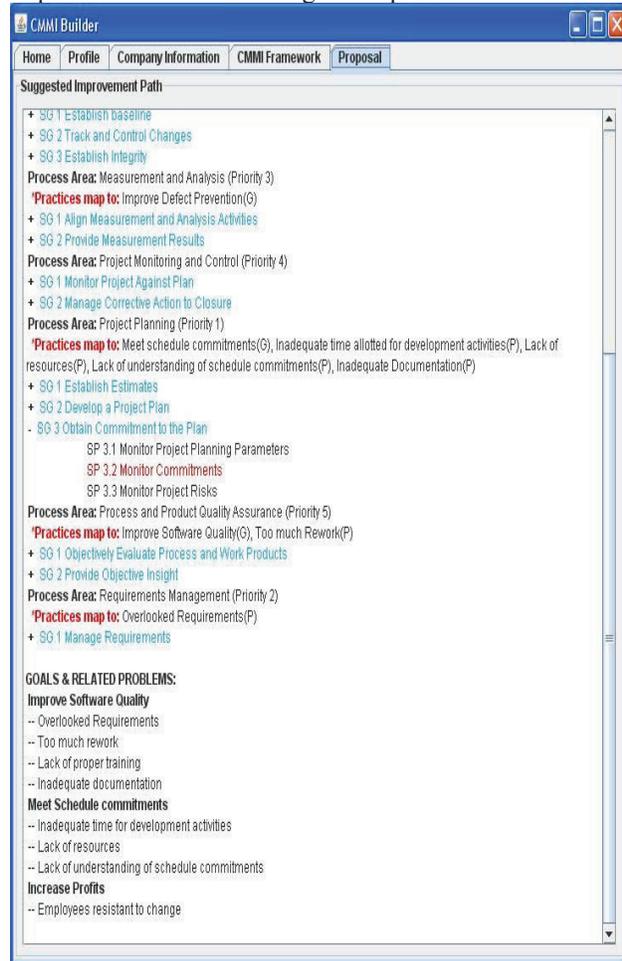


Figure 2: Suggested Improvement Path Screen 2

However, our client found our tool to be limited in its use due to the type of expert system logic chosen and the limited representation technique. We realize that our implementation did not allow for the processing of all the infinite combinations and variations of elements that are involved in a process improvement effort. That would require a much more complex system, one that would allow for better collection and representation of the user information. This future implementation would allow the user to input their own individual goals, objectives, problems and improvement goals, while still allowing them to be weighted for priority, and would result in a more detailed improvement program. This future solution

would have to parse through the user input for keywords that would associate the component to a process area, which would require each component to be well-defined, and well-represented. Thus better collection and representation techniques would allow for better associative relationships between the information collected and appropriate process areas; it would also allow for the tool to pinpoint which process within the organization needs to be improved or even established, and which specific practices within associated process areas would work to improve an existing process.

This study revealed that while the CMMI framework has been beneficial to the software engineering industry, businesses trying to implement the CMMI are faced with massive associated costs. Considering the benefits that the framework can bring to an organization, we are suggesting that perhaps a tool like our intelligent assistant, while weaker in power than the future system proposed in the previous paragraph, may be exactly what the software industry needs to get organizations to more easily adopt CMMI, where the complexity burden currently keeps a lot of prospective software companies away.

## References

- [1] Payne, D. (2009, July 29). *CMMI Implementation and Training* [Telephone interview].
- [2] CMMI Product Team. (2002). *Capability Maturity Model Integration (CMMI) Version 1.1 CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing Staged Representation* (Tech.). Pittsburgh, PA: Carnegie Mellon Software Engineering Institute.
- [3] Chapter 1: Overview of the Capability Maturity Model. (2001). In J. R. Persse, *Implementing the Capability Maturity Model* (pp. 3-11). New York: Wiley. *NetLibrary*. Retrieved July 14, 2008.
- [4] CMMI Product Team. (2006). *CMMI for Development Version 1.2, p. 42-47, 68* (Tech.). Pittsburgh, PA: Carnegie Mellon Software Engineering Institute.
- [5] Gibson, D. L., Goldenson, D. R., & Kost, K. (2006). *Performance Results of CMMI-Based Process Improvement*, p.7 (Tech. No. 004). SEI/CMU. *Software Engineering Institute/Carnegie Mellon*. Retrieved July 17, 2008, from <http://www.sei.cmu.edu/publications/lists.html>
- [6] CS/10,000 and the Capability Maturity Model. Retrieved from <http://www.cscl.com/techsupp/techdocs/cmmwp.html>
- [7] Process Max: The Process Solution. Retrieved from <http://www.pragmasystems.com/products.htm>
- [8] Gutierrez, Celia (2005), "Automation of Diagnosis for the Highest CMM Levels in Small and Medium Sized Software Companies," *WMSCI 2005, Orlando, FL, July 2005*.
- [9] Dalton, J. (2009, September 30). [Telephone interview].
- [10] Potter, N. S. (2009, October 29). [E-mail interview].
- [11] Friedman-Hill, E. J. (n.d.). *Jess, the Rule Engine for the Java Platform*. Retrieved December 3, 2008, from <http://www.jessrules.com/jess/docs/71/index.html>

# TOOLS FOR ONTOLOGY MODELING AND VISUALIZATION

*Simon Suigen Guo, Christine W. Chan, Robert Harrison*

Energy Informatics Laboratory, Faculty of Engineering and Applied Science  
University of Regina  
Regina, Saskatchewan, S4S 0A2, Canada  
Email: guosi111@uregina.ca, chancw@uregina.ca, harrisor@uregina.ca

*Abstract* -- This paper presents the design and implementation of a software suite that combines knowledge modeling and visualization for enriching and enhancing ontology authoring and acquisition. This suite of software tools has been developed based on the Inferential Modeling Technique [1], which is a technique for modeling the static and dynamic knowledge elements of a problem domain. The strength of the software suite is that it enables modeling and visualizing dynamic knowledge. The tools have been applied for constructing and visualizing a knowledge model in the petroleum contamination remediation selection domain.

**Keywords** – Ontology, Ontological Engineering, Knowledge Model, 3D Graphic

## 1. INTRODUCTION

Ontological engineering aims to model concepts, axioms and facts which reflect a real world phenomenon of interest. In the modeling process, ontological engineering software tools are often used for creating and editing application ontologies. However, tools that can simultaneously support an ontological engineering methodology and dynamic knowledge modeling and visualization are lacking. The objective of this paper is to propose a suite of tools for supporting dynamic knowledge modeling and visualization. The tools have been constructed based on the theoretical framework of the Inferential Modeling Technique (IMT) [1], and they support dynamic knowledge modeling and ontology visualization. This tool suite consists of two parts: 1) the dynamic knowledge modeling tool, called Dyna, 2) the ontology visualization tool, called Onto3DViz. Both Dyna and Onto3DViz are applied for constructing and visualizing the application ontology of petroleum contamination remediation selection domain.

This paper is organized as follows. Section 2 presents relevant background literature about the field of ontology modeling and visualization tools. Section 3 describes the development of the proposed ontology modeling and visualization tools. Section 4 presents an application study

of Dyna and Onto3DViz, and section 5 discusses some strengths and weaknesses of the tool suite for visualizing application ontologies. Section 6 provides some concluding thoughts and discusses directions for future work.

## 2. BACKGROUND LITERATURE

Software tools enable users to ignore the complexities of the ontology languages. They support quick application development, so that users can complete the tasks of knowledge modeling and visualization with automated support.

### 2.1 Inferential Modeling Technique

The Inferential Modeling Technique is a knowledge engineering method that supports developing a domain ontology consisting of both static and dynamic knowledge of the problem domain. Static knowledge consists of concepts, attributes, individuals and the relationships among them; dynamic knowledge includes objectives, tasks, and relationships among objectives and tasks. Static and dynamic knowledge are intertwined in that a task is a process that manipulates static knowledge to achieve an objective. The details of this modeling technique can be found in [1].

### 2.2 Ontology Modeling Tools

A primary function of an ontology tool is its ability to model the problem domain. A brief survey of the different ontological engineering support tools reveal there are diverse methods employed. For example, the method used by Protégé [2], Ontolingua [3], and OntoBuilder [4] for modeling ontologies is to use a hierarchical tree listing all the concepts, and input fields are provided to capture characteristics of concepts. A graphical method of modeling ontologies is employed in tools such as KAON [5] and Protégé OWLViz Plug-in [6], in which the representational method uses nodes to represent concepts and edges to represent relationships between concepts.

Existing tools can model static knowledge to varying degrees. However, there are difficulties when using these tools to model dynamic knowledge. For example, it is

awkward to link a task to an objective in Protégé using the existing input fields. This is a weakness that Dyna addresses.

### 2.3 Ontology visualization Tools

There are many ontology visualization tools. Although Protégé has some degree of visualization capabilities in that it generates a tree view of classes; the actual ontology visualization applications are implemented in its plug-ins. Some examples of these ontology visualization plug-ins include Jambalaya [7] and OntoSphere [8], which are described below. Dyna also has some limited graphic capabilities. Jambalaya is a Protégé plug-in that implemented the Simple Hierarchical Multi-Perspective (SHriMP) [9] visualization technique. SHriMP represents a hierarchical structure of information as a set of nested graphs [10].

Jambalaya provides several viewing perspectives for the ontology model, thereby enhancing user browsing, exploring and interacting with two dimensional or 2D ontology visualization. OntoSphere has been implemented as a Protégé or Eclipse [11] plug in. By employing three dimensional or 3D graphics for visualization, OntoSphere extends the volume of space available for visualizing overcrowded concepts. A main advantage of a 3D representation is that it allows users to manipulate the visualized knowledge elements of the application ontology by means of the actions of zooming, rotating and translating.

## 3. DESIGN AND IMPLEMENTATION OF DYNA AND ONTO3DVIZ

Dyna has been developed based on the IMT, and its objective is to support modeling of dynamic knowledge. Dyna has been developed as a Protégé plug-in. Dyna is a “Tab Widget” that works with both Protégé-Frames and Protégé-OWL. At the time of writing, Dyna has been tested on Protégé 3.2.1.

The user creates the static knowledge model consisting of classes, properties, and relations in Protégé, and the dynamic knowledge model consisting of objectives and tasks in Dyna. Two modules in Dyna are responsible for creating objectives and tasks. From the Objective module, tasks can be linked to objectives. The Task module supports the definition of task behaviour and the instantiation and manipulation of objects and properties created in Protégé. The Task module also supports the creation and running of test cases for verifying that the task behaviour is working as expected. Both Protégé and Dyna can import and export knowledge models specified in the OWL<sup>1</sup> file format, and Dyna can also import and export knowledge models in the XML<sup>2</sup> file format. While Dyna can support ontology modeling, its capability for supporting

ontology visualization is limited. Onto3DViz addresses this limitation.

Onto3DViz has been developed for visualizing an application ontology in 3D graphics. It is written in Java™ language and its 3D visualization engine is implemented in Java 3D™. The main difference between Onto3DViz and other ontology visualization tools is that Onto3DViz is a visualization tool developed based on the IMT [1], and it supports visualization of both static and dynamic knowledge models specified according to the IMT. As well, Onto3DViz supports knowledge sharing and re-use, by requiring ontology documents represented in OWL as the input.

The design of Onto3DViz consists of the three major components of (1) Graphical User Interfaces (GUI), (2) ontology document processor and (3) 3D graphic rendering engine. Each of the three components contains sub packages. The GUI package is used to create windows for displaying the output model generated by Onto3DViz, it also displays text and responses to user inputs. The Ontology Document Processor processes the input ontology documents, such as OWL or XML files, and extracts the knowledge model from the input document, which is then saved in the computer memory. The 3D Graphic Rendering Engine is for generating the visual objects that correspond to the knowledge types specified in the model, and it also creates other visual effects, such as lighting, color and texture.

## 4. APPLICATION OF DYNA TO MODELING OF PETROLEUM CONTAMINATION REMEDIATION SELECTION DOMAIN

### 4.1 Static Knowledge Modeling of Dyna

The static knowledge model of the application ontology for the domain of petroleum contamination remediation selection was constructed in Protégé-OWL 3.2.1 based on the ontology presented in Chen [12]. The static knowledge was modeled using an iterative process of creating a class and then its properties and other characteristics. Since the class of *media* is an important knowledge element in the domain of petroleum contamination remediation selection, this element is used in the following example for illustrating the process of creating a class and its attributes or characteristics:

1. Create a class: *Media*
2. Create a property for the class. OWL supports two types of properties: datatype and object.
  - 2.1 Specify the domain(s) of the property. For *siteSize*, the domain was set to *Media*.
  - 2.2 Specify the range of the property.
  - 2.3 Specify restrictions for the property.
    - 2.3.1 Restrict the possible allowed values.
    - 2.3.2 Specify whether or not the property is “functional”.

<sup>1</sup> OWL Web Ontology Language, <http://www.w3.org/TR/owl-features/>

<sup>2</sup> Extensible Markup Language, <http://www.w3.org/XML/>

- Repeat steps 1 and 2 until either all the classes have been represented or the knowledge engineer has enough information to develop the dynamic knowledge model.
- Specify disjointness. Figure 1 shows the class hierarchy of the static knowledge model.

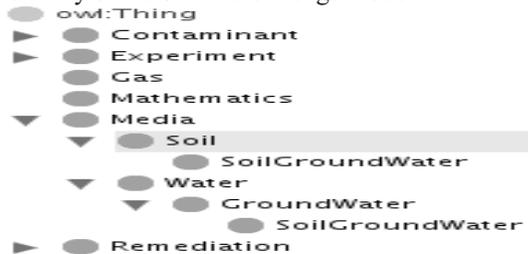


Figure 1. Class Hierarchy in Protégé

#### 4.2 Dynamic Knowledge Modeling of Dyna

The dynamic knowledge component of the domain ontology was constructed in Dyna based on the same ontology presented in Chen [12]. The dynamic knowledge was modeled in Dyna and involved an iterative process of creating an objective, and then its tasks. A sample element of dynamic knowledge is used in the following example to illustrate the process of creating objectives and tasks:

- Create an objective: *DetermineSiteSize*
- Create a task: *MeasureContaminatedSiteVolume*
  - Specify task behaviour and any inputs and outputs. The behaviour for *MeasureContaminatedSiteVolume*.
  - Specify any objects that are used in the behaviour. *MeasureContaminatedSiteVolume* uses the object site, which is an individual of *Media*.
  - Specify test cases..
- Link the task to the objective.
- Adjust the priority of the task in the objective (if necessary), as shown in Figure 2.
- Repeat Steps 1-4 until the knowledge engineer is satisfied that the model is complete.

Task	Priority
MeasureContaminatedSiteArea	1
MeasureContaminatedSiteVolume	2

Figure 2. Dyna – DetermineSiteSize Objective

#### 4.3 Knowledge Visualization of Onto3DViz

Onto3DViz was applied for visualizing the application ontology of the domain of selection of petroleum contamination remediation technology. The knowledge modeling process was originally conducted based on the IMT, and the knowledge model consists of both static and dynamic knowledge. Details about the knowledge model for this domain are presented in [2].

The default view of the 3D ontology model created by Onto3DViz shows the static knowledge elements in the foreground and dynamic knowledge in the background. All the visual objects are labelled.

The dynamic knowledge component of the application ontology is shown in figure 3. The cylindrical node represents the objective and the cones under the objectives represent the tasks associated with their corresponding objective. Analogous to the *Thing* class in the static model, the node of *ObjectiveList* is the root of the hierarchical structure that represents dynamic knowledge. The sub-nodes under *ObjectiveList* are the objectives, each of which is associated with a set of tasks. The tasks associated with a particular objective are sorted by the task priority, which means a task located at a higher position has a higher priority than the task or tasks located at the lower positions. Moreover, the priority of a task is correlated to the size of the represented node. A task that has a higher priority is larger than the task that has a low priority.

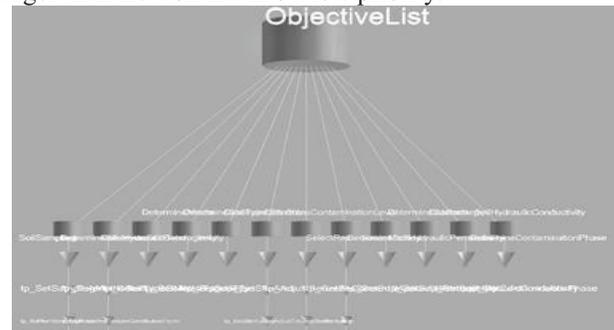


Figure 3. Dynamic knowledge model of the application ontology

The side view of the application ontology of the domain of selection of petroleum contamination remediation technology is shown in figure 4. From this perspective, it can be seen that the static knowledge and dynamic knowledge of the application ontology are connected together in the 3D model.

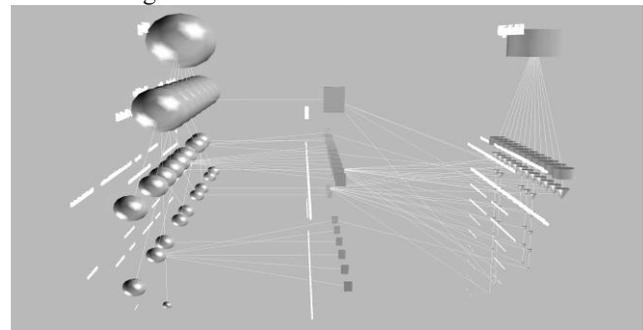


Figure 4. The side view of the visualized application ontology

Figures 3 and 4 provide some sample perspectives of the 3D model of the domain ontology generated by Onto3DViz. The user can also manipulate the default 3D model produced by the tool to produce many other views. Application of Onto3DViz for visualizing this domain ontology also constitutes a test and verification that Onto3DViz can visualize industrial domains of complexity similar to the domain of selection of petroleum contamination remediation.

## 5. DISCUSSION

Based on applications of the tools for representing the ontology of the petroleum contamination remediation selection domain, some observations about the strengths and weaknesses of the two tools can be made

While IMT has suggested that a knowledge engineer should first model static knowledge, and when the static knowledge model has been developed, the dynamic knowledge model can be specified. However, this suggested sequence of modeling is found to be not always practical. Often, the ontology author is uncertain what classes and properties are required; and as a consequence, the static knowledge model can include incorrect labels for classes or properties being attached to the wrong classes. Our experience suggests that a more fruitful approach is to use an iterative process of first specifying some dynamic knowledge, which can then serve to guide specification of the static knowledge required. By specifying tasks and objectives first, the user is forced to think about what functionality and classes are required. This approach can reduce the chances of errors.

It can be observed that Onto3DViz is able to successfully render the complex domain concepts and relationships among concepts in a 3D model, thereby creating a visualized knowledge model that IMT formulated. The user can easily examine and manipulate the 3D model by performing the operations of zooming, rotating, and translating using the computer mouse or keyboard. The 3D model clearly shows the hierarchical structure of the static knowledge; it can also support representing the tasks associated with each object and arranging the tasks in the correct order.

However, we observed a weakness in the current version of Onto3DViz. When the classes are close to each other, the visual objects and labels become overlapped, which makes it difficult for users to clearly see details of the model. This problem can be solved by adding a collision avoidance system to Onto3DViz in the future.

## 6. CONCLUSION AND FUTURE WORK

The objective of this research is to develop a suite of ontological engineering tools which enables dynamic knowledge modeling and visualization of an application ontology. An analysis of existing tools has revealed that existing tools are inadequate in that they lack support for (1) dynamic knowledge modeling and visualization, and for (2) visualizing large volume of information due to limitations of 2D graphics and space. Dyna has been created to address the first inadequacy, and Onto3DViz has been developed to address the second. We believe that Dyna and Onto3DViz are useful tools that can assist ontological engineers in creating, verifying, and visualizing application ontologies.

As an ontological engineering tool, Dyna has only targeted modeling and testing of dynamic knowledge.

There are other aspects of ontological engineering that require research attention such as support for ontology management and ontology versioning; these topics will be left for future research. To address the weaknesses noted in the current version of Onto3DViz, some improvements to the tool can be made. For example, Onto3DViz can be implemented as a Protégé-OWL editor plug-in, so that ontology visualization and editing can be done in real time. As well, more user controls can be implemented which will enhance user-friendliness of the tool.

## ACKNOWLEDGEMENT

We are grateful for the generous support of Research Grants from Natural Sciences and Engineering Research Council (NSERC) and the Canada Research Chair Program to the first author.

## REFERENCES

- [1] C. W. Chan, "From Knowledge Modeling to Ontology Construction", *Int. Journal of Software Engineering and Knowledge Engineering*, 14(6), Dec 2004.
- [2] Protégé, Thursday 04 Mar. 2009 <<http://protege.stanford.edu>>
- [3] Ontolingua, Thursday 04 Mar. 2009 <<http://www-ksl-svc.stanford.edu:5915/>>
- [4] OntoBuilder, Thursday 04 Mar. 2009 <<http://iew3.technion.ac.il/OntoBuilder/>>
- [5] KAON, Thursday 04 Mar. 2009 <<http://kaon.semanticweb.org/>>
- [6] Protégé OWLViz, Thursday 04 Mar. 2009 <<http://www.co-ode.org/downloads/owlviz/co-ode-index.php>>
- [7] Jambalaya, Computer Human Interaction & Software Engineering Lab (CHISEL) and its members, 2002-2008. Thursday 04 Mar. 2009. <<http://www.thechiselgroup.com/jambalaya> >
- [8] Alessio Bosca, Bonino, Paolo Pellegrino, "OntoSphere: more than a 3D ontology visualization tool", SWAP, 2005
- [9] SHriMP, Computer Human Interaction & Software Engineering Lab (CHISEL) and its members, 2002-2008. Thursday 04 Mar. 2009 <<http://www.thechiselgroup.org/shrimp>>
- [10] Storey, M.-A.D., Wong, K., Fracchia, F. and Müller, H., On Integrating Visualization Techniques for Effective Software Exploration. in *InfoVis '97*, (Phoenix, AZ, 1997), 38-45.
- [11] Eclipse, the Eclipse foundation. Thursday 04 Mar. 2009. <<http://www.eclipse.org/>>
- [12] Chen, Lin-Li, "Construction of an ontology for the domain of selecting remediation techniques for petroleum contaminated sites", M.A.Sc. Thesis, University of Regina, Canada, 2001

# Modeling and Testing a Knowledge Base for Instructing Users to Choose the Classification Task in Relational Data Mining

Lidia Martins da Silva

Centro Universitário Cândido Rondon - UNIRONDON  
Cuiabá, Mato Grosso

Ana Estela Antunes da Silva

Universidade Metodista de Piracicaba - UNIMEP  
Piracicaba, São Paulo

**Abstract—** The objective of this article is to integrate two domains: Knowledge Representation and Data Mining using a knowledge base as the model for such unification. This work pretends to contribute to instructional data mining by teaching users how to choose tasks to apply to their domain problem. The aim of the work is to model and test a knowledge base emphasizing the classification task of data mining. The emphasis is on the necessary knowledge for the understanding and choice of the classification task to be applied to data mining problems. The knowledge used to construct the knowledge base was acquired from experts, literature and data mining tools. Production rules were developed in order to create a model of the knowledge base. This model contains 25 rules which are executed using forward chaining reasoning. In order to get the values to execute the knowledge base eleven specific questions about the application of the classification task were created. Each question corresponds to one predicate of the knowledge base. The objective is lead users to get an affirmative or negative answer for the application of the classification task. The questions are answered according to each domain problem.

**Keywords:** Knowledge Representation, Data Mining, Classification, Knowledge Base.

## I. INTRODUCTION

Expert systems are developed to address complex problems in the real world that require interpretation and analysis of human experts and at the same time to reach conclusions and decisions which the human expert would reach if he/she were facing the same problems. These computer programs are used for executing rules on a base of knowledge making possible the solution of the problem. This solution can be a treatment, a diagnostic, a classification among others [12,18,28].

The heart of an expert system is its knowledge base. This component contains knowledge in the form of production rules, frames, semantic networks. This means that it contains a description of the knowledge needed to solve the problem addressed in the application. The knowledge base is the central element of an expert system and is responsible for structuring all knowledge about the application domain [20, 23, 28].

Data mining consists of a set of tasks that, through the use of specific algorithms, are able to exploit a large data set, creating from them, knowledge in the form of assumptions and rules. Daily companies accumulate different data in their databases making them real information treasures about the various processes and procedures of business functions, including data and habits of their customers, their success stories and failures. All these data can contribute to the company, referring trends and characteristics related to the company and its environment both indoors and outdoors, in view of rapid actions of his managers [9,25]. Currently, there are many proposals of expert systems for solving a wide range of sectors such as medicine, veterinary medicine, agriculture, mathematics, chemistry, geology, economics, law, education, computer science, among others [22,26,29,30,32].

However, in the data mining domain there is a lack of knowledge representation and explanation to help users choose the most appropriate task to be applied to their problems [11,24]. Most of the mining tools do not support the user in understanding the mining processes which includes the appropriate choice of a mining task such as classification, clustering and association rules [16,34]. This lack of representation of knowledge about data mining processes and the high level of technical knowledge necessary to perform such processes have risen the motivation for doing this work of modeling and testing a knowledge base for instructional data mining to assist users in their choice of a task for performing the data mining process.

This article is organized as follows: section I introduces the subject area and explains its motivation; section II describes the main concepts of data mining; section III is dedicated for the classification task of data mining; section IV presents the set of questions and the knowledge base for the classification task represented by production rules; section V shows some examples of tests of the production rules and finally are conclusion and future works.

## II. DATA MINING DOMAIN KNOWLEDGE

According to [14, 15] the set of activities of the Knowledge Discovery in Databases (KDD) process contains seven phases: data cleaning, data integration, data selection, data transformation, data mining, pattern evaluation and knowledge presentation. This article focus on the data mining phase. The data mining process is the application of a set of techniques which explore data in order to discover new patterns and relations in data. Because of the high level of association that can exist among data and of the high data volume it is difficult to discover such patterns without applying such techniques only by observing such data [6]. Although there are a lot of techniques that can be utilized, the data mining process still presents difficulties to be applied. In [24] some theoretical aspects of data mining are presented in order to make the process clearer.

According to [2], data mining is the exploration and analysis of large amounts of data to discover meaningful patterns and rules. According to the authors, the goal of data mining is to allow a company to improve its marketing, sales operations and customer service through the use of mining techniques. In [8] there is a definition of data mining and discovery of information in terms of standards or rules in large amounts of data. [14] considers data mining as a multidisciplinary field which includes areas such as: database, artificial intelligence, statistics and image processing. Information obtained through data mining process can be utilize by several areas: market analysis, production control, among others.

Another classification of data mining is in terms of qualitative and quantitative data. The qualitative data mining is used to find quality standards, or qualitative relationships in numerical data. A motivation for qualitative data mining comes from the fact that for some tasks, qualitative models are more suitable than the classic figures, the numerical models. This kind of process consists in finding patterns without recurring to numerical computation, and without the use of a quantitative model [5,31]. Data that are not numerical (ie, colors, names, reviews) are called qualitative data. To analyze this information, classification analysis is the best. This model of data mining is also known as the descriptive model [7].

The main tasks utilized to perform data mining are: association, classification and clustering. Association is most applied to problems which can be modeled using transactions. Each transaction is a line in a table which is searched for useful information. Classification is the task which separates all tuples of a table into classes. Clustering also separates data into classes. The main difference between classification and clustering is that to use classification it is necessary to previously know the classes. The clustering task allows the user to perform data mining without knowing previously the classes in which data will be aggregated [13,15,16].

In this work the data mining process, specifically the classification task, plays the role of the domain knowledge for the construction and test of a knowledge base. Other tasks such association and clustering are not considered and

although classification is advisable be applied to qualitative problems in data mining, quantitative problems are also included in the domain problem.

## III. THE CLASSIFICATION TASK DOMAIN

Classification is one of the most used tasks of data mining. This is due to the real life use of the classification concept. Every day people make use of such concept: which clothes to wear, which road is more appropriate to get to work etc. All these decisions are made based on certain degree of classification. People classify things according to certain characteristics. In organizations, classification task is appropriated to be used in contexts such as: fraud detection, commercial targets, medical diagnosis among others [14, 15].

According to [14, 15, 35], classification is a form of data analysis which can be used to extract models that describe important classes of data or predict future trends. This analysis can help to offer a better comprehension of general data. Classification aims at constructing a model to separate data into classes. According to the authors, the task is divided into two phases. The first phase is called training or learning phase which aims at constructing the classifier which describes a set of data separated into classes according to chosen characteristics. The second phase applies the classifier to new input data in order to classify them into the previous classes.

In order to construct a knowledge base to instruct the choice of the classification task in data mining, the domain problem was studied according to the main characteristics which could lead to the choice or the rejection of the classification task depending on the type of problem presented by the user. Following these characteristics are presented. Between parenthesis are the predicates used to construct the knowledge base. These predicates are related to each specific explanation about the classification task domain knowledge.

In [19], it is possible to verify that data sets that were previously divided into classes frequently contain several attributes (*various\_attributes*). These databases may contain many attributes which can be categorical or numeric attributes. An attribute is a data value assumed by the objects of a class. Name, age and weight are examples of attributes of Person objects. Color, weight and model are possible attributes of Car objects. Each attribute has a value for each object instance [14,15]. An example of a categorical attribute is color, whose domain includes values such as brown, black, white, etc [17]. The fact that data in the user problem presents several attributes, specially categorical ones is then an indicator that classification algorithms of data mining may be appropriate to be applied to the problem.

The concern about the type of attributes presented in the user's databases has risen another characteristic to be considered for the choice of the classification task: whether the attributes are all numeric (*all\_numeric*). In [14,15] an attribute is said to be numerical when it can be scaled and represented by a number. If we had records of people,

examples of numerical attributes could be: age, height, weight, among others. Numerical attributes may eventually be transformed into categorical, for example, the age attribute could have their numerical values represented by the categorical values: "child" or "adult" [14,15]. If it is possible to categorize a numerical attribute, then it is possible to apply classification algorithms. However if there are several numeric attributes in the database and it is not possible to transform them into categories then it not advisable to use classification algorithms.

It can be observed in [2] that classification is more appropriate when there are large volumes of data (large\_volume\_data). According to the author, most of classification algorithms requires large amounts of data in order to build the classifier model. This makes "large amounts of data" also a desirable characteristic of the user's database in order to apply classification algorithms.

Another point for using classification algorithms is the guarantee that the user can be able to identify one categorical attribute such as type\_of\_investment which can assume values like safe or risky (categorical\_attribute).

It also is important to know if the user's database is a transactional database (transaction). According to [14, 15], a transactional database consists of a file in which each record represents a transaction. A transaction typically includes a unique identification number, and a list of items that make up the operation (such as items purchased in a store). When the user's data is under the form of a transactional database is more appropriate to use association algorithms rather than classification ones.

Considering the problem that the user needs to apply mining algorithms it is important to know his/hers goals in terms of the data. If the user wants to predetermine a target to perform data mining then it is highly advisable to use classification algorithms (single\_target). According to [19], mining data with association rules does not require a goal to be predetermined, while mining data with classification rules does require a target.

For [8], classification is the process of finding a model that describes different classes of data. These classes are predetermined. In the first step, a model is constructed to describe a particular set of data classes or concepts. The model is constructed by analyzing database tuples described by attributes. Each tuple belongs to a default class, as determined by an attribute called the class label [14, 15]. This class label is the target attribute. Besides having as objective of data mining a classification process the user has to be able to identify among the attributes one that can be used as class label (identify).

Without such attribute the classification task is not able to be applied. A target attribute can be categorical, or discrete, determining classes or categories. This attribute can have values as YES or NO, a code belonging to a range of integers, such as (1 ... 10), etc. [10]. Attributes that take a variety of values can be grouped into certain categories, for example, the monthly income of the client can be grouped into 4 categories: low, medium, medium-high and high, etc.

In data mining using the classification task, the target attribute must be identified or created.

Another important aspect in the classification task is the existence of a training data set (training\_data). Records or individual samples that make up the set of training data are called training samples and are randomly selected [14, 15]. Most of the problems of pattern classification of real interest is in two phases: training (learning) an existing database and the phase of generalization, in which data presented were not used in training. The classification algorithm builds a classifier by analyzing a set of training data. Each instance of the set of training data must belong to a class previously defined by an attribute called the class attribute. The greater the amount of data the more information could be used in training. But, it is necessary to have and identify such data set.

After training the chosen data set it is necessary to have a different data set to test the classifier (different\_sets). According to [1], classification rules are tested on another database, completely independent of the database of training, called a database or set of tests. So, it is necessary to obtain two different data sets each one aiming at achieving different purposes.

R01. <b>If</b> various_attributes=yes <b>Then</b> classification=medium
R02. <b>If</b> various_attributes=no <b>Then</b> classification=low
R03. <b>If</b> all_numerical=yes and classification=medium <b>Then</b> classification=low
R04. <b>If</b> all_numerical=no and various_attributes = yes <b>Then</b> classification=medium
R05. <b>If</b> large_data_volume=yes and classification = medium <b>Then</b> classification=high
R06. <b>If</b> large_data_volume=no <b>Then</b> classification=low
R07. <b>If</b> categorical_attribute=yes and classification = medium <b>Then</b> classification=high
R08. <b>If</b> categorical_attribute=no and (classification=medium or classification=high) <b>Then</b> task_not_identified=yes
R09. <b>If</b> transaction=yes <b>Then</b> classification=low
R10. <b>If</b> transaction=no and classification = medium <b>Then</b> classification=high
R11. <b>If</b> single_target=yes and classification=medium <b>Then</b> classification=high
R12. <b>If</b> single_target=no and classification=low <b>Then</b> classification=low
R13. <b>If</b> identify=yes and classification = medium and single_target = yes <b>Then</b> classification=high
R14. <b>If</b> identify=no and single_target=no <b>Then</b> task_not_identified=yes
R15. <b>If</b> training_data=yes and classification = medium <b>Then</b> classification=high
R16. <b>If</b> training_data=no <b>Then</b> classification=low
R17. <b>If</b> training_data=yes and single_target=yes and identify=yes and classification=high <b>Then</b> classification=very_high
R18. <b>If</b> different_sets=yes and classification = medium <b>Then</b> classification=high
R19. <b>If</b> different_sets=no <b>Then</b> classification=low
R20. <b>If</b> different_sets=no and training_data=no and classification=high <b>Then</b> task_not_identified=yes
R21. <b>If</b> applied_results=yes and classification= medium <b>Then</b> classification=high
R22. <b>If</b> applied_results=no <b>Then</b> classification=low
R23. <b>If</b> decision_results=yes and applied_attribute=yes and classification=high <b>Then</b> classification=very_high
R24. <b>If</b> decision_results=no and classification= low <b>Then</b> Classification=low
R25. <b>If</b> task_not_identified= yes <b>Then</b> classification = not_possible

Figure 1. Knowledge Base Representing the degree of use adequacy of the classification task.

According to [14, 15], the task of classification consists of building a model that will be used to classify data in order to categorize them into classes, i.e., a classifier like previously mentioned. Companies of any market segment, requiring strategic knowledge, see the data mining process as a way to improve the company's profitability and a safer basis to make better decisions regarding the future forecast. In this way, the rules discovered in the process of data mining using classification task can be used to categorize future data sample (applied\_results). If companies desire to have a model to classify future input data then the classification task can be a possibility [11].

One last point that supports the choice of the classification task is the use of the data resulting from the classification process (decision\_results). On the contrary of the results obtained by the application of association rules, for example, the results of the classification tasks do not have to be immediately used. The classes can be studied for a period of time and medium term decisions may be taken during this period.

The factors previously considered were the basis for the construction of a knowledge base for the choice of the classification task in data mining. Fig. 1 represents the knowledge base which is explained in section IV.

#### IV. KNOWLEDGE BASE OF THE CLASSIFICATION TASK

A knowledge base is a set of representations of actions and events in the world. Each representation is called a sentence. According to [28], sentences are expressed in a specific language based on different techniques of representation, such as production rules, semantic nets, frames and logic. It is considered the main part of a knowledge based system and contains knowledge under one or more of the techniques mentioned above. In this work production rules are used to represent the knowledge base.

According to [27], these systems are inspired by the idea that the process of human decision making could be modeled by rules of type IF condition THEN conclusions and actions. The rules express logical relationships and equivalences settings to simulate human reasoning. This kind of representation is called production rules.

Fig. 1 presents the production rules that represent the knowledge base of the classification task. All predicates were created based on the concepts presented. For instance, the predicate "various\_attritbutes" represents the knowledge concerning the fact that having several attributes is a good feature of the database when choosing a classification task to be applied to the domain problem. The predicates in the rules follow the same order in which they were presented and explained in the text and, as mentioned earlier, for each part of the explanation there is a correspondent predicate.

#### V. EXECUTION AND TEST OF THE KNOWLEDGE BASE

The knowledge base was created specifically for the classification task with the following levels of adequacy for the application of the classification task: low, medium, high, very high and not\_possible. The order of the antecedent and

consequent parts of the rules were designed to reflect these levels of adequacy. The reasoning used to execute the knowledge base is forward chaining [28]. In order to get values to the attributes in the knowledge base specific questions are asked to users. The answers to these questions are based on eleven of the thirteen predicates existing in the knowledge base. The predicates classification and task\_not\_identified do not have a correspondent question as they assume values at the moment of the execution of the rules. These questions are presented in Fig. 2.

- Do you wish to identify several attributes for the mining process?
- Are chosen attributes all numeric?
- Is there a large volume of data in your database?
- Is the classification attribute a category?
- Are chosen attributes part of a transactional database?
- For your mining process do you need to choose only one attribute to characterize the whole process?
- Can you identify such attribute among the chosen attributes or create it?
- Is there a data set to be used for a training process?
- Is there a different data set to be used for a testing process?
- Do you wish to use mining results in order to apply them to other available data?
- Do you wish to use mining results in order to make an immediate decision about your organization?

Figure 2. Questions asked to users

After answering these questions the knowledge base can be executed. One example of execution is presented in Figure 3.

- Attribute Values:**
- 1 - various\_attributes : yes;
  - 2 - all\_numerical: no;
  - 3 - large\_data\_volum: no;
  - 4 - categorical\_attribute: yes;
  - 5 - transaction : no;
  - 6 - single\_target: yes;
  - 7 - identify : yes;
  - 8 - training\_data: yes;
  - 9 - different\_sets: yes;
  - 10 - applied\_results: yes;
  - 11 - decision\_result: yes.
- Successful Rules:** R01, R04, R05, R07 and R17

Figure 3. Example of forward chaining reasoning.

In the example in Fig. 3, the result of the execution of the knowledge base is a very\_high level of adequacy for the application of the classification task in the problem domain presented by the user through the answers of the questions presented in Fig.2. The result makes sense since the main characteristics for the application of the classification task are presented in the user domain problem.

Table 01 shows some tests made in the knowledge base. The columns contain numbers and the lines contain values. The value "Y"(N) in the line means that the value of the predicate in the correspondent column is Yes(No). In order to make Table1 able to be displayed, the name of the predicates were replaced by numbers. A legend for the predicates is presented in Fig. 4. The choice of the values for executing the tests was performed by looking at the questions and combinations of answers which would result in at least one answer for each possible level of adequacy: very high, high, medium, low and not\_possible.

TABLE I. TESTS OF THE KNOWLEDGE BASE

	2	3	4	5	6	7	8	9	10	11	12	13
Y	N	Y	Y	N	Y	Y	Y	Y	Y	Y	VH	VH
N	Y	N	N	Y	N	N	N	N	N	Y	L	L
Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	VH	VH
Y	Y	Y	N	Y	N	Y	N	N	N	Y	L	L
Y	Y	Y	Y	Y	N	N	N	N	N	N	NP	NP
N	N	N	N	N	Y	Y	Y	Y	Y	Y	L	L
Y	Y	Y	N	N	Y	Y	Y	N	N	N	L	L
Y	N	Y	N	Y	N	Y	N	Y	N	N	NP	NP
N	Y	N	Y	Y	N	Y	N	N	Y	Y	L	L
N	N	Y	N	N	Y	N	Y	N	N	N	NP	NP
Y	N	N	N	N	N	N	N	N	N	N	NP	NP
N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	L	L
Y	N	Y	N	N	N	N	N	N	N	N	NP	NP
N	Y	N	Y	Y	Y	Y	Y	Y	Y	N	M	L
N	Y	Y	N	N	Y	N	N	Y	Y	N	NP	NP
Y	N	Y	N	N	Y	Y	Y	N	N	N	NP	NP
N	N	N	N	N	N	N	Y	Y	Y	Y	NP	NP
Y	Y	Y	Y	Y	Y	Y	N	N	N	N	M	L
N	Y	Y	Y	Y	Y	Y	N	N	N	N	M	L
N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	VH	VH
N	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	M	L
Y	N	Y	Y	Y	N	Y	Y	Y	Y	N	L	L
N	N	N	N	N	Y	Y	N	N	N	N	L	L
Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y	NP	NP

Tests of the knowledge base were performed following the supervised learning paradigm [4]. Expected results were previously calculated - according to the literature presented - and then compared to the obtained results. The predicates expected\_result and obtained\_result represent the predicates used for supervising the tests.

LEGEND	
1 - various_attributes	11 - decision_result
2 - all_numerical	12 - expected_result
3 - large_data_volume	13 - obtained_result
4 - categorical_attribute	VH - very high
5 - transaction	M- medium
6 - single_target	H - high
7 - identify	L - low
8 - training_data	NP - not_possible
9 - different_sets	TNI - task_not_identified
10 - applied_results	

Figure 4. Names of the predicates in Table 1.

Considering, for example, the line 14 in Table 01 the expected\_result value is “medium level of adequacy” for the application of classification and the result obtained by the execution of the knowledge base is “low level of adequacy”. According to the answers to the given questions and applying a forward chaining reasoning to the knowledge base, the following rules were successfully verified: R02, R06, R09 and R24.

In this example of line 14 of Table 1, because the predicates that received a “no” value (1, 3 and 11) do not forbid the classification to be applied and because all other predicates received a value “yes”, the result expected from the execution was “medium level of adequacy”. This means that some rules of the knowledge base are being severe with some values. However in the 23 examples of Table 1 only

three of them presented different values between obtained and expected results.

The levels of adequacy low, medium, high, very high and not possible were chosen in order to demonstrate different possibilities of application for the classification task once there are several parameters to be analyzed and a binary answer such as suitable(not suitable) would restrict the reasoning applied to the knowledge base.

In order to perform all tests it would be necessary to make 2048 (2<sup>11</sup>) tests which result from the combination of 11 variables in the knowledge base with the possibility of response of each variable (yes or no). Thus, for a first level of testing, the most obvious answers were chosen. For this set of responses, the tests showed that most of the obtained results were consistent with the expected results.

### CONCLUSION

This paper presents the modeling and testing of a knowledge base for instructing users to choose the task of classification through the use of questions that lead them to finding out if the classification task is suitable to be used in their domain problem. In order to achieve this goal it was necessary to study several characteristics of the classification task in data mining. This study is very important once mining tools do not propose this kind of explanation to users [16, 34]. Users must have knowledge enough to know which task should be applied to their problem.

According to the knowledge acquired about the classification task, a knowledge base was modeled and eleven questions were formulated. The base contains twenty-five rules and is executed according to the answers of the given questions. For the validation of the production rules were created twenty-three examples. Among the contributions of this work the following are pointed out:

- creation of questions to instruct users about the choice of the classification task when applying mining techniques to their problems;

- acquisition of knowledge about classification task;

- knowledge modeling to solve the problem of the adequacy of application of the classification task in data mining for general domains of problems;

- initial tests of the knowledge base.

As future work authors propose:

- The performance of more tests to validate the knowledge base;

- The creation of pertinence functions for the predicates: low, medium, high and very high, turning them fuzzy sets;

- The expansion of the knowledge domain by including the association task;

- The insertion of the knowledge base in the KIRA tool [3, 21,33]. The tool is an instructional tool for the data mining process including the phases: problem description; data cleaning; data selection; application of mining tasks and data analysis.

## REFERENCES

- [1] AMO, S. Course of Data Mining. Masters Program in Computer Science, Federal University of Uberlandia, 2003. Available at: <<http://www.deamo.prof.ufu.br/CursoDM.html>>. Accessed: 01 Out. 2009.
- [2] BERRY, M. J. A.; LINOFF, G.. Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management. 2 ed. Wiley Publishing. USA, 2004.
- [3] BINDILATTI, A. Modeling a Knowledge Base for Data Mining Process of the Kira Tool . Under-graduation Research Project. Unimep: Methodist University of Piracicaba. Piracicaba, São Paulo, 2009.
- [4] BRAMER, M. Principles of Data Mining. Springer-Verlag London Limited 2007.
- [5] BRATKO, I. Qualitative Modeling. Faculty of Computer and Information Sc.,University of Ljubljana. Available at: [http://lopes1.fov.uni-mb.si/za\\_PES\\_PO\\_net/005Bratko.pdf](http://lopes1.fov.uni-mb.si/za_PES_PO_net/005Bratko.pdf). Accessed: 18 Fev. 2010.
- [6] CARVALHO, L. A. V. Data Mining – Data Mining in Marketing, Medicine, Economy, Engineering and Administration. 2005.
- [7] Data Mining. Available at: <http://www.answers.com/topic/data-mining>. Accessed: 18 Fev. 2010.
- [8] ELMASRI, R. & NAVATHE, S.B. Systems Database. 4th ed. Pearson Brazil, 2005.
- [9] FAYYAD, Usama M. Data Mining and Knowledge Discovery: Making Sense out of data. IEEE Expert, Los Alamitos, v.11, n.5, p. 20-25, Out. 1996.
- [10] FREITAS, A. A., Lavington, S. H. Mining Very Large Databases With Parallel Processing. Kluwer Academic Publishers. 1998.
- [11] \_\_\_\_\_. A.A. Understanding the crucial differences between classification and discovery of association rules: a position paper. ACM SIGKDD Exploration Newsletter. Vol. 2. Pp 65-69. ISBN 1931-0145. 2000.
- [12] GIARRATANO, J.; RILEY, G. Expert systems: principles and programming. 3. ed., Boston: PWS, 1998.
- [13] GUHA, S.; RASTOGI, R.; SHIM, K. ROCK: A robust clustering algorithm for categorical attributes. Information Systems. Stanford University. Stanford, CA 94305, 25(5), 2000.
- [14] HAN, J.; KAMBER, M. Data Mining - Concepts and Techniques. 2 ed. Nova York: Morgan Kaufmann, 2006.
- [15] DATA MINING - Concepts and Techniques. San Francisco, EUA: Morgan Kaufmann, 2001. 550 p.
- [16] HAN J.; CHIANG J. Y. CHEE S.; CHEN J.; CHEN Q.; CHENG S. GONG W.; KAMBER M.; KOPERSKI K. ; LIU G. LU Y.; STEFANOVIC N.; WINSTONE .; XIA BETTY B.; ZAIANE O. R.; ZHANG S.; ZHU H. DBMiner: A System for Mining Knowledge in Large Relational Databases. Proceedings of International Conference on Knowledge Discovery in Databases, Portland. USA. pp. 250-255. 1996.
- [17] HARRISON, Thomas H. Intranet Data Warehouse. São Paulo, Berkeley Brasil, 1998.
- [18] KULIKOWSKI, C.A. Artificial Intelligence methods and systems for medical consultation. IEEE Trans. Patt. Anal. Mach. Recogn., 2(5): 464-4765, 1980.
- [19] .LIU B.; HSU W.; MA Y. Integrating Classification and Association Rule Mining. Proceeding of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98). Nova York, USA. pp. 80-86. 1998.
- [20] LUGER, G. F. Artificial Intelligence Structures and Strategies for Complex Problem Solving. Fifth Edition. England. Addison-Wesley. 2005.
- [21] MENDES, E. F. Automation Technical Data Mining Aided Guides. Dissertation. Methodist University of Piracicaba. Piracicaba. São Paulo, 2009.
- [22] MILLER, P.L., Evaluation of artificial intelligence systems in medicine, Proc. IX SCAMC Washington DC, (IEEE 1985) 281-286.
- [23] NIKOLOPOULOS, C. Expert systems: introduction to first and second generation and hybrid knowledge based systems. New York: Marcel Dekker, Marcel Dekker Inc, 1997. 331 p.
- [24] OHSUGA, S. Difference Between Data Mining And Knowledge Discovery --A View To Discovery From Knowledge-Processing. 2005 IEEE International Conference on Granular Computing. Vol. 1. Pp 7-12. ISBN 0-7803-9017-2. 2005.
- [25] OLSON, D.L., DELLEN, D. Advanced Data Mining Techniques. Springer. Verlag – Berlin – Heidelberg. 2008.
- [26] PASSOS, E. L. Artificial intelligence and expert systems for Everyone. Rio de Janeiro, Soc. Ben. Guilherme Guinle, 1989.
- [27] REZENDE, S. O. Data Mining. In Rezende, S. O.: Smart Systems - Fundamentals and Applications. Barueri, SP: Manole, 2003.
- [28] RUSSELL, S. J.; NORVIG, P. Artificial Intelligence, A Modern Approach. 2nd ed. Upper Saddle River, New Jersey, USA: Pearson Education, 2003.
- [29] SAVARIS, S. V. A. M. Expert system of first aid for dogs. Thesis (Graduate Studies in Computer Science) – Federal University of Santa Catarina, Florianópolis. 2002. 156 f
- [30] SONAR, R.M. Integrating intelligent systems using an SQL-database. Expert system with applications, v. 17, i. 1, p. 45-49, July 1999.
- [31] SUC. Ivan Bratko and Dorian. Qualitative Data Mining and Its Applications. Faculty of Computer and Information Science, University of Ljubljana, Slovenia Journal of Computing and Information Technology - CIT 11, 2003, 3, 145-150
- [32] VEDOVELLO, R.; RIEDEL, P.S.; BROLLO, M. J.; HAMBURGUER, D.S.; CAMARGO, A.A.X. – 2002. Modeling and architecture of a System Manager Environmental Information (SGIG) as a product of geological and geotechnical assessments. Ouro Preto, MG. In: Brazilian Congress of Engineering Geology.
- [33] VIEIRA, M. T. P.; SILVA, A. E. A.; PEIXOTO, C. S. A.; GOMIDE, R. S.; MENDES, E. F. KIRA - a tool based on guides and domain knowledge to instruct data mining. IADIS International conference Applied Computing 2009. pp 12-16. ISBN: 978-972-8924-97-3.
- [34] WEKA. Available at: <http://www.cs.waikato.ac.nz/ml/weka/>. Accessed: 20 Dez. 2009.
- [35] WITTEN, I. H.; FRANK, E. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. San Diego: Morgan Kaufmann Publishers, 2000.

# gOntt, a Tool for Scheduling and Executing Ontology Development Projects

Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, Oscar Muñoz, Martín Vigo  
*Ontology Engineering Group. Departamento de Inteligencia Artificial. Facultad de  
Informática. Universidad Politécnica de Madrid*  
{mcsuarez, asun, omunoz}@fi.upm.es, mvigo@delicias.dia.fi.upm.es

## Abstract

*Nowadays the ontology engineering field does not have any method that guides ontology practitioners when planning and scheduling their ontology development projects. The field also lacks the tools that help ontology practitioners to plan, schedule, and execute such projects. This paper tries to contribute to the solution of these problems by proposing the identification of two ontology life cycle models, the definition of the methodological basis for scheduling ontology projects, and a tool called gOntt that (1) supports the scheduling of ontology developments and (2) helps to execute such development projects.*

## 1. Introduction

Planning and scheduling are related activities that are applied in different contexts, such as civil engineering, software engineering, etc. While planning<sup>1</sup> is the act of drawing up plans, a series of steps to be carried out to achieve an objective, scheduling<sup>1</sup> is defined as the activity to set order and time to planned events. Scheduling should be performed after planning; and both are crucial in any development project.

In Software Engineering, every development project has a life cycle [1], which is produced by instantiating a particular life cycle model. Life cycle models can be seen as abstractions of the phases through which a product passes along its life.

To properly manage software development projects, it is crucial to have knowledge of the entire software development life cycle [2]. Software engineers always plan and schedule every development project before starting it. The project plan devises the tasks to be done and the actors to perform them. To estimate the

effort required to perform each task, techniques such as [2] PROBE and COCOMO II can be used.

The project schedule links the tasks to be done with the resources in order to support their performance. The most common form of representing schedules is to use a Gantt chart [2]; and the most popular tool for creating a project schedule is Microsoft Project [2]. To create a schedule, Microsoft Project provides three different ways: (a) from scratch; (b) from a project template (commercial template, report template, etc.) selected by the user using a template library; and (c) from an existing project. However, according to our knowledge, any tool for managing project schedules provides guidelines on how to execute the project.

Ontologies are used for making knowledge explicit and allowing it to be shared. One of the keys when building ontologies is to plan and schedule the ontology development. However, in Ontology Engineering, planning and scheduling are still in their early stages. Only METHONTOLOGY [3] defines the scheduling activity, but it does not provide guidelines for helping ontology developers to plan and schedule their projects. Other methodologies, such as On-To-Knowledge [4] and DILIGENT [5], do not include these activities in their developments. Regarding the calculation of cost estimation of projects, the only technique available is ONTOCOM [6], a model that predicts the costs of ontology development projects.

*The ontology engineering field lacks methods for guiding ontology developers when planning and scheduling their ontology development projects. Furthermore, the template library of Microsoft Project does not have project templates oriented to ontology development projects; on the other hand, at present, no tool can provide ontology developers with ontology project schedules in the form of Gantt charts, nor can recommend developers which methodological guidelines and tools should be used for executing a process or an activity in the ontology development.*

Thus, the innovation of this paper is that (1) it identifies two ontology life cycle models, (2) defines

---

<sup>1</sup> wordnet.princeton.edu/perl/webwn

the groundings for scheduling ontology projects, and (3) presents gOntt, a tool that supports the scheduling of ontology developments and their guided execution.

The paper is organized as follows: Section 2 presents the scheduling activity. Section 3 deals with ontology life cycle models. Section 4 summarizes the methodological groundings for the scheduling activity. Section 5 shows the main gOntt functionalities. Section 6 presents an evaluation of gOntt. Finally, Section 7 provides the conclusions.

## 2. Scheduling ontology developments

Scheduling [7] refers to the activity of identifying the different processes and activities to be performed during the ontology development as well as their arrangement, and the time and resources needed for their completion. An important task within this activity is the establishment of the *ontology network life cycle*, that is, the specific ordered sequence of processes and activities that ontology practitioners have to carry out during the life of the ontology network.

To establish the particular schedule for the ontology development, four questions have to be answered: (1) Which life cycle model is the most appropriate for the development?; (2) Which processes and activities should be carried out in the ontology development?; (3) Which order and dependencies exist among processes and activities?; and (4) How many resources are needed for the development of the ontology?

The first three questions are related to the establishment of the ontology life cycle, and their responses would provide a plan for the ontology development. The fourth question (out of the scope of this paper) is related to the inclusion of time and human resources restrictions, and its response would provide the concrete schedule for the ontology development. The information about how many people should be involved in the ontology development can be obtained using the ONTOCOM model [6].

## 3. Ontology network life cycle models

The *ontology network life cycle model* establishes in an abstract way how to develop an ontology network development project. The ontology network life cycle is created by mapping the processes and activities identified in the ontology development onto a selected ontology life cycle model. While the model refers to a general framework, the life cycle refers to the concrete sequence of processes and activities. In this regard, while METHONTOLOGY and DILIGENT propose a model based on evolutionary prototypes, On-To-Knowledge proposes an incremental and cyclic model.

Along this section we will try to show that there is not a unique life cycle model valid for all ontology development projects and that each life cycle model is appropriate for projects with different features.

For this reason, we propose two ontology network life cycle models; the *waterfall ontology network life cycle model* and the *iterative-incremental ontology network life cycle model*. These models were created by (1) adapting the life cycle models described in Software Engineering to the characteristics of the ontology development (that is, the acquisition of knowledge, the evaluation and assessment of the different phase outputs, project and configuration management and documentation should be performed in all of the phases); (2) reusing the ideas presented by Larman [8]; and (3) analyzing our experiences in different ontology development projects.

### 3.1. The waterfall life cycle model

The waterfall model represents the stages of the ontology development as a waterfall, where a concrete stage must be completed before the following stage begins and where backtracking is permitted from the maintenance phase to the phase that follows that of the requirements. Taking into account the importance of reusing and reengineering knowledge resources as well as ontology merging in the ontology network development, we define five different versions of the waterfall model (as presented in Figure 1).

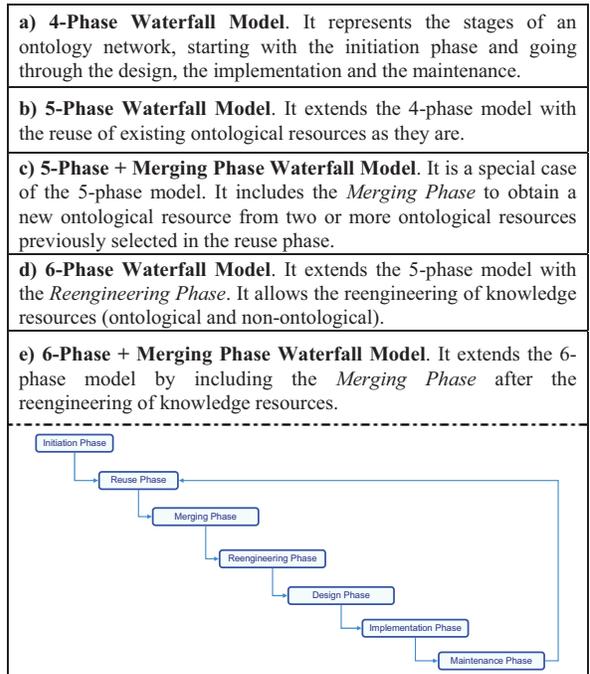


Figure 1. Waterfall model family

This model is recommended in projects that

- Have closed, non-ambiguous, unchangeable and completely known requirements at the beginning of the ontology development.
- Do not last long (e.g., 2 months).
- Re-implement an existing ontology or part of it in a different formalism or language.
- Transform a particular knowledge resource (e.g., ISO standards or thesauri) into an ontology.
- Cover a small and well-understood domain.

### 3.2. Iterative-incremental life cycle model

This model organizes the ontology development in a set of iterations (or short mini-projects with a fixed duration). Any iteration is scheduled as an ontology development project that uses one of the waterfall model versions shown in Section 3.1.

The model proposes the successive improvement and extension of the ontology by means of performing multiple iterations with feedback and adaptation. Thus, the ontology grows incrementally along the development. In each iteration, new or modified requirements are allowed for. The number of iterations will depend on the knowledge we may have of the requirements at the beginning of the project. The result of any iteration in this model is an ontology that meets the requirements identified in the iteration.

It should be noted that when using this model, no backtracking is allowed between the phases of a particular iteration because the refinement should be performed in the next iteration, and that in the initiation phase of each iteration revisions of ontology requirements and schedule should be carried out.

This model is recommended in ontology projects

- With large groups of developers in which complex scenarios are considered, such as reengineering non-ontological resources or aligning ontological resources.
- In which requirements are not completely known at the beginning or can change during the ontology development.
- In which requirements have different priorities.

## 4. Methodological groundings for scheduling ontology development projects

We propose to carry out the scheduling of ontology development projects based mainly on the scenarios identified in the NeOn Methodology [9]. For this reason and to be able to answer the first three questions presented in Section 2, we did some research that yielded the following results: (1) a set of questions that

help to select the most appropriate version of the waterfall life cycle model; (2) the correspondences between the phases of the life cycle model and the processes and activities; and (3) the order and dependencies between processes and activities.

### 4.1. The most appropriate model version

To select a particular waterfall model version from those presented in Section 3.1, we propose the set of natural language questions displayed in Table 1. These questions are related to the different scenarios identified in the NeOn Methodology [9]. If one or more questions of those proposed are answered affirmatively, then several candidate models could be used. In that case, the model version selected should be the most specific one (e.g., 5-phase is more specific than 4-phase, 6-phase + merging phase is more specific than 6-phase). Otherwise, if all answers are negative, then the 4-phase waterfall model is selected by default.

**Table 1. Questions and model versions**

<i>Will you use any non-ontological resource (NOR) in your ontology development?</i>	6-Phase
<i>Will you use any ontological resource in your ontology development?</i>	5-Phase
<i>Will you use and modify any ontological resource in your ontology development?</i>	6-Phase
<i>Will you use and merge a set of ontological resources in your ontology development?</i>	5-Phase + Merging Phase
<i>Will you use, merge, and modify a set of ontological resources in your ontology development?</i>	6-Phase + Merging Phase
<i>Will you use ontology design patterns in your ontology development?</i>	5-Phase
<i>Will you restructure your ontology?</i>	4-Phase
<i>Will you develop your ontology in different natural languages?</i>	4-Phase

### 4.2. Model phases and processes and activities

Processes and activities should be carried out in a particular phase of the selected ontology network life cycle model to fulfill the purpose and outcome of that phase. Table 2 presents an excerpt of the matching between model phases and process and activities. The processes and activities are defined in the NeOn Glossary [7] whereas the phases are defined in our repository of models and these are the initiation phase, the reuse phase, the merging phase, the reengineering phase, the design phase, the implementation phase, and the maintenance phase.

**Table 2. Model phases, processes and activities**

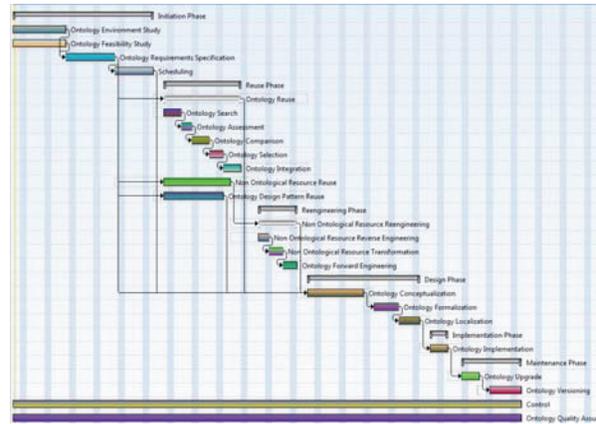
<b>Initiation Phase</b>	O. Requirements Specification O. Scheduling O. Evaluation	
<b>Reuse Phase</b>	NOR Reuse O. Search O. Reuse	O. Statements Reuse O. Evaluation
<b>Merging Phase</b>	O. Aligning O. Evaluation	
<b>Reengineering Phase</b>	NOR Reengineering O. Modularization	O. Evaluation
<b>Design Phase</b>	O. Conceptualization O. Evolution	O. Localization O. Evaluation
<b>Implementation Phase</b>	O. Evaluation	
<b>Maintenance Phase</b>	O. Evaluation	

### 4.3. Dependencies of processes and activities

The preliminary order in which processes and activities should be performed is determined by the three following major factors:

- 1) The selected ontology network life cycle model dictates an initial ordering of processes and activities, based on the order in which model phases should be performed.
- 2) The availability of output information from one process or activity could affect the start of another process or activity. For example, the ontology requirements specification activity should be performed before the scheduling one.
- 3) Processes and activities might be executed in a parallel way. For example, the ontological resource reuse could be performed in parallel with the non-ontological resource reuse rather than in serial execution if there are enough developers to carry out the reuse in a parallel fashion.

These dependencies have been represented in scheduling templates in the form of Gantt charts. The scheduling templates show ontology project default plans based on the different and possible combinations among life cycle models, scenarios [9], and processes and activities. We have identified and represented 112 templates that can be used as preliminary schedules. Figure 2 shows one of the scheduling templates. This template is used when the model is the 6-phase waterfall and the reuse of non-ontological resources, ontological resources, and ontology design patterns, as well as the localization of the ontology are considered.



**Figure 2. Example of a scheduling template**

## 5. gOntt description

gOntt is a tool for scheduling and executing development projects. It is implemented as a NeOn Toolkit<sup>2</sup> plug-in with the following main features:

- (a) It uses templates oriented to schedule ontology developments.
- (b) It generates the scheduling of ontology developments in the form of Gantt charts following the basis presented in Section 4.
- (c) It informs ontology developers about how to carry out a process or an activity using prescriptive methodological guidelines. It also informs about the specific NeOn Toolkit plug-ins to be used.

We describe gOntt functionalities for scheduling ontology projects and for helping in their executions.

### 5.1. Scheduling an ontology development

gOntt provides support to ontology developers so that they can (a) decide which ontology life cycle model is the most appropriate for building their ontologies (waterfall or iterative-incremental) and which processes and activities should be carried out and in which order (e.g., specifying ontology requirements before reengineering a knowledge resource into an ontology), and (b) create a graphical representation in the form of a Gantt chart with the processes and activities needed, including time restrictions between them. Schedules for ontology development projects can be created either *from scratch* or *in a guided way*.

**From scratch:** gOntt allows the developer to include processes, activities, phases, as well as restrictions among them, according to his needs.

<sup>2</sup> <http://www.neon-toolkit.org/>

**In the guided way:** gOntt creates a preliminary plan for the ontology development by means of (1) templates that schedule ontology projects and (2) a simple wizard that contains intuitive questions implicitly allowing the ontology developer to select the ontology life cycle model and the processes and activities needed in his development. To answer such questions the ontology developer should take into account the ontology requirements and the type of candidate knowledge resources to be reused.

gOntt uses internally the methodological foundations explained in Section 4.

gOntt main output is the initial plan for building the ontology in the form of a Gantt chart, which the developer can modify later on (a) by including, modifying, or deleting processes, activities and phases; (b) by changing order and dependencies among processes and activities; and (c) by including resource assignments and restrictions to the plan. This functionality to generate preliminary plans provides a great advantage over the tools that schedule projects.

## 5.2. Helping to execute an ontology project

With the aim of helping ontology developers to carry out a particular process or activity, gOntt provides *prescriptive methodological guidelines* by means of (1) a **filling card** that includes the process or activity definition, its goal, the inputs and outputs, the performer of the action, and the time required, and (2) a **workflow** that describes how the process or the activity should be performed with its inputs, outputs, tasks and actors involved. Figure 3 presents the methodological guidelines for the ontology requirements specification activity.

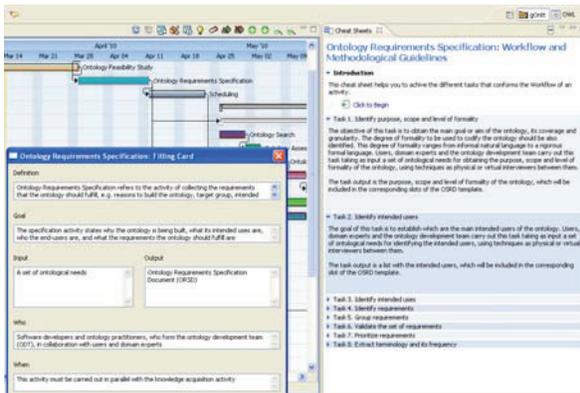


Figure 3. Example of methodological guidelines

In addition, gOntt provides a *direct and automatic access to NeOn Toolkit plug-ins*<sup>3</sup> associated to each process and activity. Besides, gOntt displays a quick-start guide for using the plug-in launched.

## 6. Evaluation of gOntt

We conducted an experiment using gOntt in two settings that involved Master and PhD students. Both groups of students had chosen the ‘Ontology Engineering and Semantic Web’ subject. The students executed the experiment during the period of October-November 2009. The students (working in pairs) had to develop an ontology network in a free domain. They were provided with (1) a set of mandatory processes and activities (e.g., to specify requirements and to reuse different types of knowledge resources) and (2) a set of optional processes and activities (e.g., localization, evaluation, and modularization) to be carried out in the ontology development project.

The main goal of the experiment was to test that the students (1) scheduled their ontology development project with gOntt, (2) performed the ontology development project following the schedule created with gOntt and using the methodological guidelines provided by gOntt, and (3) used the necessary NeOn Toolkit plug-ins for every process and activity scheduled. The plug-ins could be triggered by gOntt in the right moment of the schedule.

At the end of the experiment, the students had to fill a questionnaire. Such a questionnaire was divided in two main parts: (1) one related to background knowledge, and (2) another related to the experiences gained by using gOntt. The second part included 10 questions with fixed answers and an open question in which they were asked to express other comments.

In order to analyze the students’ perception of gOntt, we selected three questions from the questionnaire. The answers are displayed in Table 3.

Table 3. Answers to the questions selected

Question	Distribution of answers			
	Yes	No	Other	
Is gOntt useful for scheduling ontology development projects?	14	1	1	
Question	Distribution of answers			
	A useful idea?	A nice utility?	A useless idea?	I don't know
Is the idea of having methodological guidelines in gOntt:	10	6	0	0
Is the idea of triggering NeOn Toolkit plug-ins from gOntt:	6	4	0	6

<sup>3</sup> [http://neon-toolkit.org/wiki/Neon\\_Plugins](http://neon-toolkit.org/wiki/Neon_Plugins)

Analyzing the answers in Table 3, we can say that

- The students perceived gOntt as a useful scheduling tool.
- The students in general perceived as a positive and useful issue the fact that gOntt provided them with methodological guidelines for each process and activity involved in the ontology development.
- Regarding the functionality of triggering NeOn Toolkit plug-ins, their opinion was divided, but in general, they tended to consider it as a good idea.

Additionally, the students provided general comments that can be summarized as follows:

- gOntt is considered a good scheduling tool that is based on activities and processes of the ontology engineering field.
- gOntt is considered a good help in providing a preliminary schedule for a development project.
- gOntt is considered one of the easiest tool for creating Gantt charts, thanks to its simple wizard.
- gOntt is seen as a centralized environment that allows the user to obtain methodological and technological guides without searching in external locations.

The students also commented that the tool established at random the length of the processes and activities planned and not according to previous experiences. They also missed the functionality of exporting the schedule to Microsoft Project.

## 7. Conclusions and future work

This paper outlines the methodological basis for scheduling ontology development projects and explains the technological infrastructure, gOntt, which supports the automatic generation of the initial ontology development plan using a Gantt chart.

gOntt, which is integrated within the NeOn Toolkit, is the first tool that has been created for systematizing the planning and scheduling of ontology projects. It uses templates and decision trees to create initial ontology development project schedules in the form of Gantt charts. In addition, gOntt is the first meta-tool that helps ontology developers to execute ontology development projects. It includes prescriptive methodological guidelines and informs about the recommended tools to use when performing a process or activity in an ontology development.

In short, the most important difference between gOntt and Microsoft Project is that while in gOntt the system generates automatically the initial ontology development plan, in Microsoft Project the user selects a particular project template from the library. Furthermore, gOntt is the only planning tool for

creating and managing project schedules that provides developers with (a) the guidelines to how to execute the project and (b) the recommended tools to be used during the project execution.

The work described in this paper has important implications for our further research for two main reasons. The first one, as soon as the tool is used by the community, we will conduct additional experiments. The second, we plan (a) to integrate gOntt with the works carried out by the ONTOCOM team for predicting the total costs of the ontology development project, and (b) to propose how long processes and activities should take by considering past and present experiences within ontology development.

**Acknowledgments.** This work has been partially supported by the projects NeOn (FP6-027595) and *BuscaMedia* (CENIT 2009-1026).

## 8. References

- [1] Taylor, J.C.: *Project Scheduling and Cost Control: Planning, Monitoring and Controlling the Baseline*. J. Ross Publishing, 2008, ISBN: 9781932159110
- [2] Stellman, A., Greene, J.: *Applied Software Project Management*. 2005. ISBN: 0-596-00948-8
- [3] Gómez-Pérez, A., Fernández-López, M., Corcho, O.: *Ontological Engineering*. Springer Verlag. *Advanced Information and Knowledge Processing series*. ISBN 1-85233-551-3. November 2003
- [4] Staab, S., Hans, P., Studer, R., Sure, Y.: *Knowledge Processes and Ontologies*. *IEEE Intelligent Systems* 16(1): 26–34. (2001)
- [5] Pinto, H.S., Tempich, C., Staab, S.: *DILIGENT: Towards a fine-grained methodology for DIStributed, Loosely-controlled and evolInG Engineering of oNTologies*. 16th European Conference on Artificial Intelligence (ECAI 2004), pp. 393-397
- [6] Simperl, E., Popov, I., Bürger, T.: *ONTOCOM Revisited: Towards Accurate Cost Predictions for Ontology Development Projects*. In: *Proceedings of the European Semantic Web Conference 2009 (ESWC '09)*
- [7] Suárez-Figueroa, M.C., Gómez-Pérez, A.: *First Attempt towards a Standard Glossary of Ontology Engineering Terminology*. 8th International Conference on Terminology and KE (TKE 2008). ISBN: 87-91242-50-9. Pp 1-15. Copenhagen, Denmark. August 2008
- [8] Larman, C.: *Applying UML and patterns: an introduction to object-oriented analysis and design and the unified process*. Second edition. Pearson Education, Inc. 2002. ISBN: 0-13-092569-1
- [9] Suárez-Figueroa, M.C., Gómez-Pérez, A.: *NeOn Methodology for Building Ontology Networks: a Scenario-based Methodology*. In *Proceedings of the International Conference on Software, Services and Semantic Technologies (S3T 2009)*. ISBN: 978-954-9526-62-2. Pp: 160-167. Sofia, Bulgaria. October 2009

# A Project Monitoring Cockpit Based On Integrating Data Sources in Open Source Software Development

Stefan Biffl, Wikan Damar Sunindyo and Thomas Moser  
Institute of Software Technology and Interactive Systems  
Vienna University of Technology  
Vienna, Austria  
{biffl, wikan, tmoser}@ifs.tuwien.ac.at

*Abstract* — Many open source software (OSS) development projects use tools and models that come from heterogeneous sources. A project manager, who wants to analyze indicators for the state of the project based on these data sources, faces the challenge of how to link semi-structured information on common concepts across heterogeneous data sources, e.g., source code versions, mailing list entries, and bug reports. Unfortunately, manual analysis is costly, error-prone, and often yields results late for decision making. In this paper we propose linking OSS data sources using semantic web technologies as foundation for providing integrated indicators project status analysis. We introduce the design concept of a project monitoring cockpit, ProMonCo, and evaluate the feasibility and effectiveness with a prototype for calculating communication metrics in a real-world context, the Apache Tomcat project. Major result was that ProMonCo efficiently supports frequent project monitoring by calculating communication metrics based on semantically integrated data originating from heterogeneous OSS project data sources.

*Keywords* - open source software development; communication metrics; semantic integration

## I. INTRODUCTION

Stakeholders of open source software (OSS) development projects routinely use a wide range of tools and models for developing and managing software products [1]. For example, developers may use SVN<sup>1</sup> for managing source code versions, mailing lists for the communication between developers, and bug reports for tracking software defects. In OSS development projects, with stakeholders who typically work in dispersed locations and time zones, project managers need to be able to manage and monitor the status of the project based on relevant indicators, e.g., by monitoring the communication level between the stakeholders. In healthy projects, the number of communication level between the stakeholders tends to be proportional, while the challenged projects show fluctuations which signify the imbalance of the number of communication level between the stakeholders [2].

Currently, project managers use quantitative measurement approaches for managing and monitoring the status of the projects [3], e.g., the number of commits to the SVN, the number of mailing list posts, or the number of issues in a bug report

tool in a certain period. Further analyses to find related information between data sources like the correlation between the email conversation, bug status changes and SVN/CVS commits with the health status of the projects, need to be performed manually, which is costly, error-prone and often takes too much time for the analysis results to be useful for the project manager's decision making.

Communication metrics [4] can provide the foundation for relevant analyses and insights into software development processes, like finding the relationship between the project managers position and their communication forms with other stakeholders in the network. In comparison to more traditional software metrics based on software products, metrics generated from communication artifacts are fairly simple to compute and available early on in the software lifecycle begins. Insights from using communication metrics could not be achieved by traditional metrics, since they are based on different artifacts generated during a project that are not suited for indicating the quality of communication between team members. In addition, the free and widely available communication archives of OSS projects present a rich source for applying communication metrics.

Tools and models in OSS development projects typically come from heterogeneous sources. While the project stakeholders informally can agree on common concepts (e.g., project artifacts or projects issues) in the project, the representation of these common concepts usually varies across tools regarding the structure and terminologies (e.g., code and documentation artifacts, or bugs and project tasks) used. Therefore, expert know-how is needed to identify semantically equivalent concepts that look syntactically different. A project manager, who wants to analyze the state of the project based on information from these project data sources, faces the challenge of how to link semi-structured information (i.e., information that is available in a format both readable for humans and machines) on common concepts across heterogeneous data sources, e.g., source code versions, mailing list entries, and bug reports. While domain experts can conduct the linking manually, this manual analysis is typically costly and error-prone. Therefore, an integrated view of relevant project data is often not available to the project manager to support making relevant decisions, such as planning releases of new software product versions or restructuring the developer team.

---

<sup>1</sup> <http://subversion.tigris.org>

In this paper, we introduce a project monitoring cockpit, *ProMonCo*, which uses semantic integration approaches for bridging semantic gaps between heterogeneous project data sources as foundation for comprehensive automated analyses on the integrated project data model. We build on semantic web technology, the Engineering Knowledge Base (EKB) semantic integration framework [5], to explicitly link the data model elements of several heterogeneous OSS project data sources based on their data semantic definitions. The EKB consists of two types of ontology layers: the common domain knowledge layer and the local tool knowledge layer. To bridge the semantic gaps (different representations for equivalent semantic concepts) between these ontology layers, there are mappings between the local and domain ontologies that can query local knowledge using common domain knowledge syntax.

We propose a design concept for the project monitoring cockpit *ProMonCo* and implement a prototype to demonstrate how to calculate a set of communication metrics for the project manager. We evaluate the feasibility and effectiveness of the *ProMonCo* prototype by calculating three kinds of communication metrics based on real-world OSS development data originating from heterogeneous data sources in the Apache Tomcat<sup>2</sup> and other Apache projects. Major result is that *ProMoCo* efficiently supports frequent project monitoring by calculating communication metrics based on semantically integrated data originating from heterogeneous OSS project data sources.

This remainder of this paper is structured as follows: Section 2 summarizes related work on project monitoring and communication metrics, on the use of semantic web technologies for software engineering, and on OSS development. Section 3 identifies the research issues, while section 4 introduces the *ProMonCo* prototype, describes its architecture and the implemented communication metrics. Section 5 reports on the evaluation of *ProMonCo* in the context of the Apache Tomcat and other Apache projects based on the implemented communication metrics, and additionally discusses the findings. Finally, section 6 concludes the paper and identifies further work.

## II. RELATED WORK

This paper is based on previous work on project monitoring and reporting for OSS development projects. Manual reporting is a conventional approach that seems to fit well to tightly coupled organizations but is hard to apply for loosely coupled organizations like OSS development projects [6]. In this section, we summarize related work on project monitoring and communication metrics, on the use of semantic web technologies for software engineering, and on OSS development.

### A. Open Source Software Development

OSS development is a knowledge-intensive domain that could be improved by using semantic web technologies. Sharma et al. [1] state that OSS development models are considered to be successful if they meet the original requirements regarding software quality. Even though some OSS projects do not finish successfully, in general they can be used as a model for

proprietary software development as well. Sharma et al. [1] also introduced a framework for creating and maintaining hybrid-open source software communities, which consist of structures, processes and cultures. The wide range of stakeholders involved the different types of processes to be performed and cultural heterogeneities can lead to semantic gaps between the stakeholders, which need to be bridged for comprehensive tool support of OSS projects, e.g., by using semantic web technologies [7].

Mockus et al. performed an analysis on two OSS projects, namely the Apache Web Server and the Mozilla browser [8; 9]. Their goal was to quantify the aspects of developer participation, core team size, code ownership, productivity, defect density, and problem resolution for OSS projects. They collected and analyzed the data from different sources, for example developers' mailing list, concurrent version archive (CVS) and problem reporting database (BUGDB). However, they focused on the use of non-integrated historical artifacts originating from different data sources to measure the quality of software. In this paper, we integrate both historical and current artifacts originating from different heterogeneous project data sources to get more information (e.g., current project status, times needed for bug fixing, etc.) for decision making.

### B. Project Monitoring and Communication Metrics

Von Krogh and von Hippel [3] investigate OSS development processes and found differences between monitoring commercial software development and OSS development. In commercial software development, the project manager can apply tight management of processes and take precautions to prevent employees from leaking software-related trade secrets and information to competitors, while in OSS development software architecture and functionality are governed by a community consisting of developers who can commit code to the authorized version of the software.

Yamauchi et al. [10] state that in a traditional perspective, managing and leading OSS development projects seems to be impossible, because no formal quality control program exists and no authoritative leaders monitor the development. For them it is surprising that the OSS development can achieve smooth coordination, consistency in design and continuous innovation while relying heavily on electronic environment as face-to-face supplementary, however, project monitoring for OSS projects is still in a very early phase. In addition, they discuss how OSS development avoids limitations of dispersed collaboration and addresses the sources of innovation in OSS development. Further research is needed to reveal how typical project management methodologies can be adapted to the OSS domain in order to improve the software quality, e.g., by monitoring typical OSS project product and process data.

Wahyudin et al. [6] discuss that project monitoring traditionally focused on human-based reporting, which is good for tightly coupled organizations to ensure the quality of project reporting. In loosely-coupled organizations like in OSS development projects, this approach does not work well because the stakeholders typically work voluntarily and flexibly. One way to measure the performance of the project is by correlating and

---

<sup>2</sup> <http://tomcat.apache.org>

analyzing process event data (e.g., mailing list artifacts or bug reports) from the OSS community.

Sharma et al. [1] observe the OSS development projects based on three aspects: namely structure, processes, and culture. In OSS processes, stakeholders can have governance mechanisms, for example by applying membership management, rules and institutions, monitoring and sanctions, and reputation as one of the prime motivators for the OSS developers. Even though membership in OSS projects is open to anyone, the OSS communities manage membership in conjunction with rules, institutions, monitoring, and sanctions. They illustrate how OSS projects can be monitored via social interaction and sanctions from the communities. However, the relationships between the project data produced by the stakeholders, the activities of the stakeholders, and the quality measurement of OSS were not analyzed in their study. We build on their research by performing social network analyses using heterogeneous data sources to monitor the project communication and project status of OSS projects.

Communication metrics [4] can provide the foundation for relevant analyses and insights into software development processes and in addition, the free and widely available communication archives of OSS projects present a rich source for applying such communication metrics. To our knowledge there is only very little research on communication metrics. Dutoit and Bruegge [4] found initial empirical evidence that metrics based on communication artifacts can generate significant insight into the status of an application development process in the context of commercial software development. Communication metrics may even provide better insights (e.g., project status) in the development processes than code-based metrics, since they focus more on process than on product specific data. Current general practice is to measure and analyze the software code. However, this practice ignores that software code is only available late in the development process and not the only artifact generated. Communication artifacts, such as e-mails, mailing list entries, memo notes, or records generated by groupware tools are valuable information that are available early and can be used to investigate the health of the development process.

### C. Semantic Web Technologies for Software Engineering

Semantic integration is defined as the solving of problems originating from the intent to share data across disparate and semantically heterogeneous data [11]. These problems include the matching of ontologies or schemas, the detection of duplicate entries, the reconciliation of inconsistencies, and the modeling of complex relations in different data sources [12]. One of the most important and most actively studied problems in semantic integration is establishing semantic correspondences (also called mappings) between vocabularies of different data sources [13].

The application of ontologies as semantic web technologies for managing knowledge in specific domains is inevitable. Noy and Guinness [14] note five reasons to develop an ontology, namely to share common understanding of the structure of information among people or software agents, to enable reuse of domain knowledge, to make domain assumptions explicit, to

separate domain knowledge from the operational knowledge, and to analyze domain knowledge.

Happel and Seedorf [15] summarized the use of ontology-based approaches in different phases of the software engineering lifecycle. They compared research advances in software engineering and knowledge engineering, and found that the software engineering community has been focusing on software modeling, while the knowledge engineering community has been eager to promote several modeling approaches to realize the vision of the semantic web. Hence, both disciplines are closely related and the communities from both areas can contribute to each other. The paper categorized ontologies in software engineering into four areas based on time dimension (development time vs. run time) and types of knowledge (infrastructure vs. software), namely ontology-driven development, ontology-enabled development, ontology-based architectures, and ontology-ruled architecture. The authors saw the main advantage of ontologies in software engineering, among others, in the ability to define and use logic-based formalisms for software engineering in the context of the semantic web efforts. The flexibility of ontologies can make the combination of software engineering data from heterogeneous sources easier and more effective. While this paper is an important first step towards a better understanding of possible benefits of ontologies in software engineering, implementation issues remain open.

Moser et al. [5] introduced the Engineering Knowledge Base (EKB) framework as a semantic web technology approach for addressing challenges coming from data heterogeneity and applicable for different domains, in [5] for the production automation domain. Further, Biffl et al. [4] used the approach for solving similar problems in the context of OSS projects, in particular, frequent-release software projects. We build on this previous work by using and extending the proposed project data fetcher tool for extracting and integrating OSS project data from heterogeneous data repositories.

Other uses of ontologies in software engineering, e.g., for supporting software architecture decisions or for reuse of software development knowledge, have been suggested by Akerman and Tyree [1] and by Antunes et al. [2]. Since these decisions and reuse possibilities are also interesting for OSS development projects, the investigation of the applicability of ontologies for OSS development projects regarding these issues is also important.

## III. RESEARCH ISSUES

In this paper, we focus on the aim of the project managers to improve the efficiency of project reporting and monitoring. The questions which can be raised are what the current situation of project reporting and monitoring in the OSS domain is, and how its efficiency can be improved. For answering these questions, the project manager needs to be able to collect communication data from development project and perform a set of communication metrics for further analysis purposes, like project reporting (e.g., average bug response time), project monitoring (e.g., monitoring the current project status or activities) or decision making (e.g., planning releases of new software product versions) on the OSS development. In compari-

son to more traditional software metrics based on software products, metrics generated from communication artifacts are fairly simple to compute and available early on in the software lifecycle begins. In addition, the free and widely available communication archives of OSS projects present a rich source for applying communication metrics.

Collecting related information from heterogeneous data sources for project reporting and monitoring is not an easy task for project managers, because each data source typically has its own proprietary formats and data structures. Semantic web technologies can help to solve problems originating from semantic heterogeneities among different data sources, by supporting the knowledge representation and storage for all project development data. The utilization of rule and reasoning can help the user to check the correctness of related data originating from heterogeneous data sources and find the relationships (e.g., synonyms, generality-specialty) between the data. From these challenges we derived the following research issues.

**RI-1. Feasibility and effectiveness of tool support for project monitoring and reporting based on heterogeneous data sources in an OSS project context.** On the foundation of general semantic web technology we investigate the design for tools that collect and semantically integrate data from OSS sources in order to build analysis applications suitable for project monitoring and reporting. In addition, we evaluate the feasibility to build a project monitoring cockpit *ProMonCo* as example for such an application. Finally, we evaluate the effectiveness of the *ProMonCo* tool to automate relevant aspects of project monitoring and reporting.

**RI-2. Automated derivation of project health indicators from communication metrics from OSS project data sources.** As application for the *ProMonCo* tool we investigate whether relevant project indicators, such as health indicators based on communication metrics like the *user coupling* metric and the *bug history* metric can be correctly and efficiently derived from heterogeneous data sources in OSS projects.

*User Coupling Metric (UCM).* The UCM is supposed to give significant insight into the communication structure of a project team [1]. Team members with significant importance for the will become very centrally located nodes. Project members that do not participate at the intra-team communication intensively will result in less central nodes. Very incommunicative behavior may even result in sub-graphs containing only a few nodes. After building the social network graph out of the available communication artifacts, several centrality measurements can be conducted. These measurements should indicate the importance of individual project members for the existing communication network.

*Bug History Metric (BHM).* The BHM will contrast the communication effort of the project members with the bug activities during a certain period of time. Further evaluation will show, if an increasing number of found or resolved bugs results in a growing amount of communication. If a correlation exists, project managers may consider supporting and intensifying the communication effort. This may e.g. be achieved by emphasizing team meetings.

To answer these research issues we gather requirements from interviews with OSS project experts and develop a prototype of the *ProMonCo* project monitoring cockpit. We then perform initial evaluations and data analyses using the data of the OSS Apache Tomcat project and other OSS Apache projects as a test-bed. We choose the Apache Tomcat project, because it has a long story of development, is quite stable, exists in several versions, and therefore provides wide variety of development data sources for analysis.

#### IV. THE *PROMONCO* PROJECT MONITORING COCKPIT

This section describes the implemented Project Monitoring Cockpit prototype, called *ProMonCo*. First, the Project Data Fetcher Tool, used to collect and integrate OSS development project data, is described shortly. Then, *ProMonCo* is introduced and its architecture is explained. Additionally, the implemented communication metrics of *ProMonCo* are described.

##### A. *Project Data Fetcher*

Considering the requirement of semantically integrating data originating from heterogeneous OSS project data sources, a tool for the extraction of project data for Apache projects called Project Data Fetcher was initially developed [16]. This tool allows the gathering of project artifacts from the mailing list, the Bugzilla<sup>3</sup> database and the Subversion versioning system of Apache projects. The retrieved data is the basis of the evaluation of the designed communication metrics of this thesis. The Project Data Fetcher uses an ontology for the storage of the extracted project data. In contrast to a conventional database, an ontology is capable of a proper knowledge representation based on well-defined semantics. In addition, an ontology provides reasoning capabilities that may be utilized to discover previously hidden information by deducting new facts out of existing ones.

##### B. *Project Monitoring Cockpit ProMonCo*

The Project Monitoring Cockpit *ProMonCo* is an application that can be used for monitoring the status of project based on communication artifacts. By using *ProMonCo*, project manager can trigger the fetching of project data and calculate implemented communication metrics based on the fetched artifacts. As mentioned previously, *ProMonCo* uses the information gathered and integrated by the Project Data Fetcher. *ProMonCo* uses the generated ontologies for extracting the relevant information, generating communication metrics out of it and visualizing the outcome. For this purpose, the project manager has to load previously generated ontologies. All available communication metrics can be calculated automatically out of the data stored in the ontology, according to the settings selected by the user. Figure 1 gives an overview of the architecture of *ProMonCo*.

The architecture of *ProMonCo* consists of three layers, namely Input Handler, Communication Metrics Analyzer and Output Handler. In the Input Handler layer, the ontology input manager uses previously generated project ontologies as an input for *ProMonCo*. These ontologies will be processed and

---

<sup>3</sup> <http://www.bugzilla.org>

analyzed according to the different purposes and criteria of the communication metrics. The total communication metric uses the number of mail posted from the ontology in certain period and shows the visualization graphics depend on the time granularity and the chart style. The user coupling metric uses the interaction between developers by using communication means and shows the social network graph in the form of communication graph, betweenness centrality, closeness centrality and degree centrality. The bug history metric uses bug data from the ontology and shows bug activities diagram and capability to export file for further statistical analysis purposes.

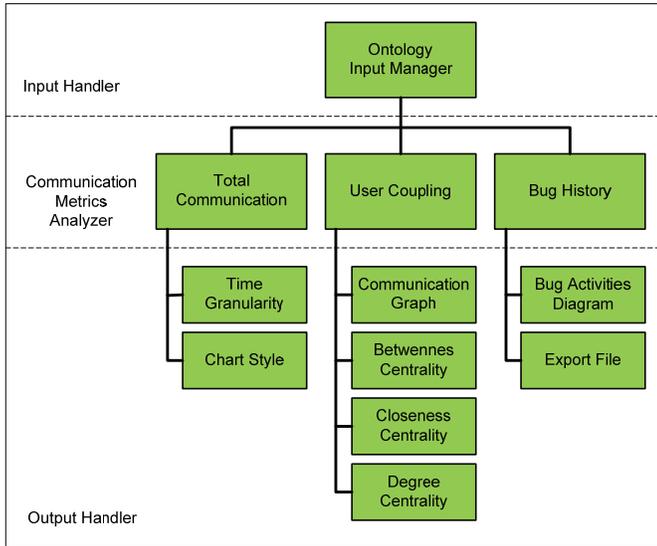


Figure 1. Overview *ProMonCo* architecture.

### C. Communication Metrics Implementation

This section shortly summarizes the three types of implemented communication metrics, namely *Total Communication*, *User Coupling* and *Bug History*.

**Total Communication Metric (TCM).** The TCM is calculated by counting the number of communication artifacts during a certain period of time. The user can change the graphical representation and the calculated time period by changing the appropriate settings. The TCM is comparable to the traditional Lines of Code (LoC) metric, and serves as basis for more advanced metrics.

**User Coupling Metric (UCM).** The basic idea of the UCM is to create a social network graph out of the available mailing list information and hence acquire important insights in the communication behavior of the development team. The social network graph is supposed to reveal how deep each team member is involved in the overall communication. Each member of the development team is displayed by a vertex within the graph while the communication between members will be represented by an edge. Alongside the graph, several centrality measurements are calculated to determine numerically which members play a central role in the inner team communication. The first difficulty in the design of the UCM consists of the decision on how to build the edges displaying communication between team members. One possibility consists of building the edges between the vertexes when a mailing list user an-

swers to a message of another user. However, the answers within a topic of the mailing list often do not only correspond to one message of another user, but rather to the topic in general. Therefore, all users participating in a topic are considered to be communicating with each other and will consequently be connected by edges in the communication graph.

**Bug History Metric (BHM).** The BHM is based on the TCM and aims at making the communication effort and bug activities comparable. The measurement compares the number of communication artifacts generated within a specified period of time with the amount of bug tracking entries. This comparison allows identifying correlations between these two activities and therefore provides vital information for project managers. If a high number of found or resolved defects results in a high communication effort, it is important to provide the project members with sufficient possibilities to communicate with each other, for example by planning more team meetings. The output of the BHM is a diagram that compares the communication effort with the corresponding bug activities for each time interval. To make these datasets visually comparable, the two plots are overlaid. This allows the user to observe the shape of the curves and to recognize if a correlation of some sort occurs. Additionally, further statistical tests can be conducted by using the export feature of *ProMonCo* to save all calculated results as CSV (comma-separated values) file.

## V. EVALUATION AND DISCUSSION

In this section, we discuss the evaluation of the project monitoring approach and the project monitoring cockpit *ProMonCo* regarding the research issues introduced in section 3.

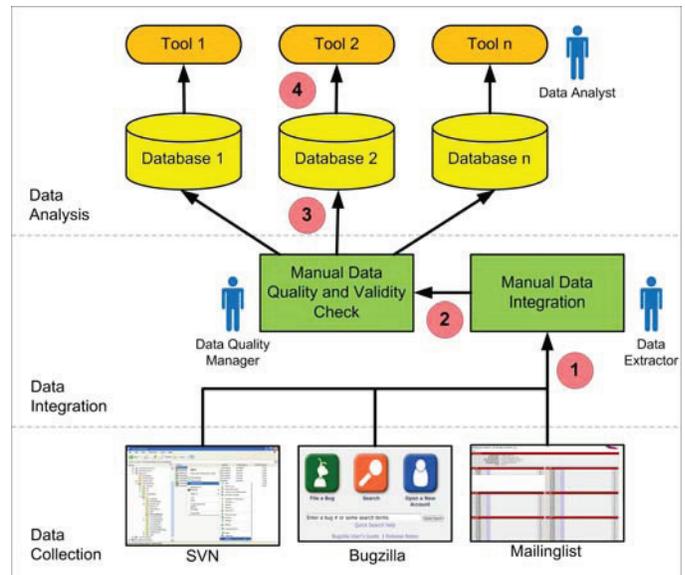


Figure 2. Manual Project Monitoring and Reporting.

**Feasibility and effectiveness of tool support for project monitoring and reporting based on heterogeneous data sources in an OSS project context.** Manual project monitoring and reporting has been used so far for commercial software development projects based on human reporting and structural organization [6]. However, this approach is not suitable for

loosely coupled forms of organizations like OSS projects that cannot rely on human reporting because of geographical distribution and primarily voluntary work of the project participants. Figure 2 shows the process of manual project monitoring and reporting and identifies limitations of this approach.

Heterogeneous data sources, such as SVN artifacts, Bugzilla data or mailing list entries, need to be manually integrated by a data extractor. The data extractor needs to identify the structure and the content of each data source, as well as their mutual relationships during the data integration process, but since the collected raw data may have incompatible formats, the data integration process can become inefficient and error prone (1). Current software quality monitoring approaches such as software defect prediction focus mainly on measuring software quality by means of static code metrics (2), such as code complexity, size and volume [17], which often fails to capture important distributed development process data (e.g., maturity level of code peer review prior to release), and in addition lacks of capability in providing early warning of certain quality status or risks due to the late data availability (i.e., too close to system deployment date). Furthermore, many studies treat all defects in similar way with no respect to defect severity level [18]. The results of this data integration need to be checked by a quality manager in order to guarantee the correctness, quality and validity of the integrated data before storing it in the database. There may exist several different database systems to support data analyses using different analysis tools (3). And finally, Project Monitoring typically is performed by reviewing the status and results of each analysis tool (4).

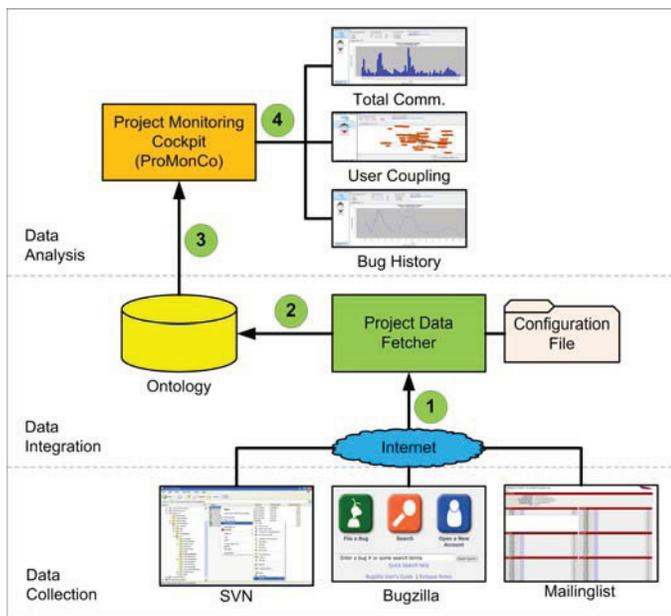


Figure 3. Project Monitoring and Reporting using *ProMonCo*.

Figure 3 shows the project monitoring and reporting process when using *ProMonCo* and identifies benefits of the approach. The Project Data Fetcher will fetches the data from the particular data sources via the web, semantically integrates the data based on a set of heuristics and then stores the integrated data in the project ontology [16], thus enabling a more efficient collection and integration of OSS project data (1) and

putting together all required information with their relationships and rules into the project ontology for further analysis (2). The project ontology is then used as input for *ProMonCo* (3) to calculate several communication metrics used for monitoring and reporting of a project’s status (4).

**Automated derivation of project health indicators from communication metrics from OSS project data sources.** *ProMonCo* provides three types of communication metrics to show the status of analyzed OSS projects: namely the *Total Communication* (TCM), the *User Coupling* (UCM), and the *Bug History Metric* (BHM).

The evaluation of the TCM shows that the amount of communication artifacts depends on the size of the project team. Projects with few participants produce evidently less mailing list entries than projects with a higher amount of developers. *ProMonCo* offers the possibility to compute the TCM for three different time intervals: yearly, monthly and weekly. The most meaningful time interval is the monthly view. On one hand, computing the metric for every week produces measurements with a very high standard deviation, making it hard to interpret the results. On the other hand, generating the metric on a yearly basis does not produce comprehensive information about the communication effort of the development team. Hence, calculating the metric for every month gives the project manager the best basis for observing the communication effort over time. Figure 4 illustrates a diagram of the TCM calculated monthly for six Apache projects, namely Tomcat, Ant, Cocoon, Lenya, Log4j and POI.

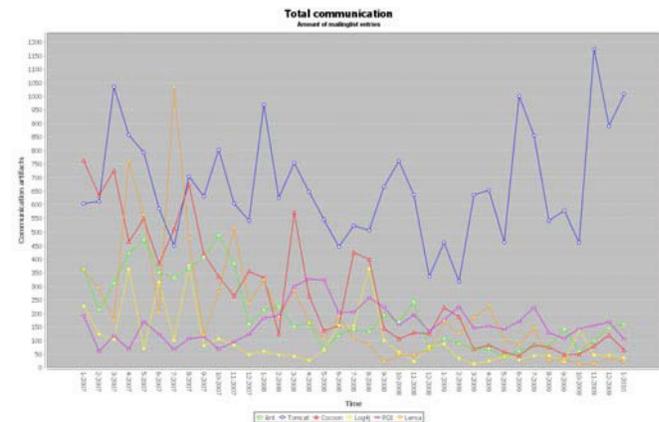


Figure 4. *ProMonCo* Total Communication Metric.

The UCM is based on principles and methods of social network analysis. It is used to generate a graph that visualizes the flow of communication between team members of a software project and to calculate the centrality measurements for each participant. The Apache Tomcat project was examined in regards of the communication behavior of the team members. For this purpose, the project data of the past three years was gathered using the Project Data Fetcher tool. The generation of the communication graph reveals that only very few sub graphs exist. These sub graphs only contain a very small amount of mailing list participants that are no core members of the project and do not commit source code to the project repository. According to the communication graph, all core project members are connected and communicating with each other. The graph

shown in Figure 5 represents the communication behavior of the Apache Tomcat project members that was calculated for the time period between November 2009 and February 2010.

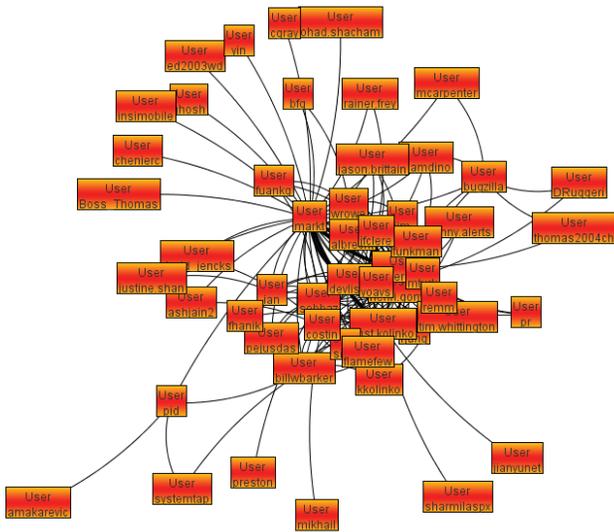


Figure 5. Communication Graph of the Apache Tomcat Project.

A closer look at the centrality measurements reveals that two persons have outstanding values compared to the rest of the development teams: Mark Thomas and Rainer Jung. Further research revealed that these two project members do not only hold a very central position within the communication network, but are also involved in the project management. Although both members lie very central, the point betweenness centrality value for Mark Thomas is extraordinarily high. An actor within a social network that has a high betweenness centrality value is supposed to have a significant influence on the flow of information. Individuals in such central positions have the possibility to coordinate the processes of the group and should therefore be part of the project management [19]. Hence, the high betweenness centrality value for Mark Thomas indicates that he possesses a high potential in influencing and coordinating the rest of the development team.

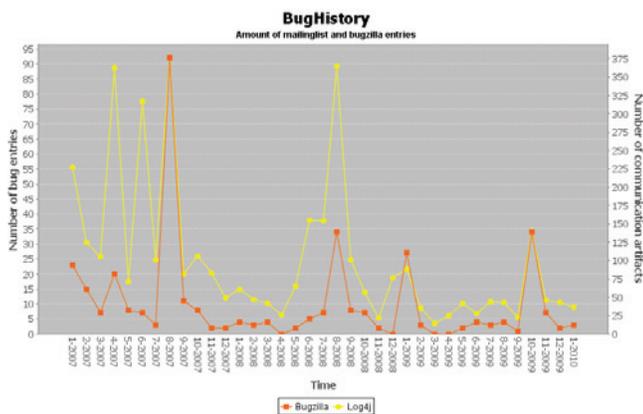


Figure 6. Resolved bugs and mailing list entries of the Apache Log4j Project.

The BHM compares the amount of bug tracking activities with the communication effort within a certain period of time. It

reveals a possible correlation between Bugzilla artifacts and the amount of mailing list entries generated by the development team. *ProMonCo* calculates the metric for new found bugs and for resolved bugs separately. Hence, project managers are able to see if the search and discovery of bugs or the correction of defects results in an increased amount of communication. An obvious correlation between Bugzilla and mailing list artifacts may be recognized visually. Figure 6 shows a line chart of the resolved bugs and the amount of mailing list entries between January 2007 and January 2010 for the Apache Log4j<sup>4</sup> project. For the evaluation of the BHM, the results of the Apache projects were investigated with SPSS.

For testing the possible correlation between bug tracking artifacts and mailing list entries, the two-tailed Pearson correlation is used. The evaluation of the results shows a significant correlation between Bugzilla entries and the number mailing list messages in nearly all projects. The correlation occurs regardless of comparing the number of new found or resolved defects with the amount of communication artifacts. The only exception is the Apache Ant project which does not show a significant correlation between resolved bugs and mailing list usage.

Table 1 contains the results of the correlation testing for new found bugs, whereas Table 2 shows the results for resolved bugs.

Table 1. Pearson correlation for new found bugs.

	n	Pearson Correlation	Significance (two-tailed)
Ant	37	0.704*	0.000
Lenya	37	0.885*	0.000
Log4j	37	0.696*	0.000
POI	37	0.706*	0.000
Tomcat	37	0.689*	0.000

\*Correlation is significant at the 0.01 level (two-tailed).

Table 2. Pearson correlation for resolved bugs.

	n	Pearson Correlation	Significance (two-tailed)
Ant	37	0.018	0.916
Lenya	37	0.808*	0.000
Log4j	37	0.706*	0.000
POI	37	0.746*	0.000
Tomcat	37	0.438*	0.007

\*Correlation is significant at the 0.01 level (two-tailed).

The significant correlation between the amount of Bugzilla and mailing list artifacts may be explained by the excessive use of the mailing list for the discussion of topics regarding defects. In addition, all comments a team member creates in Bugzilla concerning a bug are automatically forwarded to the mailing list. A high number of new or resolved bugs usually result in a

<sup>4</sup> <http://logging.apache.org/log4j>

high number of Bugzilla comments that are sent to the mailing list. Therefore, considering different sources of communication artifacts may result in a different outcome of the BHM.

## VI. DISCUSSION AND CONCLUSION

Many OSS development projects use tools and models that come from heterogeneous sources. A project manager, who wants to analyze indicators for the state of the project based on these data sources, faces the challenge of how to link semi-structured information on common concepts across heterogeneous data sources, e.g., source code versions, mailing list entries, and bug reports. Unfortunately, manual analysis is costly, error-prone, and often yields results late for decision making.

In this paper we proposed linking OSS data sources using semantic web technologies as foundation for providing integrated indicators project status analysis. We introduced the design concept of a project monitoring cockpit, *ProMonCo*, and evaluated the feasibility and effectiveness of *ProMonCo* with a prototype for calculating communication metrics in a real-world context, the Apache Tomcat project as well as other Apache projects.

Major result was that *ProMonCo* efficiently supports frequent project monitoring by calculating communication metrics based on semantically integrated data originating from heterogeneous OSS project data sources. The usage of the Project Data Fetcher to collect and semantically integrate the data originating from the different heterogeneous OSS project data sources allows for more efficient project monitoring and reporting. *ProMonCo* builds up on this integrated project ontology and can be used to efficiently and effectively calculate a set of communication metrics which can be useful for monitoring the project status and for reporting to project or quality managers. Communication metrics like the User Coupling Metric (UCM) or the Bug History Metric (BHM) can reveal new insights on OSS projects. In the evaluation, we used the UCM to identify project members of the OSS Apache Tomcat project which are also involved in the project management, based on centrality measurements of the social network communication graph resulting from the UCM. Furthermore, using the BHM, we identified a significant correlation between Bugzilla entries and the number mailing list messages in nearly all evaluated Apache projects. The correlation occurs regardless of comparing the number of new found or resolved defects with the amount of communication artifacts.

Future Work will include the integration of additional project data sources, such as different sources of communication artifacts, in order to be able to perform new types of communication metrics, as well as also other process metrics. Furthermore, we plan to expand our approach to other OSS project types, such as SourceForge<sup>5</sup> projects.

## ACKNOWLEDGMENT

The authors want to thank Raphael Zaki and Stefan Huber for implementing prototypes of the Project Data Fetcher and of *ProMonCo*. This work has been supported by the Christian

Doppler Forschungsgesellschaft and the BMWFJ, Austria. In addition, this work has been partially funded by the Vienna University of Technology, in the Complex Systems Design & Engineering Lab.

## REFERENCES

- [1] S. Sharma, V. Sugumaran and B. Rajagopalan, "A framework for creating hybrid-open source software communities," *Information Systems Journal*. City, vol. 12, pp. 7-25, 2002.
- [2] D. Wahyudin, K. Mustofa, A. Schatten, S. Biffl and A. M. Tjoa, "Monitoring the "health" status of open source web-engineering projects," *International Journal of Web Information Systems*. City, vol. 3, pp. 116-139, 2007.
- [3] G. von Krogh and E. von Hippel, "Special issue on open source software development," *Research Policy*. City, vol. 32, pp. 1149-1157, 2003.
- [4] A. H. Dutoit and B. Bruegge, "Communication metrics for software development," *IEEE Transactions on Software Engineering*. City, vol. 24, pp. 615-628, 1998.
- [5] T. Moser, S. Biffl, W. D. Sunindyo and D. Winkler, "Integrating Production Automation Expert Knowledge Across Engineering Stakeholder Domains," *Proc. 4th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS 2010)*, IEEE Computer Society, 2010, pp. 352-359.
- [6] D. Wahyudin and A. M. Tjoa, "Event-Based Monitoring of Open Source Software Projects," *Proc. Proceedings of the The Second International Conference on Availability, Reliability and Security*, IEEE Computer Society, 2007, pp. 1108-1115.
- [7] S. Biffl, W. D. Sunindyo and T. Moser, "Bridging Semantic Gaps Between Stakeholders in the Production Automation Domain with Ontology Areas," *Proc. 21st International Conference on Software Engineering and Knowledge Engineering*, 2009, pp. 233-239.
- [8] A. Mockus, R. T. Fielding and J. Herbsleb, "A case study of open source software development: the Apache server," *Proc. 22nd Intl Conference on Software engineering*, ACM, 2000, pp. 263-272.
- [9] A. Mockus, R. T. Fielding and J. D. Herbsleb, "Two case studies of open source software development: Apache and Mozilla," *ACM Trans. Softw. Eng. Methodol.* City, vol. 11, pp. 309-346, 2002.
- [10] Y. Yamauchi, M. Yokozawa, T. Shinohara and T. Ishida, "Collaboration with Lean Media: how open-source software succeeds," *Proc. ACM conference on Computer supported cooperative work*, ACM, 2000, pp. 329-338.
- [11] A. Halevy, "Why your data won't mix," *Queue*. City, vol. 3, pp. 50-58, 2005.
- [12] N. F. Noy, A. H. Doan and A. Y. Halevy, "Semantic Integration," *AI Magazine*. City, vol. 26, pp. 7-10, 2005.
- [13] A. Doan, N. F. Noy and A. Y. Halevy, "Introduction to the special issue on semantic integration," *SIGMOD Rec.* City, vol. 33, pp. 11-13, 2004.
- [14] N. F. Noy and D. McGuinness, 2001, *Ontology Development 101: A Guide to Creating Your First Ontology*, Stanford Knowledge Systems Laboratory.
- [15] H.-J. Happel and S. Seedorf, "Applications of Ontologies in Software Engineering," *Proc. 2nd International Workshop on Semantic Web Enabled Software Engineering (SWESE 2006)* held at the 5th International Semantic Web Conference, 2006, pp. 1-14.
- [16] S. Biffl, W. D. Sunindyo and T. Moser, "Semantic Integration of Heterogeneous Data Sources for Monitoring Frequent-Release Software Projects," *Proc. 4th International Conference on Complex, Intelligent and Software Intensive Systems*, IEEE, 2010, pp. 360-367.
- [17] N. Fenton and M. Neil, "A Critique of Software Defect Prediction Models," *IEEE Trans. Softw. Eng.* City, vol. 25, pp. 675-689, 1999.
- [18] S. Biffl, A. Aurum, B. Boehm, H. Erdogmus and P. Gruenbacher, *Value-Based Software Engineering*: Springer Verlag, 2005.
- [19] L. C. Freeman, "Centrality in social networks conceptual clarification," *Social networks*. City, vol. 1, pp. 215-239, 1979.

<sup>5</sup> <http://sourceforge.net>

# Software Configuration Management as a Crosscutting Concern: An Example on Software Testing

Elisa Yumi Nakagawa, João Vítor Tornisiello Trevisan, José Carlos Maldonado  
Dept. of Computer Systems  
University of São Paulo - USP  
PO Box 668, 13560-970, São Carlos, SP, Brazil  
{elisa, jcmaldon}@icmc.usp.br, jtrevisan@grad.icmc.usp.br

## Abstract

*SCM (Software Configuration Management) has substantially contributed as a mature, reliable and essential technology for successful software development, controlling changes and software evolution. In this scenario, tools that automate SCM have been proposed, developed and used in diverse software engineering activities and processes. However, each SCM tool has usually its particular architecture and data structures. Furthermore, there is a lack of work that investigate reuse and evolvability of such tools. In this paper we propose to see SCM as a crosscutting concern, i.e., as an activity that is spread across or tangled with other software engineering activities, in the same perspective of the AOSD (Aspect-Oriented Software Development). Based on this approach, we present a case study to the development of a SCM tool, named ATCMag, for software testing domain. Preliminary results show the viability of our approach and point out to the possibility of developing reusable and evolvable SCM tools.*

## 1. Introduction

Software Configuration Management (SCM) has been consolidated as a mature and reliable technology for the successful software development [11]. It is a supporting management activity that applies technical and administrative direction over the life cycle of a software product [15, 14]. It controls mainly the changes and software system evolution [11]. Regarding SCM application, different software processes have explored and used it. At one extreme, there are agile processes, such as FDD (Feature Driven Development) [18] and open source process (for instance, the CMM-like model for OSS<sup>1</sup>), that have SCM as a very important activity. At the other extreme, SCM

has been required by important capability maturity models, such as CMMI/SEI<sup>2</sup>, and by guidelines and standards, such as ISO/IEC 12207:1995 - Information Technology: Software Life-Cycle Processes [14]. More recently, SCM has also been explored in different perspectives, such as in software product line [1] and service-oriented systems [24]. The success of SCM is directly dependent on a diversity of SCM tools that have been proposed, implemented and successfully used. In spite of this diversity, we have observed that each SCM tool has usually its particular architecture and data structures. Moreover, each one has supported in a specific software engineering activity, such as coding. In this context, reuse and evolvability in SCM tools are two issues that have not been widely discussed and, therefore, worthwhile to be investigated.

In another perspective, AOP (Aspect-Oriented Programming) [16] has arisen as an approach that supports a better SoC (Separation of Concerns) and more adequately reflects the way developers think about the system. Essentially, AOP introduces a unit of modular implementation – the aspect – which has been typically used to encapsulate crosscutting concerns in software systems (i.e., concerns that are spread across or tangled with other concerns). Modularity, maintainability and facility to write software can be achieved with AOP [20]. Aspects have also been explored in the early life cycle phases, including requirement specification and architectural design, resulting in the AOSD (Aspect-Oriented Software Development). According to Baniassad et al. [5], aspects in early phases support easier identification and analysis of aspects during later activities and, as a consequence, a better SoC in the systems.

In this paper, we propose to see SCM activity as a crosscutting concern, aiming at facilitating the development of SCM tools and also aiming at improving reuse and evolvability of such tools. Based on this idea, we present a case study involving the development of a SCM tool, named

<sup>1</sup><http://www.qualipso.org/node/175>

<sup>2</sup><http://www.sei.cmu.edu/cmmi/>

ATCMag (Aspect-oriented Testing Configuration Management tool), for software testing domain. As a result, we have observed that SCM as a crosscutting concern presents perspective to the development of reusable and evolvable SCM tools.

This paper is organized as follows. In Section 2 an overview on SCM is presented. In Section 3 we discuss SCM as a crosscutting concern. In Section 4 we present the SCM tool for testing domain. In Section 5 we discuss about achieved results. Finally, in Section 6 we summarize our contributions and discuss perspectives for further work.

## 2. Overview on SCM

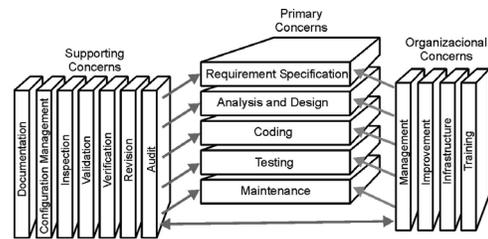
Since SCM emerged in the late 70s and early 80s, considerable efforts can be found in researches on SCM, motivated by the relevance of this activity to software development and evolution [11]. In the context of this paper, SCM is considered a set of tasks concerned with identification, organization, and control of the configurations and changes of a system, including check-in and check-out tasks, in parallel with the software life cycle. In order to support and guide application of SCM, specific standards and guidelines for SCM were established. For instance, we can highlight ISO 10007:2003 Quality management systems – Guidelines for Configuration Management [15] and IEEE 1042:1987 - Guide to Software Configuration Management [2]. In another perspective, we can identify several tools to automate SCM. From popular RCS (Revision Control System) [25], many tools have been developed and used, contributing to overall software product quality. These tools have also been inserted in different software processes, for example, RUP (Rational Unified Process) [19] and agile methods [4, 18]. They have been also inserted in open source software projects [3]. Thus, it is observed that SCM has been successfully applied in processes that present very diverse characteristics. Furthermore, SCM intends to support different software engineering activities. In particular, coding activity has taken real advantages with SCM [12]. In spite of relevance of SCM and diversity of SCM tools, as already pointed out by Estublier [11], these tools are often monolithic and their capability to interoperate is limited. Thus, investigation about how to develop SCM tools, considering their reuse and evolvability seems to be interesting.

## 3. SCM as a Crosscutting Concern

In order to systematize the development of SCM tools in an adequate way, firstly, it is important to understand the relationship between SCM and other software engineering activities and tools. For this, we have inserted SoC in the context of software development process. SoC is a software engineering principle that claims the clear identification of all

the elements that participate in a system, hiding complexity through abstraction [9]; thus, modularity and reusability are achieved through an adequate SoC.

Considering SoC principle in the software development process context, we have referred to each software engineering activity as a concern. A concern is a primary concern or a crosscutting concern. The former refers to primary activities of the life cycle of software, performing development, operation, and maintenance of software systems. The latter, in the same perspective of the AOSD (Aspect-Oriented Software Development) community, refers to concerns that are spread throughout or tangled with others concerns in the software development process; in other words, they occur during primary activities. It is important to highlight that organizational and supporting software engineering activities (i.e., the concerns) have this characteristic and are therefore categorized as crosscutting concerns, as presented in Figure 1. In order to identify all these software engineering concerns, we adopted the international standard ISO/IEC 12207 (Information Technology - Software Life Cycle Processes) [14]. This standard provides a comprehensive set of processes, activities and tasks for software that is part of a larger system, stand-alone software product, and software services. According to this standard, SCM is classified as a supporting concern; it can be conducted from requirement specification to maintenance and it is hence a crosscutting concern.



**Figure 1. Relationship among Primary, Supporting and Organizational Concerns**

We have investigated if SCM seen from this perspective facilitates its automation. The idea is to have a unique SCM tool that manages the configuration items of a whole software project, for instance, system's models, source code, maintenance information, and so on. This same idea is explored in another authors' work [22] that presents a mechanism to automate software documentation activity (also a crosscutting concern).

Furthermore, considering a specific software engineering activity, it is observed that SCM occurs also throughout this activity. For instance, considering software testing, SCM can be performed during different testing tasks, such as test planning, execution and analysis.

## 4. Case Study: Automating SCM for the Testing Domain

Considering SCM as a crosscutting concern, we have developed ATCMag, a tool that automates functionalities related to SCM for the testing domain, aiming at controlling changes in testing artifacts. Following, we present briefly an overview on software testing, discuss about which configuration items (i.e., Testing Configuration Items - TCI) we have adopted and present issues related to development of ATCMag.

### 4.1 Software Testing: A Brief Overview

Software testing is one of the most important activities to guaranteeing the quality and reliability of the software [6]. In this context, a diversity of testing tools has also made the testing a more systematic activity, minimizing the cost, the time consumed, as well as the errors caused by human intervention [21]. In order to support development of testing tools, in a previous work, we proposed a reference architecture for software testing tools, named RefTEST (Reference Architecture for Testing Tools) [23]. This architecture is based on the idea of organizational and supporting software engineering activities as crosscutting concerns, previously discussed. It contains knowledge of the testing domain, representing tasks and their relations. In short, as presented in Figure 2, RefTEST is based on well-established architectures of interactive systems and Web applications (architectural pattern MVC (Model-View-Controller) [8] and three tier architecture [10]). Furthermore, it was established based on software testing processes, architectures of testing tools, an ontology for software testing domain and characteristics and functionalities of well-known and widely used academic and commercial testing tools [23]. It is worth highlighting that `testing_tool` package of RefTEST in Figure 2 contains the core of testing tools. RefTEST is also composed by other three packages automating supporting concerns (`supporting_crosscutting_modules`), organizational concerns (`organizational_crosscutting_modules`) and general concerns (`general_crosscutting_modules`). Package `configuration_management` is detached as a crosscutting module contained in `supporting_crosscutting_modules`.

During the software testing conduction, different testing artifacts, such as test case sets, test requirement sets and error reports, are generated and their control can impact the testing quality. Although SCM is a mature discipline, the use of SCM in the testing has not received adequate attention. In the case study presented in this paper, besides to present an implementation of SCM as a crosscutting concern, we also discuss issues related to SCM in the testing

context.

### 4.2. Selecting Testing Configuration Items

To establish the TCI's, we ourselves adopted IEEE 829-1998 - Standard for Software Test Documentation [13]. This standard provides a complete table of contents for documenting a test suite, independently of the techniques, criteria, methods, strategies, resources or testing tools being used. Thus, in a previous work of our group [7], nine TCIs were identified: test plan, test case specification, test item transmittal, test project specification, test procedure specification, test incident specification, test summary, driver and stub. ATCMag must therefore be able to manage these different TCIs.

### 4.3. Developing ATCMag

In order to develop ATCMag, special attention was taken to its architecture, aiming at providing a better modularity and, as a consequence, reuse and evolvability of this tool. Figure 3<sup>3</sup> shows the main elements that compose ATCMag. In short, it is based on MVC and three tier architecture (presentation, application and persistence) that have been consolidated as architectures that have provided a better SoC in interactive and web systems. Furthermore, it has a fourth layer: the integration. This layer is essential in ATCMag, since it makes possible to implement SCM as a crosscutting concern. While other parts of ATCMag encapsulate functionalities related to SCM, this layer aggregates mechanisms (i.e., aspects<sup>4</sup> from AOP) that affect different parts of a testing tool where must be performed SCM tasks. In other words, these mechanisms establish the communication between ATCMag and the modules of a testing tool. In order to establish which parts of a testing tool must be affected by integration layer, knowledge contained in the RefTEST was used. Thus, based on RefTEST, we identified two modules of testing tools that can provide data to compose the TCIs: (i) `TestingTool` (that implements the core functionalities of the testing tools; for instance, test case and test requirement management); and (ii) `PlanManTool` (that can be part of a testing tool or an independent tool or module that deals with testing planning and management activities). In Figure 3, it is noted that there are <<crosscut>>

<sup>3</sup>Since there is not a widely accepted notation to represent aspect-oriented systems, we have used UML and stereotypes as extensibility mechanisms to represent elements related to the aspects.

<sup>4</sup>In order to facilitate the comprehension of this tool, main concepts from AOP are presented: join points are well-defined points in the execution of the program; pointcuts are a mean of referring to collections of join points and certain values at those join points; advices are method-like constructions used to define additional behavior at join points; and aspects are units of modular crosscutting implementation composed of pointcuts, advices, and ordinary member declarations. More detailed definitions of aspect, pointcut, join point and advice can be found in [17].

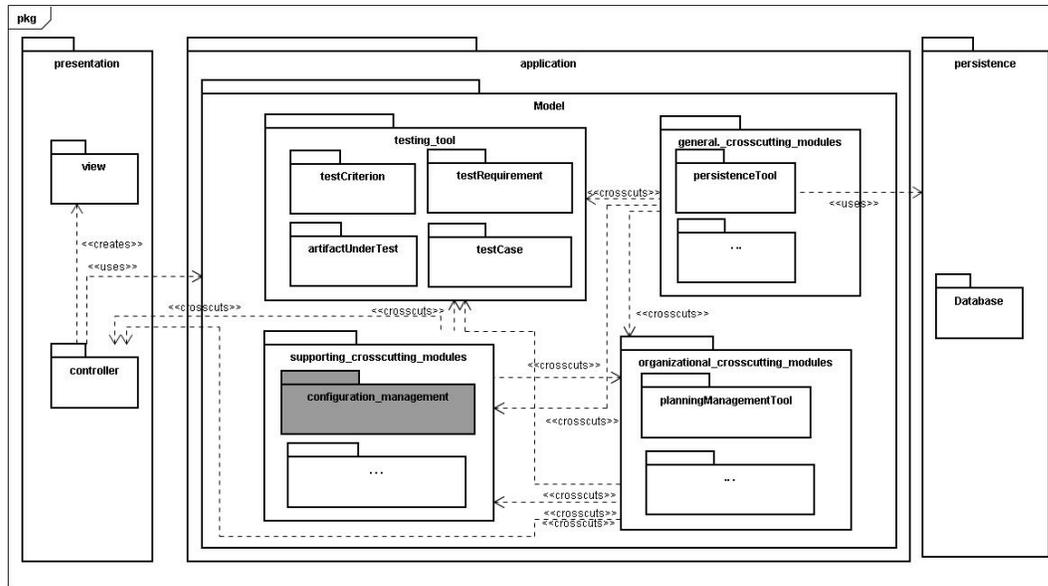


Figure 2. RefTEST Module View and SCM Module [23]

relations from integration layer to TestingTool and to PlanManTool. In more details, the main elements that compose the integration layer are:

- abstract aspect `AbstractConfigurationManagement`: It gets data from `TestingTool` and `PlanManTool`. It also changes the behavior of the testing tool user interface, making available functionalities related to SCM to testing tools. In practice, basically, it declares the abstract pointcuts and implements the crosscutting behavior (advices). It captures the SCM objects and sends them to be treated in `TestingConfiguration` (a class in application layer). A SCM object contains relevant data to compose the TCIs. For instance, a SCM object of the `TestCase` type has an input specification and the correspondent output; and
- concrete aspect `TestingConfigurationManagement`: It extends the abstract aspect and establishes which SCM objects must be intercepted and captured. Thus, the join points are established for each abstract pointcut (declared in `AbstractConfigurationManagement`).

It is observed that aspects are therefore mechanism that makes possible to affect other modules in a “transparent” way, changing behavior of these modules.

Other three layers of ATCMag are:

- Presentation layer: it is composed by `view` and `controller` that implement the user interface required to provide functionalities related to SCM. When

ATCMag is integrated in the testing tool, the behavior of the tool’s user interface is also intercepted by aspects, changing the execution flow and presenting specific windows related to SCM activity implemented by this layer;

- Application layer: class `TestingConfiguration`, `TestingConfigurationItem` and its subclasses deal with the SCM objects captured, extracting relevant data of these objects to compose the TCIs. It is worth highlighting that ATCMag builds the TCIs, since testing tools must not worry about the TCIs. It is noticed that the abstract class `TestingConfigurationItem` was extended to nine concrete classes, one for each TCI. Since XML (eXtensible Markup Language) aims at facilitating the sharing of data across different software systems, as well as it has been widely used as a format for storing and processing documents, we have adopted it to represent and store the TCIs. For this, we have established a DTD (Document Type Definition), named `DTDTD` (Document Type Definition for Testing Documentation), based on the IEEE 829-1998. `DTDTD` has 57 elements, one for each type of information that must be inserted in the TCIs. Based on this DTD, XML files can store the TCIs. Furthermore, we have adopted and used `Subversion`<sup>5</sup>, a well-known and widely used open source version control system, in order to implement SCM functionalities, such as check-in and check-out. In this way, we have taken advantages of the open

<sup>5</sup><http://subversion.tigris.org/>

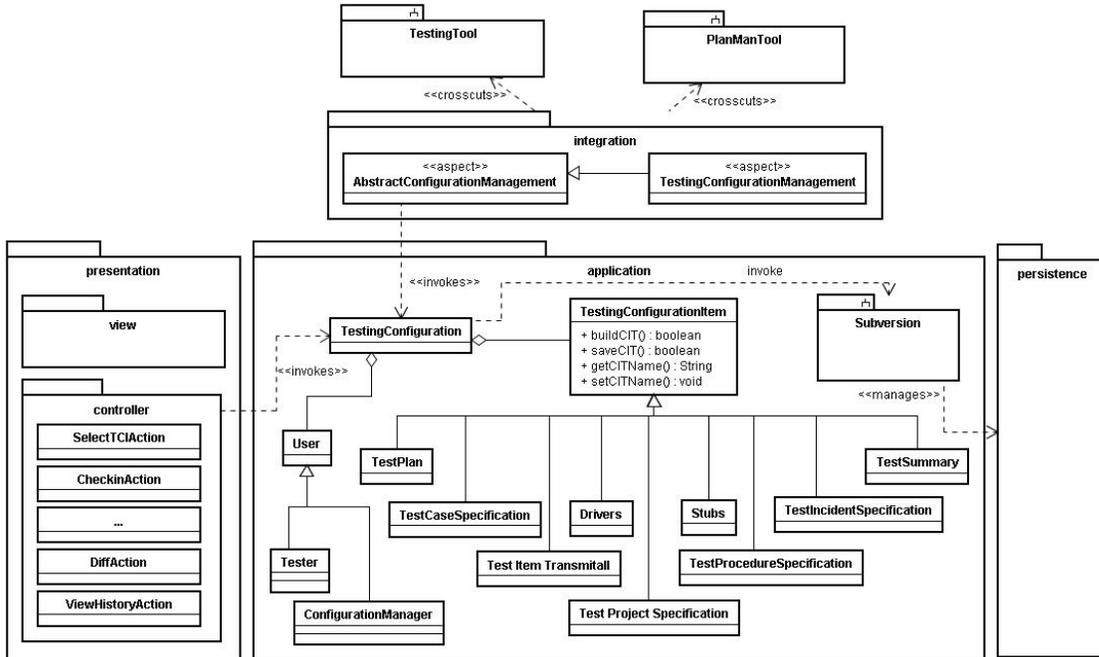


Figure 3. Architecture of ATCMag

source philosophy that has contributed with successful software systems; and

- Persistence layer: This layer aggregates the SCM database that is managed by Subversion.

Regarding technologies to implement ATCMag, we are using Java<sup>6</sup> as the general-purpose language and AspectJ<sup>7</sup> to implement aspects (i.e., the integration layer). Besides that, Struts<sup>8</sup>, an open source framework for building Servlet/JSP based web applications based on the MVC, has been used in presentation layer, together HTML and CSS (Cascading Style Sheet). Figure 4 presents main windows of ATCMag. Through these windows, it is possible, for example, to have access to the tool (Figure 4.a), select the TCIs for a given testing suite (Figure 4.b) and execute other specific functionalities (view, check-out, compare TCIs, and so on) (Figure 4.c).

### 5. Discussion

Our case study has showed that our idea about SCM as a crosscutting concern is viable, i.e., it is possible to implement SCM tool with crosscutting characteristic. SCM tool

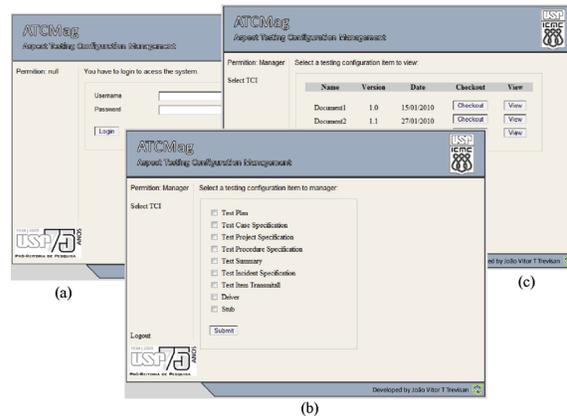


Figure 4. User Interface of ATCMag

is separately developed and, at the same time, it is developed intending an easier integration with other tools. Furthermore, testing tools can be also developed without concern with SCM activity. Our approach enables addition of SCM functionalities later in the testing tools, since aspects contained in the SCM tool present this ability. To facilitate integration between SCM tool and a testing tool, we have used a well-established and documented interface, referred as Document of Interface Description (DID). For instance, DID establishes that a testing tool must have the type `TestCase` containing the data input and output. It is known that not all testing tools will have all the type estab-

<sup>6</sup><http://java.sun.com/>  
<sup>7</sup><http://www.eclipse.org/aspectj/>  
<sup>8</sup><http://struts.apache.org/>

lished. Thus, the SCM tool considers only the intercepted objects and the data contained in these objects in order to generate the TCIs. As a result, during the use of a testing tool, since aspects have the ability to transparently insert SCM functionalities in this tool, a unique integrated tool can be perceived. Regarding reuse, we observe that ATCMag can be integrated in different testing tools, providing therefore reuse of its architecture, internal structures, as well as source code.

Since ATCMag is based on MVC and three tier architecture (two well-known architectures that favour SoC), evolution of this tool can be also benefited. In order to evolve this tool for other software engineering activity, concrete aspect and classes (aspect `TestingConfigurationManagement` and subclasses of `TestingConfigurationItem`, respectively, contained in ATCMag) must be created and adequate to such activity.

In spite of the results achieved in our work, a study using ATCMag in testing tools, as well as the development of other SCM tools must be conducted yet. Thus, more studies are certainly required to achieve a more mature level of evaluation of our approach.

## 6. Conclusions

The main contribution of this work is to present SCM as a crosscutting concern. Based on this perspective, we present a SCM tool for the testing domain. This same experience has been conducted in other crosscutting concerns, such as software documentation [22]. These case studies have pointed out to the feasibility of our idea and showed that crosscutting perspective must be more widely explored to develop software engineering tools, aiming at contributing to reusable and evolvable tools. Motivated by the promising results, we intend to use ATCMag, aiming at observing which is the impact of SCM in the testing processes and, as a consequence, in the testing quality. We intend also to use ATCMag as basis to develop SCM tools for other software engineering activities, such as requirement specification and coding, intending to explore reusability and evolvability of SCM tools built from crosscutting perspective.

**Acknowledgments:** This work is supported by Brazilian funding agencies: FAPESP, CNPq and Capes.

## References

- [1] M. Anastasopoulos. Increasing efficiency and effectiveness of software product line evolution: an infrastructure on top of configuration management. In *Proc. IWPSE-Evol'09*, pages 47–56, 2009.
- [2] ANSI/IEEE. ANSI/IEEE 1042-1987 - Guide to software configuration management, 1987.
- [3] U. Asklund and L. Bendix. A study of configuration management in open source software projects. *IEE Proceedings - Software*, 149(1):40–46, 2002.
- [4] U. Asklund, L. Bendix, and T. Ekman. Configuration management for eXtreme Programming. In *SERPS'03*, Lund, Sweden, 2003.
- [5] E. Baniassad, P. C. Clements, J. Araujo, A. Moreira, A. Rashid, and B. Tekinerdogan. Discovering early aspects. *IEEE Software*, 23(1):61–70, 2006.
- [6] B. Beizer. *Software Testing Techniques*. Van Nostrand Reinhold Company, New York, 2nd edition, 1990.
- [7] A. L. C. V. Boas. Configuration management for software testing. Master's thesis, School of Electrical and Computer Engineering, State University of Campinas (UNICAMP), Campinas, Brazil, July 2003. (in Portuguese).
- [8] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *Pattern-oriented Software Architecture: A System of Patterns*, volume 1. John Wiley & Sons, 1996.
- [9] E. W. Dijkstra. *A Discipline of Programming*. Prentice Hall, 1976.
- [10] W. W. Eckerson. Three tier client/server architecture: Achieving scalability, performance, and efficiency in client server applications. *Open Information Systems Journal*, 10(1), Jan. 1995.
- [11] J. Estublier. Software configuration management: A roadmap. In *ICSE'00*, pages 279–289, Limerick, Ireland, 2000.
- [12] T. Freese. Refactoring-aware version control. In *ICSE'06*, pages 953–956, Shanghai, China, 2006.
- [13] IEEE. 829-1998 – IEEE standard for software test documentation, May 1998.
- [14] ISO. ISO/IEC 12207. Information technology: Software life-cycle processes, 1995.
- [15] ISO. ISO 10007:2003 Quality management systems - Guidelines for configuration management, 2003.
- [16] G. Kiczales and et. al. Aspect-oriented programming. In *ECOOP'97*, pages 220–242, Jyväskylä, Finland, 1997.
- [17] G. Kiczales and et. al. An overview of AspectJ. In *ECOOP'01*, pages 327–353, Budapest, Hungary, 2001.
- [18] J. Koskela. Software configuration management in agile methods. Technical Report 514, VTT Pub, Finland, 2003.
- [19] P. Kruchten. *The Rational Unified Process: An Introduction*. The Addison-Wesley Object Technology Series. Addison-Wesley, 3 edition, 2003.
- [20] R. Laddad. Aspect-oriented programming will improve quality. *IEEE Software*, 20(6):90–91, Nov.-Dec. 2003.
- [21] G. J. Myers, C. Sandler, T. Badgett, and T. M. Thomas. *The Art of Software Testing*. John Wiley & Sons, 2004.
- [22] E. Y. Nakagawa, M. M. F. Sasaki, and J. C. Maldonado. An aspect-oriented framework for software documentation: An example on testing. In *IDEAS'2009*, pages 225–238, Medellín, Colombia, 2009.
- [23] E. Y. Nakagawa, A. S. Simão, F. Ferrari, and J. C. Maldonado. Towards a reference architecture for software testing tools. In *SEKE'2007*, pages 1–6, Boston, USA, July 2007.
- [24] R. Raygan. Software configuration management applied to service oriented architecture. In *SoftCOM'07*, pages 1–5, Dubrovnik, Croatia, 2007.
- [25] W. F. Tichy. RCS - A system for version control. *Software Practice and Experience*, 15(7):637–654, 1985.

# Decision Support for Staffing of the Next Software Product Release

Emadoddin Livani

*Electrical and Computer Engineering  
Department, University of Calgary  
elivani@ucalgary.ca*

Guenther Ruhe

*Department of Computer Science and  
Department of Electrical and Computer  
Engineering, University of Calgary  
ruhe@ucalgary.ca*

## **Abstract**

*Staffing is one of the key project management functions. In this paper, optimized staffing and re-staffing of human resources is studied. Four staffing and re-staffing scenarios are investigated. Their optimization-based assignment of developers to tasks takes into account the specific skill profiles of developers. The approach is evaluated by an industrial case study project. Productivity profile of developers was defined using the Analytical Hierarchy Process and analyzing former project performance. For generating optimized staffing and re-staffing policies, the staffing component of the decision support tool™ was applied. The case study results demonstrate that the optimized plans clearly outperform the manual ones in terms of their quality measured as project duration and their transparency.*

**Keywords:** *Decision support, release planning, software project management, staffing, case study.*

## **1. Introduction**

The discipline of project management is concerned with planning, organizing, leading, staffing, and controlling of projects [1]. Staffing is one of the key project management functions and is defined as the process of assigning human resources to a sequence of feature related tasks. Staffing takes into account the different competence levels of developers. The study in [2] shows this ratio can be 20 to 1. The general finding that there is one order of magnitude difference among programmers has been confirmed by other studies such as [3], [4], and [5] as well. Acknowledging that all developers have certain strengths in their competence profile, the question becomes identifying those people that are best suited for different development roles. Despite its importance, there is little support for this staffing task, leading to unease and misunderstanding among the people involved [6].

The main interest of this research is on providing decision support for staffing and re-staffing of the next software product release. This is a refinement of

planning at strategic level where resources are considered in a cumulative way, without looking into the details of an individual task to be done at a specific interval. The different levels of skill of the developers have been left out of consideration at strategic level as well [7].

Assuming a baseline staffing plan, four scenarios for proactive and reactive re-planning have been investigated in this work. Proactive re-planning tries to go through different scenarios upfront to better understand their possible impact and to derive actions for risk mitigation. Reactive staffing refers to the need to change things in the course of an ongoing development. In both cases, staffing is supported by the staffing component of a decision support tool called ReleasePlanner™ [8].

The structure of the paper is as follows. In Section 2, related work is discussed. The problem statement with four specific staffing scenarios is given in Section 3. The proposed solution approaches are introduced in Section 4. Main context and results from the industrial case study are presented in Section 5. Finally, Section 6 concludes the paper and gives the outlook to future research.

## **2. Related work**

Different methods exist for release planning at strategic level. They have been recently analyzed in [9]. The most recent model called EVOLVE II allows compatibility with staffing and operational planning.

Van den Akker et al. [10] applied mathematical programming to provide a solution for the next release problem. Their emphasis is on different scenarios for teams needed to implement the features. Different teams and exchange between teams is considered as well as hiring external personnel and/or extending the release dates.

The high complexity of both the planning and re-planning process was discussed and evaluated in [11]. In order to decide (i) which resources are used to perform which tasks and (ii) when each task should be performed, the authors of [12] suggested applying

genetic algorithms. The proposed method focused on schedule minimization and did not examine different productivity level of developers.

Several search-based techniques (genetic algorithm, hill climbing, and simulated annealing) were used for resource allocation in large massive maintenance projects [13]. The approach was validated by an empirical study of a large, commercial Y2K massive maintenance project, which compared these techniques with each other. According to the authors, the best search technique reduces the project duration by 50% compared to the application of baseline random search.

In summary, existing methods and techniques were either not applicable to the general staffing formulation presented in this paper, or their applicability was just shown for small scale examples without demonstrating the practical validity. Also the re-staffing problem which is the main focus of this paper has not been investigated in depth in the previous works.

### 3. Problem statement

#### 3.1. Staffing for minimum project duration

For a given set of features requested for implementation, the planning objective is to implement all the proposed features in minimum overall project duration. This is also called *make-span minimization*. More precisely, the problem formulation is composed by the following modeling components:

A set  $F$  of features is specified for the next (1)  
software product release.

Each feature consists of a sequence of related (2)  
tasks to be executed.

The makespan  $(F, x)$  for realizing feature set  $F$  (3)  
according to plan  $x$  is defined as the latest finish  
time of a feature.

A pool of developers with different levels of (4)  
productivity for the different types of tasks is  
available.

Each developer is working on no more than one (5)  
task at a time.

The productivity of developer  $d$  with regard to (6)  
task  $q$  is defined as *effort of task  $q$  realized by  
developer  $d$  in one time unit* and is denoted by  
 $prod(d, q)$ .

The effort of task  $q$  of feature  $n$  is defined as the (7)  
amount of time units needed to be fulfilled by  
an average developer and is denoted by *effort*  
 $(n, q)$ .

The average productivity of a developer  $d$  for (8)  
task  $q$  is normalized to  $prod(d, q) = 1$ .

Dependencies between tasks of the same (9)  
features are the same for all tasks.

The goal of the assignment problem is to (10)  
minimize makespan  $(F, x)$  regarding (1) to (9).

Changes can happen at any time during the release period. They can be known in advance (before starting the implementation) or in the course of implementing features. We call the re-planning for the former case proactive and the latter one reactive staffing. Proactive staffing can also be done to better evaluate the plan in the sense of tolerating possible future changes.

We investigate the characteristics of proactive and reactive staffing as four decision support scenarios. They are described in the next subsections.

#### 3.2. Decision support scenario 1: analysis of changing in productivities

The first scenario is devoted to better understanding of the impact of changed productivity of the developers. This is motivated by the question whether investment into training of personnel or hiring of more skilled personnel pays off. Here is the summary of this scenario:

- Given: (i) Baseline staffing plan generated from base level of productivities. (ii) Variation of productivity of some developers for the tasks.
- Question: What is the impact of productivity changes on make-span?

#### 3.3. Decision support scenario 2: unpredicted absence of developers

As the second scenario, the impact of periods of absence of developers is studied. Sometimes the absence is known before the start of the plan. So there is an opportunity to consider the absence in the planning (proactive re-planning). But it could also happen in the middle of an ongoing plan and therefore reactive re-planning is required. We consider both cases of proactive and reactive re-planning here.

The assumption is that no additional developer would be available at the time for replacement. Consequently, the tasks formerly assigned to the developers being absent need to be done by other developers. Very likely, the whole staffing plan needs to be changed. Here is the summary of this scenario:

- Given: Base plan, developers absent period
- Question: How to do re-staffing in case of predicted or unpredicted absence? What's the impact of the absence?

#### 3.4. Decision support scenario 3: revised efforts per task

The third scenario investigates the impact of changed efforts for some tasks. Uncertainty in effort estimation among others results from the lack of detailed knowledge about the tasks to be done. This is especially true at early stages of the development (e.g.,

during design). Consequently, it is quite common that effort estimates need to be adjusted in the course of the project. Re-staffing to accommodate these changes has been observed being difficult [11].

Another possible reason to revise the workloads refers to the situation to accommodate an additional feature in the middle of an ongoing implementation phase. The question becomes: to what extent would this additional feature delay the established schedule? And what actions need to be taken for that?

The summary of this scenario is as follows:

- Given: (i) Base plan and (ii) revised efforts of some features or coming of new features
- Question: What action should be taken to accommodate change in a best possible way?
- Assumption: New features are strictly forced to be implemented

### 3.5. Decision support scenario 4: hiring new staff

Decision support for the selection of new staff and the subsequent job assignment is rare. Existing approaches are mainly based on standard database queries that do not tackle the complexity of the task appropriately. The reason is that they focus solely on the fitness of the person with the job, neglecting the fitness of the person with the team, which is important as well [14]. The final scenario considers the case that for reducing the duration of the project, the project manager considers to hire a new developer to join the team. To summarize this scenario:

- Given: base plan, productivity profile of the candidates
- Question: which candidate is the most suited for the project?

## 4. Solution approach

Assigning most competent developers to appropriate tasks is a complex process. If this process is executed in an *ad-hoc* manner, and solely based on intuition, then the likely results will be poor resource utilization and schedule overrun [11].

For the purpose of this research, we have extended and applied the decision support tool ReleasePlanner™, able to support decision-making on both strategic and operational level. The tool applies special purpose algorithms from mathematical optimization to generate solutions with proven degree of optimality [15]. For all scenarios, the solutions gained from the tool are compared with a baseline plan. The baseline plan is shown by  $x_0$  and the optimized plan by  $x_{opt}$ . For the case study performed in Section 5, the baseline plan represents the manual plan generated from applying some form of the greedy planning technique [16].

For a given set  $F$  of features considered for the next release as a result of strategic planning, the ratio

$$\Delta = \text{makespan}(F, x_{opt}) / \text{makespan}(F, x_0) \quad (11)$$

describes the degree of change in project duration obtained from applying the optimized staffing versus the baseline staffing.  $\Delta < 1$  indicates a reduction in duration.

## 5. Case study

We did a case study for a software development company named Douran Co. [17]. They use an iterative development process which is a customized version of Rational Unified Process [18]. We selected a part of an ongoing project which had two iterations. The purpose of this case study was to understand the effect of predicted and unpredicted changes during the development process on the two solution approaches of Section 4.

We Assume implementation of a feature requires performing three tasks called *design*, *coding*, and *test*. Each feature can only be offered if all the three related tasks are performed. There are dependencies between the starting dates of tasks of the same feature which need to be fulfilled: (i) coding cannot start before design starts. Similarly, (ii) test cannot start before coding starts. There are similar dependencies between the end dates: (iii) coding cannot finish before design finishes, and (iv) test cannot finish before coding.

Each iteration is three weeks long (15 business days). There are 50 features and 5 developers available to implement them. The total effort required for implementing all the features is 1200 person/hours. Features and detailed efforts are available in [19].

Having two iterations (6 weeks, 5 business days per week, and 8 hours per day) and 5 developers, the total available human resource is 1200 person/hours. Since the total effort is 1200 person hours, the iteration seems to be feasible. But as mentioned before it highly depends on the productivity of the developers. Also precedence between tasks is another constraint which makes this problem more complicated. So we cannot be sure about the feasibility of the plan unless we create it.

To obtain the productivity of the developers we used Analytical Hierarchy Process (AHP) [20]. The pairwise comparisons of the developers for the three tasks were provided by the project manager at the company, based on the historical data about the performance of the developers for different tasks. The result of the AHP computations shows the weight of the developers for each task. E.g. (0.084, 0.184, 0.261, 0.133, 0.338) is the weight of the 5 developers for design task. The sum of these weights is 1 and the average is 0.2. To normalize these values to a

distribution around 1, we devised them by 0.2. Having done this process for all the tasks, the productivity profile of the developers is calculated as Table 1.

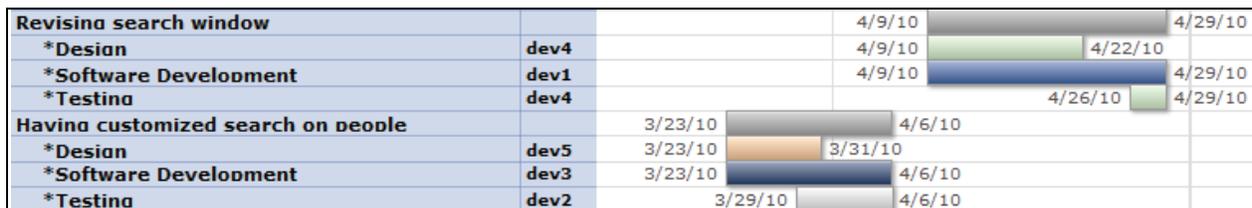
**Table 1 Developers' skill profile**

Name	Design	Coding	Test
Dev1	0.4	0.6	0.5
Dev2	0.9	0.3	1.2
Dev3	1.3	1.3	0.6
Dev4	0.7	0.4	1.7
Dev5	1.7	0.9	1

The case study includes creating the baseline plan (with both solution approaches stated in Section 4) and then re-planning for each scenario of Section 3. The baseline staffing plan serves as a reference point. The duration of the plans and  $\Delta$  are used for comparing the two plans.

### 5.1. Staffing results

The staffing at Douran co. is basically based on a Greedy-like algorithm without considering the productivity of the developers. We've simulated this method and created the baseline plan for our case



**Figure 1 Part of the plan created by the staffing component of ReleasePlanner™**

### 5.2. Decision support scenario 1: analysis of changing productivities

The first re-planning scenario is devoted to studying the effect of improved productivities of developers on the created baseline plan. We assume through a training program, it's possible to increase one skill (among design, coding, and test) of the developers by 20%. We want to select the best skill and find if this program pays off. Re-running the computations by ReleasePlanner™ for the same set of features and improved productivity of developers for design, coding, and test tasks (each one separately) resulted in new plans with 204, 194, and 210 hours duration respectively. So training the coding task pays off more than the others. This was expected, but the main point here is that increasing only the coding ability of the developers by 20% resulted in 9% reduction in the overall duration, compared to the first optimized plan. So, proactive re-planning shows how better we can get from the developers before starting the actual plan.

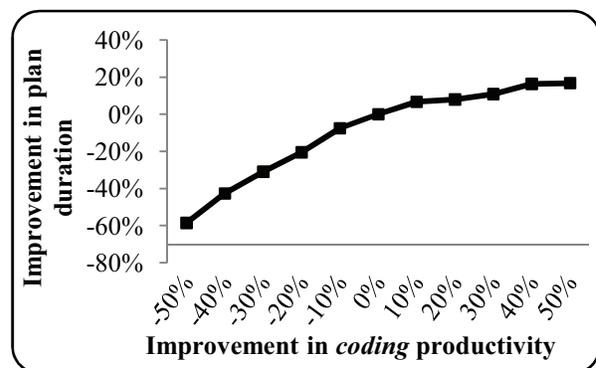
study. The duration of this plan is 303 hours. Having 240 hours maximum duration, this plan takes 63 hours (a week and half) longer than 6 weeks and obviously is not feasible.

Also regarding to the resource consumption, having 303 hours duration and five developers mean consuming 1515 person/hours resource by the manual plan. Considering the 1200 person/hours available resource in the project, this plan is over-consuming the resources by 315 person/hours.

As the second solution approach, we ran the computations using ReleasePlanner™. The optimal solution had 215 hours duration and therefore 1075 person/hours total resource consumption. So this plan is shorter than the manual plan ( $\Delta = 0.70$ ) and is feasible too. It also consumes 440 person/hours less human resource, which is a lot of saving in time and budget.

Figure 1 shows a part (only two features) of the Gantt chart obtained from the tool for the optimized plan. This means to illustrate what the plan means and how the task precedence rules work.

In the following subsections we address the four decision support scenarios introduced in Section 3.



**Figure 2 Sensitivity analysis for coding productivity**

Since the productivity of the developers plays an important role in the planning process, we've done sensitivity analysis by varying them for coding task from -50% to +50% of the current situation and measured its effect on the duration for the optimized solution. As Figure 2 implies, the relationship between improvement in plan duration and productivity of the developers is non-linear. It increases with a high pace

from -50% to -10% improvement in productivity. ‘-‘ improvement in productivity means decreasing it and ‘-‘ improvement in plan duration means taking longer comparing to the baseline plan. After this point the slip of the improvement decreases until the improvement in productivity reaches 30%. After this point the improvement increases with a lower slip.

### 5.3. Decision support scenario 2: unpredicted absence of developers

In this scenario the effect of absence of developers is being studied. We assume developers dev5 and dev3 are not available in the period of [41, 64] and [81, 104] respectively. We consider the both cases of predicted and unpredicted absence of these developers.

Using the ‘Absence Periods’ feature of ReleasePlanner<sup>TM</sup> we defined the periods of absence and re-ran the computations. The new plan had 226 hours duration i.e. 1130 person/hours resource consumption. Manual re-planning resulted in 321 hours duration in this case. So optimized re-planning created a feasible plan ( $\Delta = 0.70$ ).

In the case of an ongoing plan the scenario is different. Now the first plan has been followed up until the point  $t=40$  hours. We pre-assigned the first plan until this point (in ReleasePlanner<sup>TM</sup>) and then re-ran the computations with the first absence period (note that we don’t know about the second one yet). Therefore, we are simulating the reactive re-planning for time  $t=40$  hours. We did the same thing for point  $t=80$ , which developer dev3 will be unavailable. But this time, we pre-assigned the tasks based on the new plan, resulted after  $t=40$ .

Re-planning at time  $t=80$  resulted in a new plan with 238 hours duration or 1190 person/hours resource consumption. Same thing with manual computations results in 323 hours duration. The effect of unpredicted absence is higher than the predicted one in optimized staffing (5%). But it’s negligible in the manual plan (0.3%). The reason is manual plan doesn’t consider the absence periods in the planning process while optimized staffing creates a better plan if periods of absence are known in advance.

One may ask what if we don’t re-plan and stick to the current plan (this was the case happening in the company). We calculated this case for both solutions. Results show 263 hour duration for optimized plan (it would have been infeasible plan without re-planning) and 351 for manual.

### 5.4. Decision support scenario 3: revised efforts per task

As discussed in Section 3, revising the workload may happen either as the result of re-estimation of efforts or adding (or removing) a new feature. Assume at the time  $t=120$  (after first iteration) the re-estimation

of the features is done and causes task *coding* of every feature  $n$  with effort ( $n, \text{coding}$ )  $> 10$  to take 30% longer. These are the features which according to the first plan would be implemented in the second iteration. Reactive re-planning resulted in a new plan with 239 hours duration, which is a feasible plan, vs. manual computations with 328 hours duration.

Assume we haven’t done the reactive re-planning and have continued with the first plan (with the same task assignments). Then it would have resulted in a plan with 248 hour duration (infeasible plan) for optimized plan vs. 351 for manual.

For the second case, we added two new features with workload (15, 25, 10) each. Again, assume this event happens at point  $t=120$  hours (separate from the previous case). Reactive re-planning resulted in a new plan with 238 hours duration, which is a feasible plan. Manual solution creates a plan with 338 hours duration.

Assume we don’t re-plan and simply just add the features to the end of the plans. Then the optimized plan would have taken 253 hours (infeasible) and manual plan 350 hours.

### 5.5. Decision support scenario 4: hiring new staff

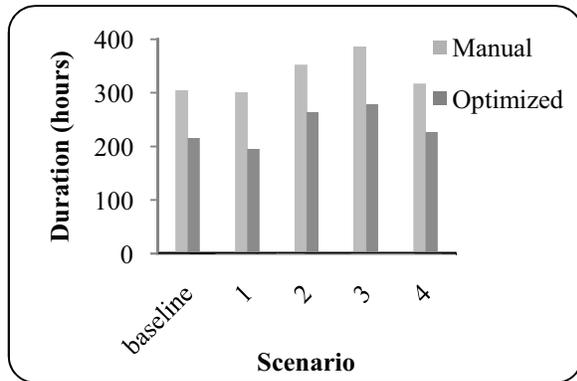
In the previous section the effect of revised workloads was discussed. Assume both cases (re-estimating the efforts and adding new features) happen. Re-planning results in a plan with 247 hours duration, which is infeasible. So the project manager has to take an action to meet the deadline. Assume the project manager decides to add a developer to the team (at time  $t=120$  hours) and there are three possible candidates; each one can only help in one type of task. So the candidates have productivity of 1 for one task and 0 for the other two. The goal is to find the best candidate who minimizes the plan duration. So, we re-planned at point  $t=120$  three times, each time with one candidate. The resulted optimized plans for the three candidates had 229, 226, and 244 hour duration. So the first two candidates can help the project in order to meet the 240 hours time limit while the last one cannot.

## 6. Conclusion and future work

In this paper we introduced the problem of proactive and reactive staffing and investigated four decision support scenarios. The four scenarios of re-planning give insight into the possibilities to proactively or reactively adjust to changed project situation. The analysis was done from the perspective of staffing for shortest overall project duration (or minimizing make-span).

Figure 3 summarizes the results of the experiments done in the case study (more details in [19]). It

compares the reactive manual (the company's approach) and proactive optimized (our approach). The results show that plans created by optimization method are always shorter than the manual ones. Also re-planning (with either solution) is always better than continuing with the ongoing plan. Proactive re-planning before starting the plan helps to mitigate the potential risks that might happen during the development. There is an about 30% time reduction in comparing reactive re-planning of optimized plan with manual 'not re-plan' case for all the re-planning scenarios.



**Figure 3 Case study summary**

There are some challenges in the decision modelling process. For example productivity of the developers and workload of the features need to be estimated. In this research, we've applied Analytical Hierarchy Process (AHP) on real data to create precise productivities. The baseline plan created by manual solution is almost identical to the one used by the company. Also, for the four scenarios the 'not re-plan' alternative was always computed, that reflects the real case happening in the company. Also, ReleasePlanner™ has been used in many case studies and has proven to generate valid solutions in both cases of planning and re-planning with constraints [21].

This paper illustrates some of the features that seem to be necessary for a decision support system for project staffing. Further empirical evaluation is necessary to demonstrate the usefulness and scalability of the proposed approach. The next step would be extension of the existing tool ReleasePlanner™ to support what-if analysis for staffing and built-in sensitivity and cost/benefit analyses.

At the end, we would like to thank Natural Sciences and Engineering Research Council (NSERC) of Canada for partly supporting this research under discovery grant no. 250343-07. We also would like to appreciate team members of Douran Co. for their support and assistance when running the case study.

## 7. References

- [1] R. K. Wysocki, *Effective software project management*, Wiley, 2006.
- [2] H. Sackman, W. J. Erikson, and E. Grant, *Exploratory Experimental Studies Comparing Online and Offline Programming Performance*, *Communications of the ACM*, 11 (1968), pp. 3-11.
- [3] J. D. Valett, and F. E. McGarry, *A summary of software measurement experiences in the software engineering laboratory*, *Journal of Systems and Software*, 9 (1989), pp. 137-148.
- [4] T. DeMarco, and T. Lister, *Peopleware: Productive Projects and Teams*, 2nd ed., Dorset House, 1999.
- [5] L. Constantine, *The Peopleware Papers: Notes on the Human Side of Software*, Prentice Hall, 2001.
- [6] S. T. Acuna, N. Juristo, and A. M. Moreno, *Emphasizing Human Capabilities in Software Development*, *IEEE Software*, 23 (2006), pp. 94-101.
- [7] A. Ngo-The, G. Ruhe, "A Systematic Approach for Solving the Wicked Problem of Software Release Planning", *Soft Computing*, Vol. 12 (2008), pp. 95-108.
- [8] <sup>TM</sup>, [www..com](http://www..com), Last access: March 2010.
- [9] M. Svahnberg, T. Gorschek, R. Feldt, R. Torkar, S. B Saleem and M. U. Shafique, , *A Systematic Review on Strategic Release Planning Models*, *Information and Software Technology*, Vol. 52 (2010), pp. 237-248.
- [10] M. van den Akker, S. Brinkkemper, G. Diepen and J. Versendaal, *Software Product Release Planning through Optimization and What-If Analysis*, *Information and Software Technology*, Vol. 50 (2008), pp. 101-111.
- [11] P. Kapur, A. Ngo-The, G. Ruhe, and A. Smith, *Optimized Staffing for Product Releases and Its Application at Chartwell Technology*, *Journal of Software Maintenance and Evolution*, Vol. 20 (2008), pp. 365-386.
- [12] E. Alba, and J. F. Chicano, *Software Project Management with GA's*, *Information Sciences*, Vol. 177 (2007), pp. 2380-2401.
- [13] G. Antoniol, M. D. Penta, and M. Harman, *Search-Based Techniques Applied to Optimization of Project Planning for a Massive Maintenance Project*, *Proceedings IEEE International Conference on Software Maintenance, ICSM, 2005*, pp. 240-252.
- [14] M. A. West, and N. A. Allen, *Selecting for teamwork*, *International Handbook of Selection and Assessment*, (1997), pp. 491-505.
- [15] G. Ruhe, *Product Release Planning – Methods, Tools, and Applications*. CRS Press, appears in June 2010.
- [16] T. Cormen, C. Leiserson, and Z. Rivest: *Introduction to Algorithms*. MIT Press 1990, Chapter 17 "Greedy Algorithms" p. 329.
- [17] <http://www.douran.com/HomePage.aspx?Lang=en-US>
- [18] <http://www-01.ibm.com/software/awdtools/rup/>
- [19] <http://www.ucalgary.ca/~elivani/StaffingSEKE10.xls>.
- [20] T.L. Saaty, *The Analytic Hierarchy Process, Planning, Priority Setting, Resource Allocation*, McGraw-Hill, New York, 1980.
- [21] A. Ngo-The, G. Ruhe, *Optimized Resource Allocation for Software Release Planning*, *IEEE Transactions on Software Engineering*, Vol. 35 (2009), pp 109-123.

# Predicting Project Health Prior to Inception

Rose Williams, M.S.C.S., Krishna Ratakonda, PhD.  
Department of Software and Services Science  
I.B.M. Thomas J. Watson Research Center  
Hawthorne, N.Y. 10532, U.S.A.  
[Contact: rosemw@us.ibm.com](mailto:rosemw@us.ibm.com)  
914-784-7794

Jim Graham, Katrina Reffett, Myles Wallace  
Department of Risk Management  
Global Business Services  
I.B.M.  
U.S.A.

## ABSTRACT

*The causes for the failure of complex IT projects are well known in the computer industry. Assessment of project risks typically include metrics gauging team performance, stakeholder commitments, schedule adherence, among others. The process of obtaining an aggregated delivery risk score for the project involves assigning a score to each of the individual dimensions and then aggregating the results either linearly or through a more complex aggregation function. The outcome is typically an ordinal grade such as “A” through “D” where “A” denotes a high performing project while “D” denotes a poorly performing project. Thus, project performance studies typically look “backwards” at the historical trend of root causes, as they evaluate “what went wrong.” Furthermore, this measurement is quickly outdated and the process of assigning scores to project performance measurement can only provide a point-in-time- measurement. Given the overhead involved in repeating the assessment, the process of evaluation is repeated once a quarter or less frequently*

*We have developed a predictive algorithm which gives the risk manager a score indicating the likelihood of project success or failure before the project begins. This opportunity allows us to quantify the “odds” of the next troubled project, thus supporting managers in effective risk mitigation.*

## 1. INTRODUCTION

Patterns of failure that emerge can be associated with a number of root causes, many of which are well known, such as: requirements creep, sudden changes in scope/schedule changes or poor project management practices. Most systems that provide models of project risk are narrowly focused on checklists, or include naive scoring systems. Some systems generalize mitigation actions.

Typically found in these systems are delivery risk metrics which are the result of aggregation and basic computation on the following attributes:

- Are stakeholders are committed?
- Are business benefits being realized?
- Are Work and Schedule predictable?
- Is the Team High Performing?
- Is the scope realistic and managed?
- Are risks being mitigated?
- Are delivery organization benefits being realized?

A variant of such a measurement system is used by most large enterprises which undertake many complex projects simultaneously. The process of obtaining an aggregated delivery risk score for the project involves assigning a score to each of the individual dimensions and then aggregating the results either linearly or through a more complex aggregation function. The outcome is typically an ordinal grade such as “A” through “D” where “A” denotes a high performing project while “D” denotes a poorly performing project. The process of assigning scores to individual keys is typically performed by a certified risk management professional or project manager. However, such performance measurement can only provide a point in time measurement of project performance that is quickly outdated. Furthermore, the process of obtaining the risk measurement is often labor intensive. Given the overhead involved in repeating the assessment, the

process of evaluation is repeated once a quarter or less frequently.

The purpose of our system is to provide a method for using the data available before project inception to predict the project's performance as indicated by an aggregate delivery risk score a short period (3 months to a year) after the project goes into delivery. In our prediction algorithms, we focus on the analysis of the quantitative metrics collected at the project proposal stages of an IT engagement to gain an early insight into key patterns of trouble. The emphasis is on both identifying a pattern of trouble early into the engagement cycle, as well as gaining a quantitative understanding of the scope of the problem during development or delivery of service.

The attributes used in our algorithms are derived from a set of questions that leverage the experience and intuition of risk managers. Our system essentially quantifies 'the odds' of the next likely troubled or successful project, and thus enables risk and project managers to plan appropriately. Our method, called Initial Delivery Index (IDI), has been deployed to risk managers world wide for more than a year, and the actions taken by risk managers in response to the IDI predictions have positively influenced the outcomes of projects.

## 2. BACKGROUND

Software risk management is a well established discipline [1, 2] that has generated continued academic interest as the complexity and nature of the software projects have evolved over time. Traditional risk management techniques have been focused on identifying and codifying best practices that prevent or reduce the failure rate [3, 4, 5, 6, 7]. Large IT organizations have assimilated many of these findings in their risk management practice. However, adopting these best practices does not guarantee that risk is eliminated or even reduced to an acceptable level – new software development models driven by globalization, competition and an ever changing software landscape throw up new patterns of trouble.

In recent literature, the trend towards a systems approach to risk management, wherein statistical techniques are used to classify and mine the project metrics data, with a view to pro-actively learn the new trends is gaining acceptance [8, 9, 10]. These methods actively use the metrics collected during traditional risk management reviews and then employ techniques to derive models that describe the relationship between the collected metrics and eventual project outcomes. It is important to note that

these techniques in most cases cannot identify *why* a project may be troubled i.e., they can only serve as a starting point for further investigation. The ability to use the results of the analysis in a dispassionate manner to start and cull IT projects is still a contentious topic [11]. As Keil et al. [12] observe, there is a strong bias in many organizations to continue with IT projects even though financial analysis indicates otherwise. Despite the intensive research into models for project risk assessment and management, there are still significant numbers of large IT projects failing.

In addition to understanding the health status of a project under development, we have demonstrated that it is equally important to predict the likelihood of project failure (or success) *before* the project begins - at the project proposal stage. The advantages of our IDI prediction are many. IDI predicts both "good" and "poor" projects. Depending on the prediction, this allows the user to take actions that may allow more competitive pricing, or more client favorable terms. This prediction also allows scarce resources to be allocated in a prioritized manner before the project begins. Risk and project managers may take actions that will reduce risk prior to project inception.

## 3. METRICS AND STATISTICS

In this section, we introduce a few of the metrics that we include in our data set, and particular attributes of our classification model that are particularly important to our risk managers. In this paper, we will focus on the algorithms that analyze the bidding and early phases of an IT engagement. We illustrate the complexity of attributes and the information produced by our classification models.

A number of risk assessment related activities are conducted prior to project inception with an objective of determining the risk entailed in delivery. These are called our pre-engagement metrics. We collect a set of 16 questions that are answered prior to project on-boarding (inception). We then use a classification algorithm to predict the value of the letter grade that will be assigned to the project in the first few project management reviews based on the answers to the question set. Towards this end, we collect a statistically representative set of historical data from past projects that span a wide variety of geographies, sectors, sizes and contract structures.

The prediction model is based on a Bayesian decision tree learning algorithm that has been modified to improve the accuracy and precision when subject to

variability in input data. Decision trees provide an effective means to predict outcomes that are generated by a combination of factors, each of which individually may not be strong predictors of the outcome. Generating an optimal decision tree model given a historical data set is a challenging computational problem as one has to balance precision, accuracy and robustness.

The model once generated goes through a six-fold cross validation to ensure that the model is not over fitted and that the results can be reproduced in practice. In addition to a predicted letter grade rating for a project, we also provide an estimated probability that the predicted grade will be realized. This probability can be interpreted as a confidence measure for the prediction.

There are several factors that distinguish our approach from typical risk management approach:

- (a) We use a set of questions, which not only comprise questions that determine complexity or nature of the project, but assess the risks entailed due to the nature of the client relationship, the ability of the delivery organization to assign resources to the project on a timely basis, contract structure related issues and past history from the dimensions of an industry sector, technology or geography,
- (b) We use objective statistical learning theoretic measures to determine a predicted letter grade for the project instead of choosing a simple heuristic like averaging the answers to the questions,
- (c) Each possible grade is assigned a probability by the algorithm instead of making a single prediction which helps ensure a measured response that is proportional to the perceived risk and
- (d) the questions themselves are calibrated to serve as indicators of trouble-spots i.e., a higher score elevates the delivery risk – thus they can be used in conjunction with the risk assignment to identify remedial actions for trouble spots.

Another aspect of our system is its ability to incorporate real time changes to the answers to the questions, which can then be used in conjunction with the analysis algorithm to generate an alert when certain parameters, such as gross profit, go out of bounds after project inception. Thus, the static prediction obtained at project inception can be tweaked in real time to respond to the dynamic project situation.

The key decisions facilitated by this process include the amount of contingency budget that is allocated to the project and the frequency/composition of project

management reviews (PMR) during the project lifecycle.

All inputs to IDI, except the project specific answer to each of the 16 questions, are pre-determined based on the past performance of projects within the database. The database is refreshed and expanded every 6 to 9 months. IDI is not a customizable framework, but is a predictive tool.

The result of IDI is a prediction of future project performance displayed in a fixed format. IDI predicts that a project will have one of 4 project “health check” ratings: A, B, C or D as defined below:

**A:**  
**Project is under control. Minor problems may exist, but the Project Manager has an effective plan in place to solve the problems. No major existing or potential problems have been identified.**

**B:**  
**Project is currently under control. However, existing or potential problems have been identified which will require positive management attention in order to keep the project under control.**

**C:**  
**Significant problems currently exist which require corrective plans. Probability exists for exceeding estimates or budgets, customer dissatisfaction, and/or limited financial exposure. Aggressive management action is required to bring the project under control.**

**D:**  
**Major problems exist with definite, serious financial exposure and/or customer dissatisfaction.**

The probability percentage of the prediction occurring is also calculated and displayed, as examples:

PRED	A_PROB	B_PROB	C_PROB	D_PBOB
A	58	17	17	8
OR...				
PRED	A_PROB	B_PROB	C_PROB	D_PBOB
C	0	25	50	25

In our system, IDI looks forward in time to predict success or failure of a project, and is not based on traditional “root cause” analysis. IDI is conducted *before* the project is underway. We use this

information to predict the outcome of project management reviews. The first project management review is conducted and a letter grade is assigned within 12 weeks after project inception. IDI is also used to help make “no bid” or “do not proceed” decisions regarding certain proposals.

In the US and Canada over 2,000 project proposals have had IDI applied. The effective accuracy has ranged from 94% to 98% over the various measurement periods.

#### 4. EXECUTIVE VIEWPOINT

IDI is breakthrough thinking in optimizing operations. By channeling IBM Research mathematic skills, and senior risk manager expertise into analytics with feedback from actual results enables leadership to make better and earlier decisions in our client solutions. IDI, like other portfolio and project predictive analytics that we have implemented, adds value to the business by providing headlights into potentially troubled projects early in the process, allowing corrective actions to be taken. Substantial sums over the last three have been invested in troubled projects that could have been at least partially avoided with enough early warning to take actions.

#### 5. RECOMMENDATIONS

The following list summarizes important considerations in using predictions to warn about the likelihood of a troubled deal / project. They are:

- The set of survey questions asked need to be concise with minimal time required to complete. This will not only be more acceptable to the user community, but is likely to generate more reasonable models for analysis – thus more accurate results.
- The types of questions asked at the proposal stage should include various attributes that allow all parties to fully understand the risk areas involved. For example, IDI questions probe into the framework of the intended contract structure (including whether the project will be based on Time & Materials billing or on Fixed Price billing). IDI probes into the maturity of the proposed solution. IDI assesses the availability of skilled resources. IDI examines if the deal is being “rushed” and incorporates related risk accordingly into the statistical equations.
- Like IDI, a project health prediction metric should take into account the number of iterations the proposal has gone through. These are but a few of the considerations

that are factored into the IDI prediction of the project’s potential risk, success, or failure.

- The risk assessment should be completed by independent reviewers who are not directly invested in the outcome of the project, so that objective answers are obtained regarding the potential risks.

#### 6. ACKNOWLEDGEMENTS

We would like to thank the GBS risk management community for their help in understanding the prevalent risk management practices. The project had a number of executive sponsors over time who were responsible for funding, setting the goals of the project and also helping us to roll this analysis out into practice including Greg Dillon, Ralph Nelson, Doug Bunch, Sergey Makogon, Theresa Ell, Patric Shaffer, Fausto Bernardini, Sridhar Iyengar and Robert Morris.

#### 7. REFERENCES

- [1] Boehm, B. W. (1989), *Tutorial: Software Risk Management*, IEEE Computer Society Press.
- [2] Charette, R. N. (1989), *Software Engineering Risk Analysis and Management*, New York: McGraw-Hill.
- [3] Hall, E., M. (1997), *Managing Risk*, Addison Wesley, Reading, MA.
- [4] Dorofee, A. J., W. J. A. A. C. J. H. R. P. M. T. J. & Williams, R. J. (1996), *Continuous Risk Management Guidebook*, Software Engineering Institute, Pittsburgh, PA.
- [5] Addison, T. & Vallabh, S. (2002), Controlling software project risks: an empirical study of methods used by experienced project managers, in 'SAICSIT '02', pp. 128-140.
- [6] Freimut, B.; Hartkopf, S.; Kaiser, P.; Kontio, J. & Kobitzsch, W. (2001), An industrial case study of implementing software risk management, in 'ESEC/FSE-9: Proceedings of the 8th European software engineering conference', ACM, New York, NY, USA, pp. 277-287.
- [7] Kasap, D. & Kaymak, M. (2007), Risk Identification Step of the Project Risk Management, in 'Proc. Portland International Center for Management of Engineering and Technology', pp. 2116-2120.
- [8] Sun, A. & Li, C. (2007), Research on Project Risk Evaluation Method Based on Markov Process, in 'Proc. International

- Conference on Wireless Communications, Networking and Mobile Computing WiCom 2007', pp. 5293--5296.
- [9] Hu, Y.; Huang, J.; Chen, J.; Liu, M. & Xie, K. (2007), Software Project Risk Management Modeling with Neural Network and Support Vector Machine Approaches, *in* 'Proc. Third International Conference on Natural Computation ICNC 2007', pp. 358--362.
- [10] Jianyi, G.; Li, Z.; Xusheng, L.; Yuejuan, H. & Zhengtao, Y. (2008), Implementing a quantitative-based methodology for project risk assessment DSS, *in* 'Proc. 27th Chinese Control Conference CCC 2008', pp. 730-734.
- [11] Amram, M. & Kulatilaka, N. (2000), *Real Options: Managing Strategic Investment in an Uncertain World*, Harvard Business School.
- [12] Keil, M., M. J. & Arun, R. (2000), 'Why software projects escalate: An empirical analysis and test of four theoretical models', *MIS Quarterly* 24(4), 631-664.

# OCL Evaluation on AUTOSAR Model

Sachoun Park<sup>1</sup>, Taeman Han<sup>1</sup>, Hyoungju Lim<sup>2</sup>, and Gihwon Kwon<sup>2</sup>

<sup>1</sup>Electronic and Telecommunications Research Institute,  
138, Gajeongno, Yuseong-Gu, Daejeon, Korea  
{sachem, tmhan}@etri.re.kr

<sup>2</sup>Department of Computer Science, Kyonggi University,  
San 94-6, Yiui-Dong, Youngtong-Gu, Suwon-Si, Kyonggi-Do, Korea  
{limehome01, khkwon}@kgu.ac.kr

## Abstract

*To overcome the increasing complexity of automotive software, it is inevitable to employ a standardized software platform. AUTOSAR, the automotive software platform, was proposed and developed by the European automotive industry. In the AUTOSAR documents, meta-models and its well-formedness rules were described by a natural language, so modelers apt to violate constraints of meta-model. In this paper, to help automotive engineers to reduce mistakes during the design phase, we propose the method of checking well-formedness rules against AUTOSAR model with OCL evaluation.*

**Keywords** : AUTOSAR, Meta-model, Well-formedness rule, OCL, XML Schema, EMF.

## 1 Introduction

Traditionally, the software has played an auxiliary role in automotive industry to support hardware products. However the increasing demand for sophisticated features in today's vehicles has led to the development of complex software and electronic system. To overcome the ever increasing complexity of automotive software, it is inevitable to employ a standardized software platform. AUTOSAR(AUTomotive Open System Architecture) is the well known automotive software platform from the European automotive industry. AUTOSAR is a partnership of automotive manufacturers and suppliers working together to develop and establish a de-facto open industry standard for automotive architecture. The driving forces

for the intended standardization can be enable detection of errors in early design phases[1].

To define the syntax and semantics of the language of AUTOSAR model, that is the AUTOSAR meta-model, UML profile was defined in [2]. The profile describes basic concepts that should be used when creating content of the meta-model. The meta-model of software components are described in [3]. However, the well-formedness rules of the meta-model were specified by a natural language, so ambiguities of model constraints allows an automotive engineer to make some errors in his or her model.

In this paper, we propose the method and tool for checking well-formedness rules to be used in an early design phase. The AUTOSAR meta-model is provided by the form of XML schema[4] and the user model from an authoring tool is also expressed by XML files. Firstly, we transformed AUTOSAR meta-model into AUTOSAR EMF(Eclipse Modeling Framework)[5] model and then translate the user XML file to the instance of the EMF model. Secondly, we extract OCL(Object Constraint Language)[6] constraints as well-formedness rules from the specification document. Finally, the OCL constraints are evaluated on the EMF instance corresponding to the automotive software model.

This paper is organized as follows: background such as OCL and EMF are provided in Section 2. Section 3 gives a brief overview of AUTOCL tool and the explanation of process for the extracting OCL constraints from AUTOSAR templates and some problems for checking OCL on an EMF instance which has tree like shape in the implementation point of view. Finally, conclusions are discussed in Section 4.

---

This work was supported by the GRRC program of Gyeonggi province.[200911921, Development of Secure Web Software and Content Service Technology]

## 2 OCL and EMF

The AUTOSAR meta-model is specified by UML class diagrams. A UML class diagram is typically not refined enough to provide all the relevant aspects of a specification. There is a need to describe additional constraints about the objects in the model. Such constraints are often described in natural language. Practice has shown that this will always result in ambiguities. In order to write unambiguous constraints, so-called formal languages have been developed. OCL is a formal language used to describe expressions on UML models. These expressions typically specify invariant conditions that must hold for the system being modeled or queries over objects described in a model.

Eclipse Modeling Framework (EMF) is an Eclipse-based modeling framework and code generation facility for building tools and other applications based on a structured data model. From a model specification described in XML, EMF provides tools and runtime support to produce a set of Java classes for the model, a set of adapter classes that enable viewing and command-based editing of the model, and a basic editor. As shown in figure 1, EMF unifies the three important technologies: Java, XML, and UML. Most important of all, EMF provides the foundation for interoperability with other EMF-based tools and applications[5].

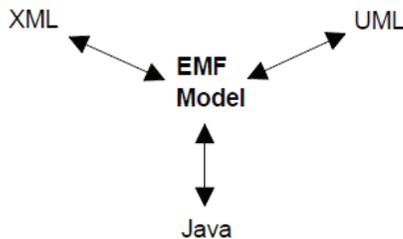


Figure 1. EMF unifies Java, XML, and UML

The Eclipse infrastructure for modeling is based on EMF, with support for OCL 2.0 provided by the Model Development Tools (MDT) project[7]. The combined expressive power of Ecore and OCL allows capturing an amount of constraints in a declarative manner. These constraints as well-formedness rules cannot be checked by XML Schema validation.

## 3 Checking Well-Formedness Rules

AUTOCL is a well-formedness checking tool which is Java plug-in developed with Eclipse RCP Ganymede. In the figure 2, it has three inputs: OCL Constraints,

AUTOSAR EMF model, and EMF instance, corresponding to well-formedness rules, AUTOSAR meta-model, and user model, respectively. OCL constraints are extracted from Software Component Template Specification which has many templates for classes in AUTOSAR meta-model. AUTOSAR meta-model is provided by the form of XML Schema that can be translated into EMF model. And the given user model also is expressed by the form of XML file. To check the user model's well-formedness, we transform XML file from user model into EMF instance using XML parsing APIs.

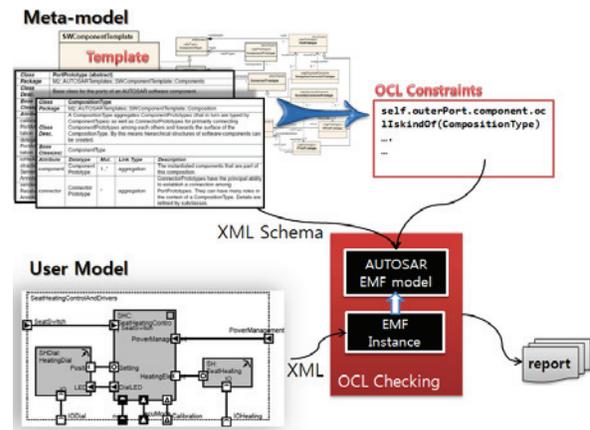


Figure 2. The overview of AUTOCL

### 3.1 Extracting Constraints from Templates

In the AUTOSAR SW-C(Software Component) Template document, the meta-model is described by UML Class diagrams and templates. For each class, one template is provided like a figure 3 and figure 4.

With carefully reviewing the below template, we can catch four syntactic constraints:

- ① The type of provider is PPortPrototype.
- ② The type of requester is RPortPrototype.
- ③ Assembly connector has exactly one provider and exactly one requester.
- ④ The provider and the requester linked on the same connector must reside in the same composition.

First three constraints are so trivial and can be checked by XML Schema validation. However, in the case of the last constraint, it is impossible to check its satisfaction with XML validation technique. Therefore we have to rewrite the constraint to an OCL expression to check the validity of the well-formedness rule. When converting the constraint to OCL, firstly the context of the expression is indicated, and then according to the context, OCL expression is written carefully like below.

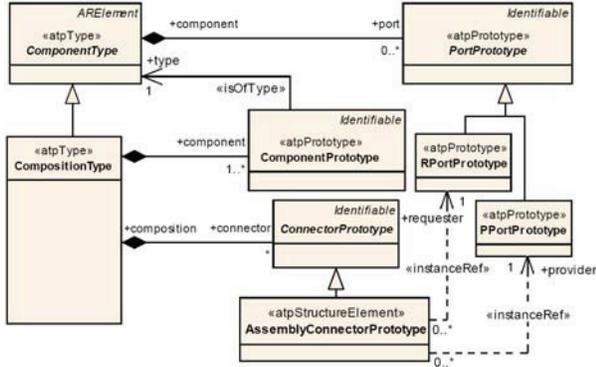


Figure 3. A part of a meta-model in AUTOSAR SW-C

<b>Class</b>	AssemblyConnectorPrototype				
<b>Package</b>	M2::AUTOSARTemplates::SWComponentTemplate::Composition				
<b>Class Desc.</b>	AssemblyConnectorPrototypes are exclusively used to connect ComponentPrototypes in the context of a CompositionType. ④				
<b>Base Class(es)</b>	ConnectorPrototype				
<b>Attribute</b>	<b>DataType</b>	<b>Mul</b>	<b>Link Type</b>	<b>Description</b>	
provider	PPort Prototype	1	instanceRef	Instance of providing port. ①	
requester	RPort Prototype	1 ③	instanceRef	Instance of requiring port. ②	

Figure 4. The template of Assembly Connector Prototype

```

context CompositionType inv inTheSameCp:
  let Ass : Set(AssemblyConnector) =
    self.connector->select(k|k.oclIsKindOf
      (AssemblyConnector))
  in
  Ass -> forAll(s|
    self.component->exists(c|c.type.port->
      includes(s.provider))
  and
  self.component->exists(c|c.type.port->
    includes(s.request))
  )

```

Examining the SW-C Template document, we can find 520 syntactic constraints from 118 classes in the meta-models. Table 1 show results of our examination for the SW-C Template. About 40 constraints from classes' description, after cautiously reviewing those constraints, we can conclude that only 17 among 40 constraints need to be converted to OCL expression, and 12 constraints of the rest could be checked XML Schema validation, and 11 constraints are semantic constraints which cannot be dealt with our method yet.

Type of constraints	The number of constraints
Class inheritance relation	118
Type of properties	181
Multiplicity of properties	181
From classes' description	40
Total	520

### 3.2 Evaluating OCL expression on EMF Instance

Now, we explain some implementation features of our tool. Because the AUTOCL is compliant to Model Persistence Rules for XML[8], it is independent of the change of AUTOSAR meta-model. Figure 5 describes the model persistence rules for XML in the overall context. The meta-levels of the AUTOSAR modeling approach are described on the left side of the image. The syntax and semantics of the language UML2.0 is described on the meta-meta-level (M3), the AUTOSAR meta-model is a UML model that defines the language for describing AUTOSAR systems, and an AUTOSAR model is an instance of the AUTOSAR meta-model.

The meta-levels of the XML language are described on the right side of Figure 5. The W3C XML schema specification defines how a W3C XML schema can be defined. The AUTOSAR XML schema is a W3C XML schema that defines the language for exchanging AUTOSAR models. Finally an AUTOSAR XML description describes the XML representation of an AUTOSAR model.

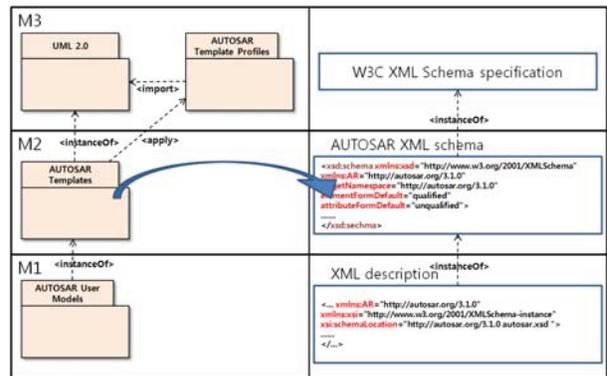


Figure 5. Context of Model Persistence Rules of XML

For OCL evaluation in AUTOCL, the naming of OCL expressions is changed on accordance of the AUTOSAR XML schema which is written in upper-case letters. In order to increase the readability of the XML names, hyphens are inserted in the XML names which separate parts of the names. Below example shows the translation of the UML names to XML names:

TestECUClass12ADC → TEST-ECU-CLASS-12-ADC

If the default mapping is not suitable, the XML name can be explicitly defined by specifying the tagged value 'xml.name' for the corresponding UML model element. And the EMF instance generated by user model which is XML file has the Tree-like Structure in AUTOCL,, while the interpretation of an OCL expression is based on graph structure. To overcome this problem we use linking method.

References between meta-classes are represented through XML-elements suffixed by <...-REF> or <...-TREF>. Referenced XML-descriptions must define a <SHORT-NAME> which must be unique within its namespace. The referencing mechanism via <SHORT-NAMES> doesn't require all referenced elements to be available in one document. Furthermore the <SHORT-NAME> referencing mechanism provides the concept of namespaces. Therefore the value of <SHORT-NAME> is not required to be globally unique and "speaking identifiers" can be used. Thus a XML-description can be referenced by specifying an absolute path. Absolute paths are composed of sequences of <SHORT-NAME>'s separated by "/". The following rules apply to SHORT-NAME paths used in AUTOSAR:

- 1) An absolute path is calculated by collecting the SHORT-NAMES of the model.
- 2) Absolute paths begin with the character "/".
- 3) No relative paths are allowed.

Figure 6 shows a simple example and their XML representation of reference between class D and the abstract class B.

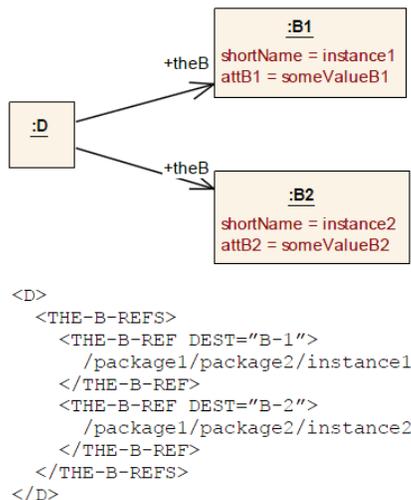


Figure 6. Reference and its XML representation

To resolve the linking problem, we use the algorithm such as blow box. This algorithm explores the abstract syntax tree of a given OCL expression. When a terminal node labeled with <...REF> of <...TREF> is reached, the referencing object can be obtained by following function.

```

// o is the current object
// q is a queue storing values of SHORT NAME

Object refer(Object o, Queue<String> q){

    if(q.size()==0) return o;
    String shortName = q.pick();
    o = searchObject(o, shortName);
    o = refer(o, q);
    return o;
}

```

## 4 Conclusions

AUTOSAR is an open and standardized automotive software architecture, jointly developed by automobile manufacturers, suppliers and tool developers. Many tools which are compliant with AUTOSAR are developed[9]. One of main functionality of these tools is checking whether user model is compliant to AUTOSAR meta-model or not, but, to the best of our knowledge, no OCL syntax checker was made. OCL has enough expressiveness power to represent well-formedness rules that cannot be checked with XML Schema validation. To build more accurate model, this kind of syntax checker is needed.

AUTOCL has three important benefits: first, it can interact with almost all AUTOSAR modeling tool because it is developed as Eclipse plug-in using EMF and its input as user model is XML file. Second, due to its compliance with Model Persistence Rules for XML, it is not influenced from any changes of AUTOSAR meta-model. Finally, because the traceability between extracted OCL constraints and the specification document are obvious, it is ease to manage the changeability of constraints.

## References

- [1] AUTOSAR GbR, Technical Overview, 2008.
- [2] AUTOSAR GbR, Template UML profile and Modeling Guide, 2008.
- [3] AUTOSAR GbR, Software Component Template, 2009.
- [4] [http://www.autosar.org/download/R4.0/AUTOSAR\\_MM\\_OD\\_XMLSchema.zip](http://www.autosar.org/download/R4.0/AUTOSAR_MM_OD_XMLSchema.zip)
- [5] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, Eclipse Modeling Framework, 2/E, Addison-Wesley Professional, 2009.
- [6] OMG. Object Constraint Language Specification, OMG Available Specification Version 2.0, 2006.
- [7] <http://www.eclipse.org/modeling/mdt/>
- [8] AUTOSAR GbR, Model Persistence Rules for XML, 2009.
- [9] IBM, AUTOSAR System Modeling in IBM Rational Systems Developer, 2008. [http://www.ibm.com/developerworks/rational/library/08/0812\\_paul-rooks/index.html](http://www.ibm.com/developerworks/rational/library/08/0812_paul-rooks/index.html)

# Intertwining Implementation with the RealSpec Executable Real-Time Specification Language

Amir A. Khwaja  
Intel Corporation  
Austin, TX 78746  
[amir.a.khwaja@intel.com](mailto:amir.a.khwaja@intel.com)

Joseph E. Urban  
Department of Computer Science  
Texas Tech University  
Lubbock, TX 79409-3104  
[joseph.urban@ttu.edu](mailto:joseph.urban@ttu.edu)

## Abstract

*Software development is evolutionary in nature. One suggested approach is to incrementally evolve specifications into implementation by intertwining implementation with specifications. This paper presents an evolutionary method by intertwining RealSpec real-time specifications and corresponding C programming implementation that supports joint execution of specifications and intertwined program code.*

## 1. Introduction

In their research of twenty nine companies, MacCormack, et al. analyzed the data from completed projects and identified that successful development was evolutionary in nature [1]. Companies first would release a low-functionality version of a product to selected customers at an early stage of development which would then evolve iteratively based on customers' feedback. Although the evolutionary model has been around for several years, MacCormack, et al. demonstrated with strong data that there is a connection between the practices that support the model and the quality of the resulting product [1].

Swartout and Balzer pointed out that since the developed systems use physical devices that have limitations, these limitations introduce either a restriction that limits the domain of input or introduce the possibility of error [2]. Another issue is that the developers are unable to foresee all the implications and interactions in complex systems that are usually exposed during implementation. As such, Swartout and Balzer proposed an approach of intertwining implementation with specification [2]. Minkowitz, et

al. indicates that an efficient development process consists of activities where an implementation can be smoothly derivable from a specification, reducing the chances of errors occurring in the implementation [3].

This paper describes a method for evolutionary software development using RealSpec by intertwining RealSpec real-time specifications and corresponding C code that can be jointly executed. The remainder of the paper is organized in the following sections. Section 2 provides an overview of the RealSpec real-time specification language. Section 3 discusses intertwining concepts. Section 4 presents intertwined compilation process. Section 5 demonstrates an example. Section 6 compares related work. Section 7 summarizes the paper.

## 2. RealSpec Specification Language

RealSpec [4, 5, 6] is a declarative and executable specification language for specifying real-time and concurrent software systems. The statements in a RealSpec specification are equations defining streams and filters. For example, the following RealSpec specification defines  $x$  (the output) to be the data stream  $\langle 1, 2, 3, 4, 5, \dots \rangle$  at time index  $\langle t_0, t_1, t_2, t_3, t_4, \dots \rangle$ , respectively, an infinitely varying sequence:

```
x
  where {
    x = 1 fby x+1;
  }
```

In the above specification, the binary operator **fby** (followed by) provides abstract iteration over sequences.

A RealSpec specification starts with a system construct that provides a context for the rest of the specification.

```
system s {
  resources { ... }
  processes { ... }
  functions { ... }
}
```

A process object is defined using a process construct. The functions in a process are declarative assertions. All of the functions of a process execute simultaneously and synchronously. A process has a single execution thread by default, but may have as many threads as possible, defined as part of a process definition. In the example below, `x` gets the value of `x+1` if the executing thread is `th1`, indicated by the property `pid`, otherwise `x` gets the value of `x*2`.

```
process p() threads th1, th2 {
  x
  where {
    x = 1 fby if pid == 0 then x+1
      else x*2;
  }
}
```

The processes communicate with each other via message passing by using a pair of send and receive thread functions.

```
process p1() {
  ... p2.send(data) @tout 50 us; ...
}
process p2() {
  ... x = p1.receive() @tout 75 us; ...
}
```

The RealSpec compiler has a two stage compile process. A RealSpec specification is first compiled into an equivalent C code. The generated C code is then compiled for an Intel architecture based embedded platform running the Embedded Configurable Operating System (eCos) 3.0 RTOS [7].

### 3. Intertwining Concepts

In order to support the evolutionary development process, RealSpec's dataflow and C language's control flow models need to work in an intertwined manner. From a functional standpoint, the two models are similar in that both map a set of input data to output data. The difference is in the way output is computed from a given set of input. In the case of the dataflow model, the input data flows through a set of fixed functional nodes that transform the data until the final output is achieved. In the case of control flow, the data remains in fixed storage locations where command control flows through various storage locations updating and transforming the stored data until the final

output data is achieved. For example, for a RealSpec dataflow specification that outputs the 10th approximation to an input number `n`'s square root:

```
aprx asa c eq 10
where {
  aprx = 1 fby (aprx + first n/aprx)/2;
  c = 1 fby c+1;
}
```

The corresponding C function may be as follows:

```
void calcApprox()
{
  ...
  while(1) {
    aprx = 1.0;
    scanf("%d", &n);
    for(int c = 0; c < 10; c++) {
      aprx = (aprx + n/aprx)/2.0;
    }
    printf("%f\n", aprx);
  }
}
```

Both the RealSpec specification and the corresponding C program would be accepting a series of input numbers, and outputting the 10<sup>th</sup> approximation of their square roots. Hence, intertwining C programming based modules with RealSpec's dataflow modules can be achieved at the module level.

### 4. Intertwined Compilation

The intertwined compilation of RealSpec specifications with C code requires the following considerations for the RealSpec compiler: (a) a mechanism to indicate to the compiler intertwined code segments versus specification segments, (b) code requirements for the developer based on the eCos model to ensure code replaced by the developer is consistent with the code generated by the compiler, and (c) a mechanism to supply missing parse tree information to the compiler related to the replaced code for code generation.

**Code and Specification Segments:** The intertwined code and data segments are indicated to the compiler by adding C code in special enclosure sections. The RealSpec compiler takes the C code and declarations within the enclosures and copies the code as is into the generated target output C language file.

**Code Replacement Requirements:** Implementation code is expected to follow the eCos model with some of the following requirements:

- Each process or resource construct replaced with corresponding code needs to have the following:

- A RealSpec process or resource specific C **struct** needs to be defined with predefined process or resource variables, as well as any user defined variables.
- A global instance of the RealSpec process or resource specific C struct needs to be declared.
- An initialization function for the RealSpec construct needs to be defined that should initialize construct variables as defined in the construct structure as well as create a message box for inter-process communication.
- If the replaced construct is a process construct, then a main function for the process needs to be defined that is based on the RealSpec process definition.
- The system construct replaced with corresponding code needs to have the following:
  - Thread specific global variables need to be declared in the **declaration\_section** for each process construct.
  - An eCos required application specific main function, **cyg\_user\_start()**, needs to be defined that is called by eCos. This function calls the initialization functions for each process and/or resource construct in the specification, as well as creates and starts eCos threads for each RealSpec thread.
  - All resources declared within the **resources** block need to be globally declared in the **declaration\_section**.
  - If the abort function is defined in the system construct, then a corresponding function needs to be implemented as an eCos application exception handler.

**Map File for Missing Parse Tree Information:** An implementation map file is created to supply missing parse tree information to the code generator for the RealSpec constructs converted into implementation code in the intertwined system. The map file consists of pairs of keywords and values for various types of information needed by the code generator.

## 5. An Intertwined Example

This section uses the following producer-consumer example to demonstrate the intertwined development process.

```
system sys {
    processes {
```

```
        producer; consumer;
    }
}
process producer() {
    int in;
    consumer.send(in)
    where { n = 1 fby in+1; }
}
process consumer() {
    producer.receive();
}
```

Assume that the first step in the development process is that the consumer process is replaced with its corresponding code. The resulting intertwined specification and code along with the map file may be as follows using the rules outlined in the previous section:

```
declaration_section {
    struct consumer_struct {
        int priority, pid;
        float stime, etime;
        cyg_handle_t mbox_handle;
        cyg_mbox mbox;
    };
    struct consumer_struct consumer_instance;
}

system sys {
    processes { producer; consumer; }
}

process producer() {
    int in;
    consumer.send(in)
    where { in = 1 fby in+1; }
}

code_section {
    void consumer_init()
    {
        consumer_instance.pid = 0;
        cyg_mbox_create(
            &consumer_instance.mbox_handle,
            &consumer_instance.mbox);
    }
    void consumer_func1()
    {
        int* result = 0;
        while(1) {
            result = (int*)cyg_mbox_get(
                consumer_instance.mbox_handle);
            if(result) {
                printf("consumer: %d\n",*result);
            }
            else {
                printf("consumer: no result\n");
            }
        }
    }
}

producer.im:
PROCESS_CONSTRUCT = consumer;
MAIN_FUNC(consumer) = consumer_func1;
```

In the following increments, the producer process and the system construct may also be replaced with its corresponding code. The declaration and code section

markers can be removed. The map file is also not needed as well.

## 6. Comparative Analysis

The intertwined development approach may be evaluated using the following criteria: (a) joint execution of intertwined specification and code should be possible, (b) clear distinction of intertwined code from specification without blurring into each other, and (c) real-time features support.

IRPE [8] and Choppy's framework [9] provide true intertwining of implementation with specification where algebraic specifications are progressively replaced with implementation modules allowing mixed evaluation at each step. Both of these approaches also provide a clear demarcation between specification and implementation by using corresponding notations. However, both of these specification languages do not support real-time features. Gist [10], Estelle [11], LOTOS [12], MHSC [13], and SpecCharts [14] allow incremental refinement of specifications, but no direct implementation intertwining is supported. Estelle, LOTOS, MHSC, and SpecCharts, however, do support some form of real-time capability. Gypsy [15] and C++ Spec. Library [3] methods are really programming languages with some form of specification support in the form of assertions. RealSpec, on the other hand, provides full support for all three criteria.

## 7. Summary

The paper presented an evolutionary development approach by intertwining C code with RealSpec real-time specifications. Intertwining between dataflow and control flow models was achieved at the module level. Various rules were presented to convert RealSpec specifications incrementally into eCos based C code. A producer-consumer example was used to demonstrate the applicability of the concepts. Future directions of the research include language and processor advancements.

## References

- [1] A. MacCormack, R. Verganti, and M. Iansiti, "Developing Products on "Internet Time": The Anatomy of a Flexible Development Process," *Management Science*, Vol. 47, No. 1, Jan 2001, pp. 133-150.
- [2] W. Swartout and R. Balzer, "On the Inevitable Intertwining of Specification and Implementation," *Communications of the ACM*, Vol. 25, No. 7, Jul 1982, pp. 438-440.
- [3] C. Minkowitz, D. Rann, and J. H. Turner, "A C++ Library for Implementing Specifications," *Proc. of the 1995 Workshop on Industrial-Strength Formal Specification Techniques*, Apr 5-8, 1995, pp. 61-75.
- [4] A. A. Khwaja and J. E. Urban, "RealSpec: An Executable Specification Language for Prototyping Concurrent Systems," *Proc. of the 19<sup>th</sup> IEEE/IFIP Intl. Symp. on Rapid System Prototyping*, Jun 2-5, 2008, pp. 3-9.
- [5] A. A. Khwaja and J. E. Urban, "RealSpec: An Executable Specification Language for Modeling Resources," *Proc. of the 20<sup>th</sup> Intl. Conf. on Software Eng. and Knowledge Eng. (SEKE 2008)*, Jul 1-3, 2008, pp. 97-102.
- [6] A. A. Khwaja and J. E. Urban, "Timing, Precedence, and Resource Constraints in the RealSpec Real-Time Specification Language," *Proc. of the 2008 IASTED Intl. Conf. on Software Eng. and Applications*, Nov 16-18, 2008, pp. 192-198.
- [7] A. J. Massa, *Embedded Software with eCos*, Prentice Hall, New Jersey, 2003.
- [8] M. B. Ozcan and J. Siddiqi, "Interchanging Specifications and Implementations in Evolutionary Prototyping," *Software - Practice and Experience*, Vol. 26, No. 9, Sept 1996, pp. 999-1023.
- [9] C. Choppy and S. Kaplan, "Mixing Abstract and Concrete Modules: Specification, Development and Prototyping," *Proc. of the 12<sup>th</sup> Intl. Conf. on Software Eng.*, Mar 26-30, 1990, pp. 173-184.
- [10] M. S. Feather, "Language Support for the Specification and Development of Composite Systems," *ACM Trans. on Programming Lang. and Sys. (TOPLAS)*, Vol. 9, No. 2, Mar 1987, pp. 198-234.
- [11] ISO 9074, *Estelle - A Formal Description Technique Based on an Extended State Transition Model*, 1989.
- [12] ISO 8807, *LOTOS - A Formal Description Technique Based on the Temporal Ordering of Observational Behavior*, 1988.
- [13] Y. Jing, H. Zhijun, W. Zhaohui, L. Jianguyun, F. Weicheng, and X. Zhaohui, "A Methodology for High-Level Software Specification Construction," *ACM SIGSOFT Software Eng. Notes*, Vol. 20, No. 2, Apr 1995, pp. 48-54.
- [14] F. Vahid, S. Narayan, and D. D. Gajski, "SpecCharts: A VHDL Front-End for Embedded Systems," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Sys.*, Vol. 14, No. 6, Jun 1995, pp. 694-706.
- [15] A. L. Ambler, D. I. Good, J. C. Browne, W. F. Burger, R. M. Cohen, C. G. Hoch, and R. E. Wells, "Gypsy: A Language for Specification and Implementation of Verifiable Programs," *Proc. of the ACM Conf. on Lang. Design for Reliable Software*, Mar 28-30, 1977, pp. 1-10.

# Self-Management of Component Executors for Robot Applications

Michael E. Shin, Hemanth Thimme Gowda,  
Taeghyun Kang  
*Department of Computer Science*  
*Texas Tech University*  
*Lubbock, TX 79409-3104*  
*806-742-3527*  
*{michael.shin, hemanth.gowda,*  
*th.kang}@ttu.edu*

Sunghoon Kim, Seungwook Jung,  
Choulsoo Jang, Byoungyoul Song  
*Intelligent Robot Control Research Team*  
*ETRI*  
*161, Gajeongno, Yuseong-gu, Daejeon, 305-*  
*700, Korea,*  
*+81-42-860-6767*  
*{saint, sejung, jangcs, sby}@etri.re.kr*

## Abstract

*This paper describes a self-managing mechanism for failed component executors in a component execution engine for robot applications. Each component executor for robot applications should execute all the assigned components periodically within a defined cycle time. A component executor falls into a failure if it does not meet the cycle time requirements for the assigned components. To handle the failures of component executors, this paper describes the design of a self-managing mechanism, which includes detection and self-managing algorithms. The designed self-managing mechanism is implemented as a prototype before it is integrated with a component execution engine. The performance of the algorithms has been analyzed using the prototype.*

## 1. Introduction

One of robot application domain features is that many components in robot application systems need to be executed periodically in order to capture robot environmental situation using multiple sensors and to respond to the sensed input using several actuators. As a number of different robot applications run together, more components come to have the same cycle time to be executed periodically. However, the execution time of each component is unknown because it varies depending on different platforms to run the component, devices and actuators interacted with the component, or its dependent components. Robot application developers can define only the required cycle time of components for robot applications if a component needs to be executed periodically.

Under the circumstances of uncertain execution times of components, how many same cycle-timed components are assigned to a thread would be a policy to be determined in the platform or middleware to run robot application components. All the components with the same cycle time may be assigned to each different thread to execute, but this approach may generate too many

threads that make the platform or middleware difficult to manage those threads – eventually all the resources may be dried. On the other hand, all the same cycle-timed components may be executed by just one thread, but the thread may not execute all the components within a cycle time. Even though each reasonable number of same cycle-timed components can be assigned to each different thread, this also may meet a problem if some of the components take more processing time than the cycle time.

The Open Platform for Robotic Service (OPRoS) [Song08] that is a container-based component execution engine takes a policy that assigns all the same cycle-timed components to a thread, referred to as a component executor. This policy was decided and implemented before a project for self-managing component executor started, so the policy could not be changed. When some components assigned to a component executor take too much time to process, other components executed by the same component executor cannot be executed within the cycle time. Sometimes this may lead the OPRoS component execution engine to a critical failure. Even though there is no fault [Torres-Pomales00] in a component or OPRoS component execution engine, the application may come across a failure.

## 2. Related Work

Considerable researches have been suggested to self-heal or self-manage [IBM03, Kramer07] failures of components or systems. Some approaches [Oriezy99, Dashofy02, Garlan03, Kim04] use the software architecture for self-healing software systems in which the knowledge of self-healing, separated from the application system, is encapsulated in the centralized software architecture. The self-healing mechanism is performed at the level of centralized software architecture. In some approaches [Shin05, Shin09], the knowledge of self-management is localized in each component and connector between the components. The self-managing mechanism is performed by each localized component and connector between the components.

Several works have been done in the field of robotics for component-based robot software development. Robotic Technology Component (RTC) [OMG06] is the OMG (Object Management Group) robot software component model, which can introspect components, ports, and connections between ports at runtime. Open Robot Control Software (OROCOS) [Soetens07] is initiated with an open source project for robot control software development, allowing application designers to build highly configurable and interactive component-based real-time control applications. Middleware for Autonomous Mobile Robots (MIRO) [Enderle01] is a distributed object-oriented framework for mobile robot control using CORBA (Common Object Request Broker Architecture) technology. It does not support real-time application development, but supports distributed computing and parallel processing. Open Platform for Robotic Service (OPRoS) [Song08] is a platform for network based intelligent robots, which provides a software component model for supporting reusability and compatibility of the robot software component in the heterogeneous communication network. However, these approaches for component-based robot software development do not have clear capabilities for self-managing failures.

One of the earlier works related to self-management of component executors in this paper is the process migration [Smith01] implemented in operating systems over a distributed system environment. A process is migrated from one overloaded machine (system) to another machine (system) that has fewer loads. This approach helps in load balancing and increases efficiency. During the migration, the process does not execute, resuming the execution only after the migration is completed. Process migration focuses on prevention of process failure by means of load balance between distributed processes, whereas our approach recovers failed component executors (threads) executing components for robot applications.

### 3. Design of Self-Managing Mechanism for Executor

#### 3.1 Software Architecture for Self-Managing Executors

The software architecture for self-managing executors in the OPRoS component execution engine is designed as a layered software architecture, structured into the service layer and self-management layer. The service layer contains executors processing components periodically within the defined cycle times, whereas the self-management layer monitors and self-manages failures of the executors. Figure 1 depicts executors in the

service layer, and the monitor and executor repair manager in the self-management layer. The monitor is composed of the monitor listener and monitor handler detecting executors violating the cycle time requirements, whereas the executor repair manager self-manages failed executors:

- **Executor Monitor Listener.** The Executor Monitor Listener receives notification messages from the executors and extracts component execution information from the messages. A component executed by an executor is compared with the current component of the executor in the Time Table that is constructed from the Component Sequence Table. The Component Sequence Table has a list of all the executors and the components assigned to each executor.
- **Executor Monitor Handler.** The Executor Monitor Handler periodically scans the Time Table for detecting any cycle time violations of executors. A violation is found when the total execution time for all the components assigned to an executor is more than the cycle time.
- **Executor Repair Manager.** Upon the notification from the Executor Monitor Handler, the Executor Repair Manager splits the components in the failed executor to a new executor based on the self-management policy for the OPRoS component execution engine. Failed executors are self-managed sequentially in an order that failures in executors are detected by the Executor Monitor Handler.

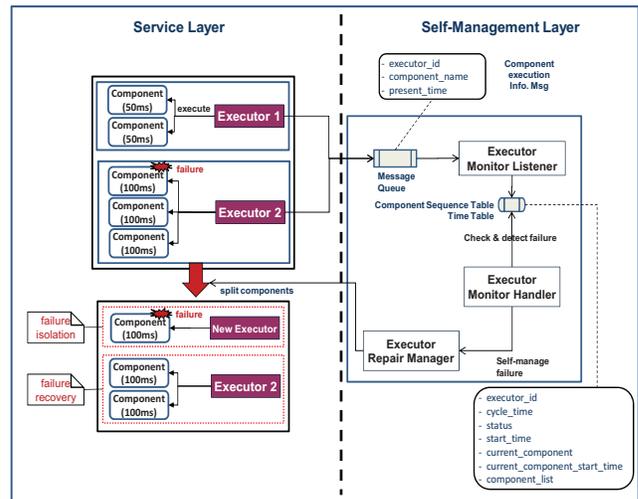


Figure 1. Software Architecture for Self-Management of Executors

#### 3.2 Self-Managing Mechanism for Executors

Each executor is monitored to detect the cycle time violation of components executed with the same cycle time, and any violation is self-managed in the next cycle

time. When an executor finishes processing a component, it notifies the monitor that checks whether the components assigned to the executor can be finished completely within the cycle time. If an executor does not finish processing all the assigned components within a cycle time, the components assigned to the executor are split and reassigned to a new executor.

The self-management policy for the OPRoS component execution engine addresses how a failed executor is self-managed in terms of split of components assigned to a failed executor. An executor executes the components periodically in a sequential order assigned to the executor. An executor may meet a failure either at the first component whose execution time exceeds the cycle time or at the middle of component sequence at which the sum of execution times of executed components is greater than the cycle time.

When the execution time of the first component of an executor is over the cycle time, the thread assigned to the component by the executor is preempted and the component is isolated from the other components of the executor. The isolated component is assigned to a new executor so that it is executed using the thread of the new executor again. The remaining components are executed by the current executor.

When an executor meets a failure at a component (referred to as a failed component) in the middle of the component sequence, all the components before the failed component are split from the other components of the failed executor and are assigned to a new executor. It is guaranteed that the components assigned to a new executor are successfully executed within their cycle times periodically. This is because the total execution time of those components is less than the cycle time. The remaining components including the failed component are still executed by the current executor. The failed component is not preempted because the failed component itself may finish its execution within the cycle time. The detection of failure is based on the sum of execution times of components from the first component to the failed component in the component sequence, but it is not only on the execution time of the failed component.

#### 4. Analysis of Self-managing Mechanism

A prototype for self-managing executors has been developed before the self-managing mechanism is integrated with the OPRoS component execution engine. The prototype is implemented in C++ programming language with the Boost library, which provides classes for creating and managing threads. The performance of self-managing mechanism developed in this paper is evaluated using the prototype so as to know how correctly the detection and self-managing algorithms encapsulated

in the Executor Monitor Handler and Executor Repair Manager work as designed.

As the detection algorithm in the Executor Monitor Handler continues to run, it judges falsely a normal executor as a failed executor before the normal executor fails to execute all the assigned components. This type error of the detection may result from several factors, such as the total number of executors running at the same time, the cycle time of each executor, the duration between the finishing time of executing all the components assigned to an executor in a cycle and the next cycle starting time, and machine speed to run the algorithm. However, our detection algorithm determines failures by means of only comparison between the current time and each cycle starting time of executors.

Figure 2 depicts the analysis results with just one executor, which executes five same cycle-timed components at two different cycle times (that is, 120 ms and 150 ms). This test has been carried out for 10 minutes (600,000 ms) in the same machine that has Intel Core 2 Duo T6400 with 2GHz and 3GB SDRAM. Each line in Figure 2 represents the times of splitting an executor to two executors when an executor meets a false failure. The percentages below the lines present the ratio of total execution time of five components to a cycle time for an executor. The 150 ms cycle-timed executor fails only at 85% and 90% ratios, whereas the 120 ms cycle-timed executor fails at 75%, 80%, 85% and 90%. We have also tested a 180 ms cycle-timed executor under the same condition. In this case, the executor does not meet a false failure within 10 minutes. Figure 2 indicates that the split time of an executor is affected by both the cycle time and the ratio of total execution time of components to the cycle time. The bigger the ratio of total execution time of components to a cycle time is, the earlier a split occurs. Given the same ratio of total execution time to a cycle time, a split takes place earlier as the cycle time of an executor is shorter.

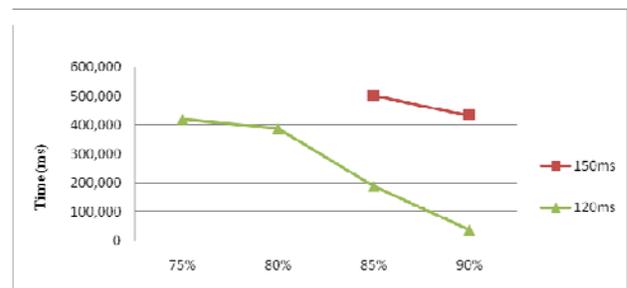
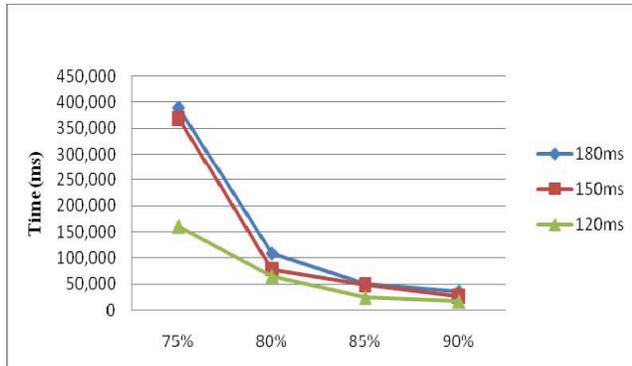


Figure 2. Analysis of Self-Managing Mechanism with one Executor

The number of executors running at the same time affects the performance of self-managing mechanism. Figure 3 depicts the split times of the shortest cycle-timed executor when five executors, each of which executes

five components, run together. In this test, given the fixed cycle times (1000 ms, 400 ms, 300 ms, and 200 ms) to four executors, only one executor has its different cycle times (120 ms, 150 ms, and 180 ms) per each test. Also the cycle time of each shortest cycle-timed executor and the total execution time of components assigned to the executor are the same as the first analysis in Figure 2. The comparison between Figures 2 and 3 indicates that, given the same ratio of total component execution time to a cycle time, the same cycle-timed (e.g., 120 ms) executor in Figure 3 splits earlier than that in Figure 2 as the number of executors increases.



**Figure 3. Analysis of Self-Managing Mechanism with Five Executors**

## 5. Conclusions

This paper has described the self-managing mechanism for failed component executors in the OPRoS component execution engine. A component executor meets a failure when it does not execute all the assigned components periodically within a defined cycle time. The software architecture for the self-managing mechanism has been designed separately from the OPRoS component execution engine. The designed detection and self-managing algorithms are implemented as a prototype before we integrate it with the OPRoS component execution engine. The performance of the algorithms has been analyzed using the prototype.

This paper can be extended to further research. An extension to this paper is that an executor, which may potentially fail into a failure, is split earlier before it fails. An executor may be split when it reaches a predefined percentage of each cycle time, such as 80% of a cycle time. On the other hand, as executors are getting stable, same cycle-timed executors may be combined to reduce the cost of system if the components assigned to the executors are executed by an executor within a cycle time.

## 6. Acknowledgement

This work was partly supported by the Industrial Foundation Technology Development Program of MKE/KEIT. [KI001800, Development of OPRoS (Open Platform for Robotic Services) Technology].

## 7. References

- [Dashofy02] Dashofy, E. M., Hoek, A. van der, and Taylor, R. N., "Towards Architecture-based Self-Healing Systems," Proceedings of the first workshop on Self-healing systems, Charleston, SC, November 18-19, 2002.
- [Enderle01] Enderle, S., Utz, H., Sablatnog, S., Simon, S., Kraetzschmar, G., and Palm, G., "MIRO: Middleware For Autonomous Mobile Robots," Telematics Applications in Automation and Robotics, 2001.
- [Garlan03] Garlan, D., Cheng, S., and Schmerl, B., "Increasing System Dependability through Architecture-based Self-repair," in *Architecting Dependable Systems*. R. de Lemos, C. Gacek, A. Romanovsky (Eds), Springer-Verlag, 2003.
- [IBM03] IBM, "An architectural blueprint for autonomic computing," IBM Autonomic Computing, April 2003.
- [Kim04] Kim, H. C., Choi, J., and Ko, Y., "Architecture-based Adaptive Software Systems for Sensor Networks," 3rd ACIS International Conference on Computer and Information Science, August 2004.
- [Kramer07] Kramer, J., and Magee, J., "Self-Managed Systems: an Architectural Challenge. Future of Software Engineering," Minneapolis, USA, May 23-25, 2007.
- [OMG06] OMG, "The Robot Technology Component Specification," November 2006.
- [Oriezy99] Oriezy, P., Gorlick, M.M., Taylor, R.N., Johnson, G., Medvidovic, N., Quilici, A., Rosenblum, D., and Wolf, A., "An Architecture-Based Approach to Self-Adaptive Software," In *IEEE Intelligent Systems* 14(3):54-62, May/June, 1999.
- [Shin05] Shin, M., "Self-Healing Component in Robust Software Architecture for Concurrent and Distributed Systems," *Science of Computer Programming*, Vol. 57, No. 1, 2005, pp 27-44.
- [Shin09] Shin, M., and Paniagua, F., "Design of Wrapper for Self-Management of COTS Components," *International Journal of Software Engineering and Knowledge Engineering*, Volume 19, Issue 4, 2009, pp. 529-551.
- [Smith01] Smith, J. M., "A Survey of Process Migration Mechanisms," A Technical report, Computer Science Department, Columbia University, 2001.
- [Soetens07] Soetens, P., "The OROCOS (Open Robot Control Software) Component Builder's Manual," Version 1.10.2, FMTC, 2007.
- [Song08] Song, B., Jung S., Jang, C., and Kim, S., "An introduction to Robot Component Model for OPRoS (Open Platform for Robotic Services)," *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, Venice, Italy, 2008.
- [Torres-Pomales00] Torres-Pomales, W., "Software Fault Tolerance: A Tutorial," NASA/TM-2000-210616, October 2000.

# A Multi-Agent Model for a Business Continuity Information Network

Lily Chang and Xudong He

School of Computing and Information Sciences, Florida International University

Miami, FL 33199, USA

Email: {lchan003, hex}@cis.fiu.edu

## Abstract

*In this paper, we apply a multi-agent modeling paradigm based on predicate transition nets (PrT nets) to model a Business Continuity Information Network (BCIN) for disaster mitigation. A two-level nested PrT net is defined to address a multi-agent architecture. We extend the semantic definitions of PrT nets and give the algorithms for modeling the interactions between nets at different level. The resulting BCIN net is translated to Process Meta Language (PROMELA) and run under SPIN. The objective of this work is to provide a dynamic model to study the interdependencies between distributed entities and resources in disaster mitigation.*

## 1. Introduction

Natural disasters such as hurricanes usually cause widespread damages and economical losses [11]. As such, how to build up a collaboration model among public and private sectors prior disasters, and how to ensure the dependability of a recovering process are critical issues in disaster mitigation. Disaster management has been one of the major application areas for multi-agent systems [14] due to its socially significant nature. In recent years, several projects based on multi-agent approach have been conducted to investigate various aspects in disaster management. For example, resource management after earthquake [2], emergency evacuation [10] and RoboCup rescue project [6]. These works were in the effort to build a simulation model to study the social behaviors after a major disaster. In this paper, we apply a multi-agent modeling approach based on predicate transition nets (PrT nets) [3] to model a Business Continuity Information Network (BCIN) [11], which aims to prepare private sectors for a rapid recovery after major disasters. At current stage, BCIN system supports static information sharing including (1) the advisories from public agencies, and (2) the resources provided by private sectors. Yet, the interdependencies between the recovery plans and resources from private sectors were not studied. The availability of resources is rather dynamic in the aftermath of a disaster, and affects the feasibility of recovery plans. The objective of this work is to provide a dynamic model to study the interdependencies of activities and the dynamics of resource consumptions in BCIN prior to deployment to ensure system dependability.

We chose PrT nets as the modeling language to model the controls and data flows in BCIN. PrT nets, a visual formalism, not only provide an intuitive way to study the interdependencies among distributed entities, but also provide a rigorous model to reveal inappropriate designs at an earlier stage of system development. We define a two-level nested net structure based on a two-layered multi-agent architecture, where the upper level net models the interdependencies between data tokens and agent tokens in a global view, while the lower level nets model heterogeneous agents. The interactions between upper and lower level nets are through a pair of synchronized transitions. The decomposition of global and local processes allows heterogeneous modules to be developed and analyzed independently, and to be reused as well. Although there were several approaches also adopted nets-within-nets paradigm [13] in modeling agent systems, their focus was on agent mobility [7, 15]. A few works related to the study of emergency response has adopted Petri nets as the modeling language [8, 9]; however, they used low level nets. There are several disadvantages of a flat low level net: (1) it has a tangling net structure, thus, is lack of the flexibility for further extensions; (2) it models only the controls that cannot handle important and meaningful data; and, (3) it provides only one level view with regard to global processes. On the other hand, our work has several features: (1) a nested net structure that decouples the global and local processes for modeling multi-agent systems; (2) the dynamics of nets as tokens to address agent autonomy and heterogeneity; and, (3) the synchronization controls of communication channels to address dynamic interactions in a multi-agent system. To show the feasibility of our modeling approach as well as to provide a dependable model, we translate the BCIN net into PROMELA [5] codes and execute the model for validation using SPIN tool.

The remainder of this paper is organized as follows. Section 2 gives the formal definitions of PrT nets and the extensions for modeling the interactions between agents and the system. Section 3 elaborates a case study including the algorithms for building nested nets and the execution of resulting BCIN net. The conclusion is drawn in Section 4.

## 2. Modeling a Two-Layered Multi-Agent System

Agents are heterogeneous or pre-existing entities in a multi-agent system context [14]. Accordingly, on top of

multiple agents, a well defined coordination model is usually needed in resource sharing. We define a two-level nested PrT net to address a two-layered multi-agent architecture, in which upper layer is considered as a mediator to accommodate autonomous agents at lower layer. In a two-level nested PrT net, lower level nets are tokens at upper level net. For coupling, we extend the constraint definitions of a transition to model the interactions between nets at different level. In the following sections, we first give a formal definition of PrT nets, and then introduce the extended notations.

## 2.1 Predicate Transition Nets

A PrT net is a tuple  $(N, Spec, ins)$ , where  $N = (P, T, F)$  is a net structure.  $P$  and  $T$  are finite sets of places and transitions of  $N$ , where  $P \cap T = \emptyset, P \cup T \neq \emptyset$  and  $F \subseteq (P \times T) \cup (T \times P)$  is a set of arcs, which define the flow relations.  $Spec$  is an algebraic specification, which includes sorts, operators, and equations. Terms defined in  $Spec$  include tokens in  $P$ , labels on  $F$  and constraints associated with  $T$ . An inscription  $ins = (\varphi, L, R, M)$  maps net elements to their denotations in the algebraic specification  $Spec$ .  $\varphi$  is a mapping from  $P$  to the set of sorts;  $L$  is a sort-respecting mapping from  $F$  to the set of labels;  $R$  is a mapping from  $T$  to the set of constraints; and  $M_0$  is the initial marking – a mapping from  $P$  to the set of tokens. The dynamic semantics of a PrT net can be defined as follows.

- (1) A marking of a PrT net is a mapping from  $P$  to sorts defined in  $Spec$ ;
- (2) An occurrence mode of  $N$  is a substitution  $\alpha = \{x_1 \leftarrow c_1, \dots, x_n \leftarrow c_n\}$ , which instantiates typed label variables. We use  $e:\alpha$  to denote the result of instantiating an expression  $e$  with  $\alpha$ , in which  $e$  can be either a label expression or a constraint;
- (3) Given a marking  $M$ , a transition  $t \in T$ , and an occurrence mode  $\alpha$ ,  $t$  is  $\alpha$ -enabled at  $M$  iff the following predicate is true:  $\forall p: p \in P. (\bar{L}(p,t):\alpha \subseteq M(p)) \wedge R(t):\alpha$ ; where

$$\bar{L}(x, y) = \begin{cases} L(x, y) & \text{if } (x, y) \in F \\ \emptyset & \text{otherwise} \end{cases}$$

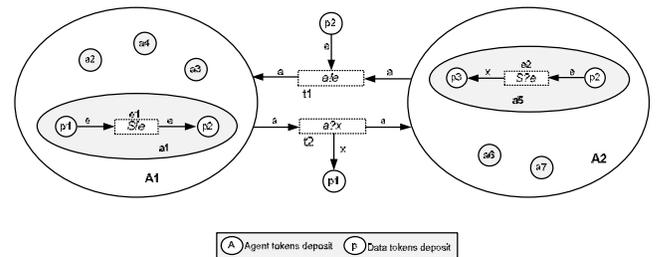
- (4) If  $t$  is  $\alpha$ -enabled at  $M$ ,  $t$  may fire in occurrence mode  $\alpha$ . The firing of  $t$  with  $\alpha$  returns the marking  $M'$  defined by  $M'(p) = M(p) - \bar{L}(p,t):\alpha \cup \bar{L}(t,p):\alpha$  for  $p \in P$ . We use  $M[t/\alpha \triangleright M'$  to denote the firing of  $t$  with occurrence  $\alpha$  under marking  $M$ . As in traditional Petri nets, two enabled transitions may fire at the same time as long as they are not in conflict;
- (5) For a marking  $M$ , the set  $[M \triangleright$  of markings reachable from  $M$  is the smallest set of markings such that  $M \in [M \triangleright$  and if  $M' \in [M \triangleright$  and  $M'[t/\alpha \triangleright M''$  then  $M'' \in [M \triangleright$ , for some  $t \in T$  and occurrence mode  $\alpha$  (note: concurrent transition firings do not produce additional new reachable markings);
- (6) An execution sequence  $M_0 T_0 M_1 T_1 \dots$  of  $N$  is either finite when the last marking is terminal (no more enabled

transition in the last marking) or infinite, in which each  $T_i$  is an execution step consisting of a set of non-conflict firing transitions;

- (7) The behavior of  $N$  is the set of all execution sequences starting from the initial marking.

## 2.2 Nested PrT Nets and Communication Channels

The system net at upper level has a global view for overall processes; thus, agent nets are active tokens in the system net. By active tokens, we mean that agents have their own behaviors that cannot be described by static data. As a result, we have to treat agent tokens as black boxes in system net. Only their identities are visible and accessible. This treatment allows these agent tokens to be grounded and thus well-defined. This treatment also respects the autonomy of an agent. To address the interactions between the system net and agent nets, we extend the constraint definition in a PrT net to include communication expressions. We borrow the channel commands in CSP (Communicating Sequential Processes) [4] for the communication expressions in transition constraints. A communication expression is an output command  $n!e$  or input command  $n?x$ , where  $n$  is a channel name,  $e$  is an expression, and  $x$  is a variable. Channel names are the identifications of nets and are used to control the synchronization of transitions. A synchronized communication occurs when two enabled transitions at different level have a matching pair of input and output expressions. After synchronization, a unidirectional information flow occurs such that the value in  $e$  of the output command is assigned to the variable  $x$  of the input command. For example, as shown in Figure 1, a transition  $t2$  in the system net with identification  $S$  contains  $a?x$ , and a transition  $e1$  in an agent net with identification  $a1$  contains  $S!e$  are a matching pair of input and output expressions. The firing of both transitions is an *interaction* denoted as  $[t2, e1]$ . Transition  $t2$  is called *input channel* that inputs data tokens from lower level net, and transition  $e1$  is called *output channel* that outputs data tokens to upper level net.



**Figure 1. Upper level view of a Nested PrT nets with input/output channels and net tokens.**

To enforce vertical communications (that is, no direct communications between agent nets), the channel names in agent nets must be constants. In this case (in Figure 1), the channel names in all agent nets should be  $S$ . However, the channel names in a system net can be a variable ranging over the agent identifications, which are instantiated with

an enabling agent token. For example, variable ‘ $a$ ’ in transition  $t1$  and  $t2$  in the system net shown in Figure 1.

Formally, a two-level nested net is a tuple  $(S, AG, I)$ , where  $S$  is a PrT net at upper level and called system net;  $AG$  is a finite set of PrT nets and called agent nets, which are active tokens in  $S$ ;  $I \subseteq T_S \times T_{AG}$  is the set of possible interaction relations, where  $T_S$  is the set of transitions in  $S$ ,  $T_{AG} = \bigcup T_i$  such that  $i = 1$  to  $|AG|$ , and  $T_i$  is the set of transitions in agent net  $AG_i$ . We say  $[t, e]$  is an *interaction* in a firing sequence of a nested net such that  $t \in T_S$  and  $e \in T_{AG}$ ;  $t$  and  $e$  are called *communication channels*. We give the formal definitions for communication channels by changing the dynamic semantics defined in section 2.1 as follows.

- (3) Given a marking  $M$ , a transition  $t \in T$ , and an occurrence mode  $\alpha$ ,  $t$  is  $\alpha$ -enabled at  $M$  iff the following predicate is true:  $\forall p: p \in P. (\bar{L}(p, t): \alpha \subseteq M(p)) \wedge R(t): \alpha$ ; where  $R(t) = R_u(t) \wedge R_c(t)$ .  $R_u(t)$  is a non-communication constraint, and  $R_c(t) = n!e \mid n?x$  is a communication constraint.
- (4a) An  $\alpha$ -enabled transition  $t$  with communication constraint  $R_c(t)$  is *ready* if a transition with a matching channel expression is also *ready*. A ready transition  $t$  under marking  $M$  with occurrence  $\alpha$  is fireable.
- (4b) The firing of fireable transition  $t$  with channel expression  $R_c(t)$  under  $M$  with  $\alpha$  returns the marking  $M'$  defined by  $M'(p) = M(p) - \bar{L}(p, t): \alpha \cup \bar{L}(t, p): \alpha$  for  $p \in P$ .  $M[t/\alpha \triangleright M'$  denotes the firing of  $t$  with occurrence  $\alpha$  under marking  $M$ .

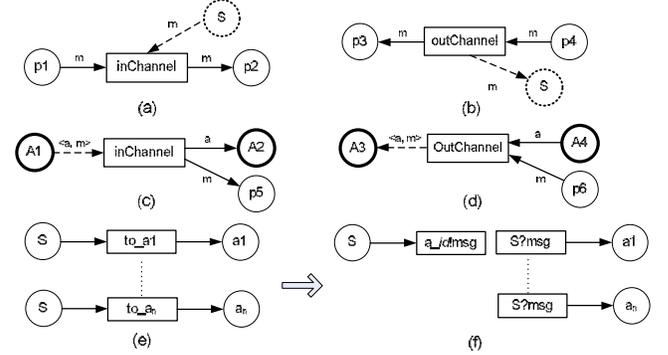
The effect of information flows after an interaction has been reflected in the arc label expressions. For example, the value of variable  $x$  in an input channel command can affect the new marking  $M'$ .

### 2.3 Modeling Communication Channels

We show the typical net structures of modeling communication channels in Figure 2. In an agent net, an incoming token from external environment is considered as an additional enabling condition of a transition (Figure 2(a)), while an outgoing token to external environment is to trigger some external event (Figure 2(b)). The dashed places and arcs denote external resources, thus are decoupled from an agent net. Channel commands such as  $S?m$  and  $S!m$  are then added to the transition constraints of related transitions, which become communication channels.

In a system net, agent nets are identified by their agent ids. A transition with a channel command must have at least a pair of input and output places for holding agent net tokens; for example, places  $A1, A2, A3$  and  $A4$  in Figure 2(c) and 2(d). Formally, a system net  $S$  is a PrT net with a net structure  $(P, T, F)$ ; and,  $\forall t_c \in T. (\exists p_{in} \in \bullet t_c \wedge \exists p_{out} \in t_c \bullet)$ ,  $\varphi(p_{in}) = NET$  and  $\varphi(p_{out}) = NET$ , where  $t_c$  is an input channel or an output channel,  $\bullet t_c$  and  $t_c \bullet$  denote the input and output places of  $t_c$  respectively, and  $NET$  defines the token type of agent nets. Places  $p5$  and  $p6$  in Figure 2(c) and 2(d) hold message tokens from/to an agent net

respectively. Figure 2(f) shows a concise version of net structure from Figure 2(e) by decoupling global and local processes after applying channel commands.



**Figure 2. (a)-(d) communication channels; before (e) and after (f) applying channel commands.**

### 3. Case Study: BCIN Net

We consider BCIN system as a coordination model to accommodate multiple agents in a multi-agent system context. BCIN currently employs a role-based access control (RBAC) mechanism [12]. The following show some of use case scenarios:

*Scenario 1:* David, a member from EOC (Emergency Operations Center), intends to get the most up to date information about Hurricane Wilma, which has been announced as a category one hurricane. The advisory for Wilma has been published and is available in BCIN.

*Scenario 2:* Alice, a supervisor from Hardware Depot for emergency response, intends to assign John as the primary person to publish the resources provided by their company.

*Scenario 3:* Eric, the supervisor of EOC, is going to publish a new advisory for an incoming hurricane.

*Scenario 4:* Emily, a primary contact for emergency response from Shop Mart, has entered BCIN system and read information about resources provided by other companies. She decided to send a message to Hardware Depot, which has three power generators available. Shop Mart needs the generators for frozen foods.

For the simplicity of demonstration, we do not include all possible scenarios. However, the above scenarios are typical regarding information access in current BCIN system. We list the actors and their associated actions in Table 1 based on the above scenarios.

**Table 1. Actors and their associate role and actions.**

Actors	Roles	Actions
David	Company Observer	read information
Alice	Company Supervisor	assign roles
John	Company Participant	publish resources
Eric	EOC Supervisor	publish advisories
Emily	Primary Contact	send messages

Here, an actor is an external entity that interacts with the system. Each actor plays a different role regarding system access. That is, each role has different permission

assignments (PAs) [12], which describe the operations that a role can perform in the system.

To this end, we model the behaviors of an actor as an agent net, and BCIN system as the system net that accommodates multiple agents. In the following sections, the steps for generating agent nets and the system net with communication channels are introduced respectively. Some modeling examples based on the above scenarios in BCIN system are given as well.

### 3.1 Modeling Interactions in an Agent Net

We consider the behaviors related to PAs as an *interaction aspect* of an agent net [1]. An interaction aspect addresses the sociality of an agent and is called a *role model*. Since the information stored in BCIN database is organized based on categories, a typical flow of event for an action in a role model is: (1) send a *request(action, x, criteria)* to trigger the system process *action* regarding category *x* based on the *criteria*, and (2) get the result of *action* from a *receive(action, x, result)*. A ‘*request*’ denotes an outgoing data flow to the system, while a ‘*receive*’ denotes an incoming data flow from the system. Formally, a role model *RM* with net structure  $(P_{RM}, T_{RM}, F_{RM})$  is a PrT net, which models the interaction aspect of an agent net. The set of permission assignments *PAs* of an *RM* defines the set of operations *OP*, where each  $op \in OP$  is either a *request(action, x, criteria)* or a *receive(action, x, result)* operation. A *request* operation involves an outgoing data flow to the system, and a *receive* operation involves an incoming data flow from the system. The set of transition  $T_{op} \subseteq T_{RM}$  models the set of *OP* in an *RM*. As a result, a role model *RM* with communication channels can be generated based on the algorithm in Figure 3.

```

1: set  $T_{op} = \emptyset$ ;
2:  $\forall op \in OP$ 
   {  $t_{op} = action$ ;
      $T_{op} = T_{op} \cup t_{op}$ ;
     add  $\bullet t_{op}$  and  $t_{op} \bullet$  and associated arcs;
      $\forall p \in (\bullet t_{op} \cup t_{op} \bullet)$ , define  $\phi(p)$ ;
     define  $R_u(t_{op})$ ;
     /*  $R_u(t_{op}) \in R(t_{op})$  is a non-communication constraint */
     label each arc of  $t_{op}$  with sort-respecting variables;
     if  $action \in request$  {
        $R_c(t_{op}) = S!x$ ;
       /*  $S$  is the system  $id$ ,  $x \in L(t_{op}, p)$  and  $p \in t_{op} \bullet$  */
        $R(t_{op}) = R_u(t_{op}) \cup R_c(t_{op});$ 
     }
     if  $action \in receive$  {
        $R_c(t_{op}) = S?e$ ;
       /*  $S$  is the system net  $id$ ,  $e \in L(p, t_{op})$  and  $p \in \bullet t_{op}$  */
        $R(t_{op}) = R_u(t_{op}) \cup R_c(t_{op});$ 
     }
   }

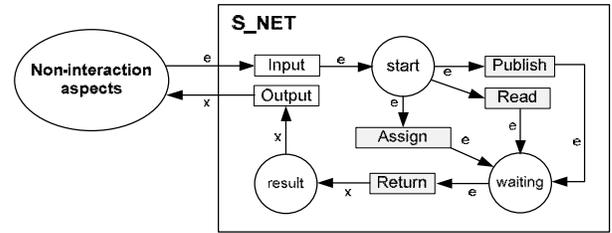
```

**Figure 3. An algorithm for generating a role model.**

*Example1: A PrT net for a supervisor role model.*

Let us look at an example to build a role model for a *supervisor* role by applying the algorithm in Figure 3. Based on Table 1, the operations permitted for a *supervisor* role are read information, publish advisories, and update role assignments. Therefore, we instantiate the *action* in

*request(action, x, criteria)* results in *request(Read, x, criteria)*, *request(Publish, x, criteria)*, and *request(Assign, x, criteria)*. For simplicity, we use only action *Return* in *receive(action, x, result)*. As a result, the set of operations *OP* is defined as  $\{request(Read, x, criteria), request(Publish, x, criteria), request(Assign, x, criteria), receive(Return, x, result)\}$ . The denotation of an *op* is not restricted, however should be differentiated in terms of the direction of an information flow. For example, all operations in *OP* denoted as *request/receive* can be denoted as *output/input* instead. After *OP* is defined, we can generate a PrT net for the *supervisor* role based on the algorithm in Figure 3. The resulting net structure for a *supervisor* role is shown in Figure 4, where relevant semantic definitions are given in Appendix A. Note that we add two boundary transitions (with no input or output places) ‘*Input*’ and ‘*Output*’ to denote the interfaces of the role model for non-interaction aspects.



**Figure 4. A role model for a supervisor role.**

### 3.2 Modeling Interactions in the System Net

The system net has three major components in order to accommodate multiple agents; namely access controls, communications and resource controls. Access controls address the security aspect of the system, while communications and resource controls address the coordination among agents.

#### Modeling Role-based Access Control

From a system view, each user has to be assigned a role for the purpose of access control. Based on Table 1, each user plays a different role. However, it is possible that different users play the same role and one user plays different roles at the same time. Therefore, we consider a *role as a pattern of interactions* within the system. The role assignments happen at runtime, that is, a user becomes an actor enacting a role given by the system after successfully started a session. To this end, our strategy is to keep a net template for each distinct role in a repository. A net template is the behavior model of a role interacting with the system, and is activated while a valid user enacting the role. This strategy not only conforms to agent-oriented design, but also allows the adaptation of interaction behaviors at runtime. The net templates can be built for every distinct roles based on the algorithm described in Figure 3. Activated role models are executing concurrently. In other words, the role models that are activated from the same template (users enacting the same role) may have different markings (states).

Formally, a RBAC model is a tuple  $(RAs, PAs, Sessions, AssignRole, AssignPA)$ , where  $RAs$  is a set of  $U \times R$  relations of users  $U$  and roles  $R$ ,  $PAs$  is a set of  $R \times RM$  relations of  $R$  and role model  $RM$ ,  $Sessions$  is a set of user sessions represented by  $U \times R$ ,  $AssignRole: U \rightarrow R$ , is a function that maps a user in  $U$  to a role in  $R$ ,  $AssignPA: R \rightarrow RM$  is a function that maps a role in  $R$  to a role model  $RM$ . We map each element in the RBAC model to an appropriate net element; for example,  $RAs$ ,  $PAs$ ,  $Sessions$  are modeled as places and the functions are modeled as transitions (see RBAC in Figure 7), where transition  $AssignRole$  assigns a valid user a role based on predefined RAs, and transition  $AssignPA$  assigns PAs by activating an associated role model from net templates. Activated role models (net tokens) are kept in place  $Activated$  with their identifications. Note that boundary transitions  $UserIn$ ,  $UserOut$ ,  $ActorOut$  and  $End$  are added to generate or eliminate tokens. A session constraints [12] is enforced by the formula:  $\exists s \in S. (s = u)$  in transition  $UserOut$  to eliminate an invalid user token that has an active session in place  $Sessions$ . Relevant dynamic definitions can be found in Appendix B.

### Modeling interactions and resource controls

In the system net, a typical interaction sequence of an operation is: (1) an agent net sends the request for an operation (incoming information flow); (2) the system net performs the operation requested (resource controls); and, (3) the result is sent back to the agent net (outgoing information flow). The information flows involved in the above sequence is modeled by a pair of transitions with input/output channel commands to address the interactions with agent nets. Let the set of operations  $OP_s$  denotes the resource controls in a system net  $S$ ; we say a RC net with net structure  $(P_{rc}, T_{rc}, F_{rc})$  models the data and controls of an operation  $op \in OP_s$ , and there exists a pair of transition  $t_{in}$  and  $t_{out}$  model the input and output channel respectively, where  $t_{in} \in T_{rc}$  and  $t_{out} \in T_{rc}$ . The algorithm in Figure 5 describes the steps to build a RC net that models an operation  $op$ .

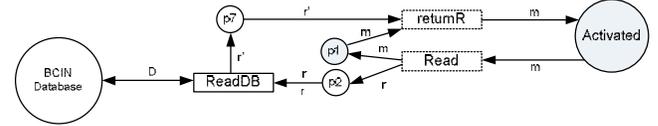
- 1: define  $P_{rc}$ ,  $T_{rc}$  and  $F_{rc}$  for operation  $op \in OP_s$ .
- 2: for all  $p \in P_{rc}$  define  $\phi(p)$ ;
- 3: for all  $t \in T_{rc}$  define  $R_u(t)$ ;
- 4: for all  $f \in F_{rc}$  define  $L$ ;
- 5: add an input channel  $t_{in}$  to  $T_{rc}$ ;  
define  $\bullet t_{in}$ ,  $t_{in} \bullet$  and associated arcs;  
define  $R_u(t_{in})$ ;  
 $R_c(t_{in}) = a!x$ ;  $!$  \*  $a$  is the agent net id \*  
 $R(t_{in}) = R_u(t_{in}) \cup R_c(t_{in})$ ;
- 6: add an output channel  $t_{out}$  to  $T_{rc}$ ;  
define  $\bullet t_{out}$  and  $t_{out} \bullet$  and associated arcs;  
define  $R_u(t_{out})$ ;  
 $R_c(t_{out}) = a!e$ ;  $!$  \*  $a$  is the agent net id \*  
 $R(t_{out}) = R_u(t_{out}) \cup R_c(t_{out})$ ;

**Figure 5. The algorithm for generating a RC net.**

*Example 2: A PrT net for operation 'read' in system net.*

For simplicity, we use a transition  $readDB$  and a pair of input place and output place to model data access against BCIN database (resource controls). Input channel

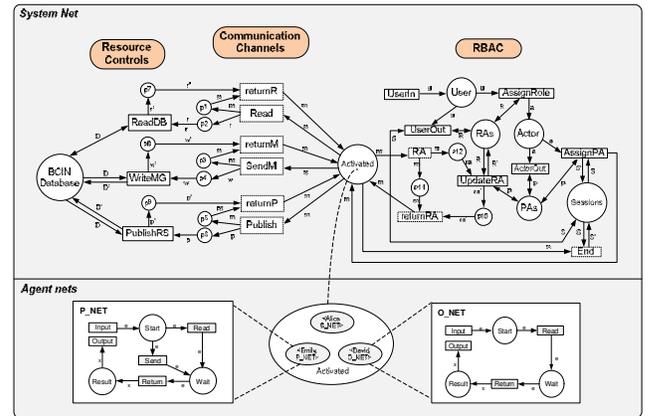
transitions 'Read' and output channel transition 'returnR' are added to address the data flows from/to agent nets. The net structure for operation 'read' is shown in Figure 6. Note that a transition with channel command in a system net has an input and an output places that hold agent net tokens, which can be instantiated for interactions. For example, places  $Activated$  and  $p1$  stores the agent net tokens to enable input and output channel transitions  $Read$  and  $returnR$  when interactions are desired. Detailed constraint definitions are elaborated in Appendix B.



**Figure 6. A resource acquisition operation.**

### The BCIN Net

A BCIN net based on the operations in Table 1 is shown in Figure 7 by applying the algorithms described previously for modeling RBAC, communication channels and resource controls. The detailed semantic definitions for the net are given in Appendix B.



**Figure 7. BCIN Net.**

The resource controls addressed are rather straightforward in this example for demonstration purpose. However, further policies for resource sharing during a global process can be modeled in two ways: (1) define further constraints (rules) in output channel transitions for dispatching resources; and/or, (2) add a input place (or multiple input places) representing additional pre-conditions for the enabling of output channel transitions. Either way, the policies are enforced at the point of synchronization, which results in outgoing information flows downward to an agent net.

### 3.3 Soundness of BCIN Net

There are several important properties regarding information access in BCIN need to be ensured: (1) an invalid user cannot play a role in the system; (2) an actor cannot perform the operations that were not assigned; (3) a user cannot play more than one role at the same time; and, (4) for a request received from an agent net, there will be a

response to the agent net eventually. We briefly give the prove sketches for the above properties as follows.

*Property (1):* The constraint formula:  $\forall r \in R. (r[1] \neq u)$  defined in transition ‘*UserOut*’ (Appendix B) enforces the elimination of invalid user tokens, where  $R$  is the set of valid role assignments.

*Property (2):* The constraint formula:  $\exists p \in P. (p[1] = a[2])$  defined in transition *AssignPA* make sure that an associated role model is available to be activated. Since role models are predefined based on PAs, user behaviors are well controlled. The constraint formula  $\forall p \in P. (p[1] \neq a[2])$  defined in transition ‘*ActorOut*’ eliminates the user token that has not yet been assigned any permissions.

*Property (3):* A session constraint is enforced by the formula  $\exists s \in S. (s = u)$  in the constraint definition of transition *UserOut*; that is, if a user is already in *Sessions*, it is considered as an invalid user and is eliminated.

*Property (4)* The semantic definitions for a *RC* net constructed by applying algorithm given in Figure 5 has to be carefully defined to make sure that if there is a request token  $rq$  in  $p$  such that  $p \in t_{in}^*$ , there will be a result token  $rs$  in  $p$  such that  $p \in t_{out}^*$ , and the agent  $id \in rq$  equals to the agent  $id \in rs$ . For example, in Figure 6, there will be an agent token in place  $p1$  and a request token in place  $p2$  with the same agent  $id$  if transition *Read* fired. Since the constraint definition of transition *ReadDB* (Appendix B) encompasses exception handling, there will be a result token sent to place  $p7$  eventually.

### 3.4 Model Execution

A multi-agent system involves complicated agent interactions and concurrency, thus its behaviors are difficult to be directly observed through textual specifications. One of the advantages to build a formal model such as PrT nets is the operational semantics that are amenable for manipulating model execution. To validate our models, we provide a method to translate a nested PrT net into PROMELA codes [5], which can be executed in SPIN (Simple PROMELA Interpreter). SPIN [5] is a well-developed software tool for validating desired properties of a distributed system. It is especially useful in simulating asynchronous process interactions in concurrent systems.

In terms of program structure in the PROMELA codes simulating BCIN net, we first define necessary constants, string variables (mtype) and data structures (typedef) to describe various types of data tokens; and then, define the system net (Figure 7) as the main process (active proctype) and each agent net (Figure 4) as individual processes to be activated (run) by the main process during simulation. Processes are executed concurrently. In each process, transitions are organized as a set of atomic sequences within a *do...od* loop to be repeatedly executed in a non-deterministic manner. To address the transitions in conflict, *if...fi* statement is used to randomly choose a transition whichever *pre*(condition) is evaluated to be true. To address the interactions (synchronization controls) between nets, rendezvous channels (buffer size 0) are used for agent

communications. Fundamental net elements are translated into PROMELA codes following the principles listed in Table 2.

**Table 2. Translation Principles from PrT nets to PROMELA.**

Net Elements	PROMELA Codes
Place (token deposits) (1) single term (2) Cartesian product (3) Power set.	(1) basic data types (2) typedef { ... } (3) TYPE p[MAX]; /* multiple tokens */ byte p_idx; /* index of array p */
Transition (constraints) (1) $pre \rightarrow post$ (2) $\forall s \in S. (pre \rightarrow post)$ (3) $\exists s \in S. (pre \rightarrow post)$	(1) if :: pre $\rightarrow$ post :: else ... fi (2) do :: index < MAX && !pre $\rightarrow$ break :: index < MAX && pre $\rightarrow$ post; index++ :: index >= MAX; break od (3) do :: index < MAX && pre $\rightarrow$ post; break :: index < MAX && !pre $\rightarrow$ index++ :: index >= MAX; break od
Communication Channels	(1) system channel: chan $S\_id = [0]$ of { MSG, chan}; chan $a\_id$ ; (2) agent channels: chan $me$ ;
Dynamic Interactions (Synchronization)	(1) agent initiate communication: $S\_id!$ start( $me$ ); (agent net send) $S\_id?$ msg( $a\_id$ ); (system net receive) (2) system response: $a\_id!$ msg; (system net send) $me?$ msg; (agent net receive)

The resulting BCIN model in PROMELA language contains 237 lines of codes including five processes: (1) the system net; (2) supervisor role (S\_NET); (3) observer role (O\_NET); (4) primary contact role (P\_NET); and, (5) participant role (PR\_NET). We run the model under XSpin version 5.2.3 based on the initial markings (states):  $M_0(User) = \{<Emily>\}$ ;  $M_0(RAs) = \{<David, observer>, <Alice, supervisor>, <Emily, contact>\}$ ;  $M_0(PAs) = \{<observer, O\_NET>, <supervisor, S\_NET>, <contact, P\_NET>, <participant, PR\_NET>\}$ ;  $M_0(BCIN\_Database) = \{<advisory, Wilma>, <message, generator>\}$ . We generate user tokens ‘*david*’ and ‘*alice*’ at different point in the PROMELA model to test the correctness of interactions between processes. We also intentionally introduce unknown user token to appear at some point of the execution to test the exception handling of the PROMELA model. The properties discussed in Section 3.3 were checked by examining the simulation trails and sequence chart, which enable us to find the subtle errors in our model. The executable verifier for BCIN model consumed 2.501 Mbytes of memory at a rate of around 500 states per second in a search with a maximum depth of 42, where 15 states were stored, 2 states were matched, 17 transitions were explored, and 328 execution trails were generated.

### 4. Concluding Remarks

In this paper, we define a nested PrT net to address a two-layered multi-agent architecture. We incorporate the

channel commands in CSP into PrT nets to model the dynamic interactions between agent nets and the system net. As a result, agents are loosely coupled with the system through communication channels. Agent net as tokens also addresses agent autonomy and heterogeneity. We demonstrate our modeling approach through a disaster mitigation system called BCIN and shows the feasibility of the modeling approach. We validate our models by translating BCIN net into PROMELA codes that can be executed using SPIN tool. Our future works include (1) generalize and define the reusable patterns for modeling the interactions in a multi-agent system using communication channels; and, (2) develop a method to automate the translation from nested PrT nets to PROMELA codes for simulation-based validation.

**Acknowledgements.** The authors would like to express their gratitude to Dr. Shu-Ching Chen and Li Zheng at DMIS Lab in FIU for the discussions of BCIN system. This work was partially supported by NSF grants HRD-0833093.

## Reference

- [1] L. Chang and X. He: Towards Adaptable BDI Agent: A Formal Aspect-Oriented Approach, Proceedings of the International Conference on Software Engineering and Knowledge Engineering (SEKE09), Boston, July, 2009, pp. 189-193.
- [2] F. Fiedrich, an HLA-based Multi-agent System for Optimized Resource Allocation after Strong Earthquakes. In Proceedings of WSC 2006, Monterey, pp. 486-492.
- [3] H. J. Genrich, Predicate/Transition nets. Advances in Petri Nets 1986, pp. 207-247.
- [4] C.A.R. Hoare, Communicating Sequential Processes, Prentice Hall, 1985.
- [5] G. J. Holzmann, The SPIN Model Checker: Primer and Reference Manual, Addison-Wesley, September, 2003.
- [6] H. Kitano, and S. Tadokoro, RoboCup Rescue: A Grand Challenge for Multi-agent and Intelligent Systems. AI Magazine Vol. 22, No. 1, 2001, pp. 39-52.
- [7] M. Kohler, H. Rolke, Modeling Mobility and Mobile Agents Using Nets Within Nets, Proc. of International Conf. on Application and Theory of Petri Nets, LNCS vol. 2679 (2003), 121-139.
- [8] K. F.R. Liu, Agent-Based Resource Discovery Architecture for Environmental Emergency Management, Expert System with Applications, Vol. 27, No.1, July 2004, pp. 77-95.
- [9] N. Okada, E. Hideshima: A Petri Net Approach for Modeling Bottlenecks in the Restoration and/or Restructuring Process of Multiplex disaster-Damaged Urban Infrastructure Systems, Journal of Natural Disaster Science, Vol. 17, No. 2, 1995, pp.75-86.
- [10] X. Pan, C. Han, K. Dauber, K. H., A Multi-Agent Based Framework for the Simulation of Human and Social Behaviors During Emergency Evacuations, AI & Society, Vol. 22, No. 2, 2007, pp.113-132, Springer-Verlang.
- [11] K. Saleem, S. Luis, Y. Deng, S.C. Chen, V. Hristidis, T. Li: Towards a Business Continuity Information Network for Rapid Disaster Recovery, Proceedings of the 9th Annual International Conference on Digital Government Research, Partnerships for Public Innovation, Montreal, Canada, pp. 107-116, 2008.
- [12] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, C. E. Youman: Role-Based Access Control Models. IEEE Computer Vol. 29, No.2, pp. 38-47, 1996.
- [13] R. Valk, Petri Nets as Token Objects: An Introduction to Elementary Object Nets. Proceedings of International Conference on Application and Theory of Petri Nets (1998), LNCS, Vol. 1420, pp.1-25.
- [14] M. Wooldridge, "An Introduction to Multi-agent Systems", J. Wiley, New York, 2002.
- [15] D. Xu, J. Yin, Y. Deng and J. Ding: A Formal Architecture Model for Logical Agent Mobility. IEEE Transactions on Software Engineering. Vol. 29, No. 1, pp. 31-45, Jan. 2003.

## Appendix

### A. Semantic definitions for supervisor role in Figure 4.

$$\begin{aligned} \varphi(start) &= \varphi(waiting) = NAME \times ACTION \times CATEGORY \times CONTENT; \\ \varphi(result) &= CONTENT; \\ R(Read) &= e[1] = 'read' \wedge S!e; \\ R(Publish) &= e[1] = 'publish' \wedge S!e; \\ R(Assign) &= e[1] = 'assign' \wedge S!e; \\ R(Return) &= S?r \wedge r[2] = e[1] \wedge x = r[4] \end{aligned}$$

### B. Semantic definitions for BCIN net in Figure 7.

$$\begin{aligned} \varphi(Users) &= NAME; \\ \varphi(RAs) &= \wp(NAME \times ROLE); \\ \varphi(Sessions) &= \wp(NAME); \\ \varphi(Actor) &= NAME \times ROLE; \\ \varphi(PAs) &= \wp(ROLE \times RM); \\ \varphi(Activated) &= \varphi(p1) = NAME \times RM; \\ \varphi(BCIN\_Database) &= \wp(CATEGORY \times CONTENT); \\ \varphi(p2) = \varphi(p7) &= NAME \times ACTION \times CATEGORY \times CONTENT \\ \varphi(p3) = \varphi(p5) = \varphi(p11) &= NAME \times RM; \\ \varphi(p4) = \varphi(p8) = \varphi(p6) = \varphi(p9) = \varphi(p12) = \varphi(p10) &= NAME \times ACTION \times CATEGORY \times CONTENT; \\ (UserOut) &= \exists s \in S. (s = u) \vee \forall r \in R. (r[1] \neq u); \\ R(AssignRole) &= \exists r \in R. (r[1] = u) \wedge a = r; \\ R(AssignPA) &= \exists p \in P. (p[1] = a[2]) \wedge m[1] = a[1] \wedge m[2] = p[2] \wedge S' = S \cup a; \\ R(ActorOut) &= \forall p \in P. (p[1] \neq a[2]); \\ R(End) &= m[1]?op \wedge op[1] = 'quit' \wedge \exists s \in S. (s = m[1]) \wedge S' = Ss; \\ R(Read) &= m[1]?op \wedge r = op; \\ R(ReadDB) &= (\exists d \in D. (d[1] = r[3]) \wedge r'[4] = d[2]) \vee (\forall d \in D. (d[1] \neq r[3]) \wedge r'[4] = 'error'); \\ R(returnR) &= m[1] = r'[1] \wedge m[1]!r'; \\ R(SendM) &= m[1]?op \wedge w = op; \\ R(WriteMG) &= D' = D \cup \langle w[3], w[4] \rangle \wedge w'[4] = 'message sent'; \\ R(returnM) &= m[1] = w'[1] \wedge s!w'; \\ R(Publish) &= m[1]?op \wedge p = op; \\ R(PublishRS) &= D' = D \cup \langle p[3], p[4] \rangle \wedge p'[4] = 'published'; \\ R(returnP) &= m[1] = p'[1] \wedge m[1]!p'; \\ R(RA) &= m[1]?op \wedge ra = op; \\ R(UpdateRA) &= (\exists p \in P. (p[1] = ra[4]) \wedge ((\forall r \in R. (r[1] \neq ra[3]) \wedge R' = R \cup \langle ra[3], ra[4] \rangle \wedge ra'[4] = 'added') \vee (\exists r \in R. (r[1] = ra[3]) \wedge r'[2] = ra[4]) \wedge ra'[4] = 'updated')) \vee (\forall p \in P. (p[1] \neq ra[4]) \wedge ra'[4] = 'error')); \\ R(returnRA) &= m[1] = ra'[1] \wedge m[1]!ra'; \end{aligned}$$

# Agent-based Architecture for Service Ontology evolution management

Soumaya Slimani, Salah Baina, Karim Baina

ENSIAS, Mohammed V-Souissi University,

BP 713 Agdal, Rabat, Morocco

slimani.soumaya@gmail.com, {sbaina, baina}@ensias.ma

**Abstract**— Communication between services requires a common vocabulary to allow reliable exchange of information. However services are evolving independently and their ontologies evolve too. The existing distributed ontology evolution approaches are not scaled to dynamic environments like Semantic Web service architecture. As the distributed system grows in size, the complexity of ontology change management increases, especially if the ontologies are heterogeneous. In this paper, a new approach is proposed to manage ontology evolution in distributed architecture. Our proposal consists of a set of collaborative agents which dynamically manage different ontologies and their mapping.

## I. INTRODUCTION

Semantic interoperability of heterogeneous Web services can be achieved through an agreement between the underlying ontologies. Indeed, ontologies allow services taking into account semantics of data during exchanges. Based on ontologies, several standards were proposed to describe semantic Web services (WSDL-S, WSMO, OWL-S, etc.) and thus, motivate the development of semantic services. Several works have developed tools and adopted approaches to develop semantic services [1, 2, 3]. In this context two scenarios are possible: (1) Distributed System based on single ontology and multiple instances of this ontology (SOMI). (2) Distributed System based on multiple ontologies and single or multiple instances of these ontologies (MOSI, MOMI). Indeed, ontology management can be complex and time consuming. Moreover, in a collaborative context, an ontology that evolves can cause the evolution of other ontologies or instances of ontology. Several studies have addressed this issue [4, 5, 6]. However, research in ontology evolution and change management deal particularly with the versioning and evolution of the same ontology. When ontologies are considered as instances of a source ontology, the ontology evolution will require applying the same changes on the instances.

In this paper we present a new approach for managing ontologies in a distributed environment, we treat the case of different ontologies and not instances of the same ontology. Note also that in the case of different ontologies, related mappings must be managed in parallel.

## II. OUR PROPOSITION FOR MANAGING ONTOLOGY EVOLUTION

The typical characteristics of ontology and service ontology-based applications can be identified as:

distributed, heterogeneous, and highly dynamic. Agent technology is thus ideally suited to this context.

### A. Architecture overview

Our architecture consists of a set of Ontology Agent (OA) that acts on a service ontology. Agents are responsible of change detection and propagation of these changes to other dependent agents. Dependent agents manage and update their ontologies and the related mapping. The overall architecture of our proposal is shown in Fig. 1.

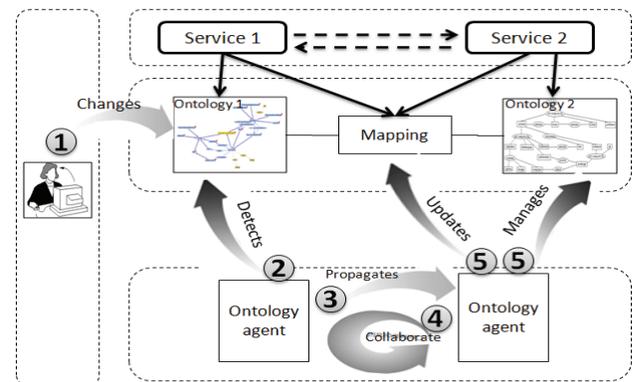


Fig. 1. Architecture overview

An agent will take one of two roles: (i) Initiator Ontology Agent, when its ontology changed, and (ii) Dependant Ontology Agent, when its ontology depends on a changed ontology. So, we present two algorithms to illustrate agent roles: (1) *Detection/propagation of change* (initiator role). (2) *Receipt/processing of change* (receiver role).

### B. Algorithm of detection / propagation of change (initiator side)

Our algorithm illustrated in Fig.2, takes as input an ontology and a list of receivers (agents of the dependent ontologies). Once a change is made, the agent lists these changes automatically. We define a change  $C_i$  by a quadruplet:

$$C_i = \langle \text{Operation}, \text{Object}, \text{Subject}, \text{Predicate} \rangle \quad (1)$$

An *Operation* specifies the change operation (Add, Delete, etc). The *Object* specifies the type of component that has changed (Class, Property, individual). The *Subject* represents the name of the component that has changed. *Predicate* represents a set of information about

the change. If a class was changed, this field contains the position of the class in the hierarchy of the ontology (its *super-classes* and *sub-classes*). But, if a property was changed, it contains the *Range*, *Domain*, *super- propertie* and *sub-properties*.

We also define a message or *changeSet* by a list of changes:

$$\text{changeSet} = \langle C_1, C_2, \dots, C_n \rangle \quad (2)$$

After listing all changes, the agent classifies these changes into two categories:

(1) Changes contained in the related mapping between initiator ontology and receiver ontology or that are connected (*sub-class*, *property*, etc..) to an elements in this mapping. These changes are encapsulated into a message and sent to all dependent ontologies agents.

(2) Changes that are not related to the mapping between the two ontologies. These changes are ignored because they do not involve communication between the two services.

For changes to be propagated, translation phase is necessary. The *translateChange* method generates a new list of changes by replacing the objects of the field *Predicate* in the original list by their match in the related mapping.

```

Algorithm : Change Detection and propagation
Require : Ontology O, List of Receiver
receivers
While (true)
  O.changed()=listen(O)
  If(O.changed() == true) then
    changeSet= calculateChangeSet()
    For each Receiver Ri of receivers do
      Mapping Mi= GetMapping(O, Ri)
      For each change c of changeSet
        If (c.isRelated(Mi) == true) then
          Change Nc = translateChange(c,Mi)
          newChangeSet.add(Nc)
        end if
      end For
      Send( newChangeSet, Ri)
    end For
  end if
end While

```

Fig. 2. Initiator side Algorithm

### C. Algorithm receipt / processing of change (receiver side)

As shown in Fig. 3, the receiver side algorithm consists of six main operations:

- (1) **Semantic search:** this method looks for a semantic corresponding object of change (synonymy, etc.).
- (2) **Syntactic search:** find if there is a concept in the ontology which has the same syntax as the changed object.
- (3) **Computation of object definition:** Class definition is expressed by the position of the class in the hierarchy of the ontology (its super classes and subclasses).

Property definition is made based on its "*Range*", its "*Domain*", and its supers or under properties.

- (4) **Negotiation of object definition:** Negotiations, in our case, correspond to the reformulation of the object definitions, and the exchange of these definitions. The receiver agent calculates a definition of an object, and then sends it to the initiator agent. If the initiator agent finds that the two definitions are equivalent, it sends its agreement; otherwise it recalculates the definition, and send the new definition to the receiver agent, and so on. The negotiation ends when either receiver and initiator agent have a common definition of the object or they find a conflict.
- (5) **Annotation of ontology objects:** the *annotateOntology* method allows us to add new object as annotation of existing object. An annotation is a comment, a note or any other external remarks that can be attached to an entity of the ontology. In OWL DL, it is possible to annotate classes, properties and individuals. There are five predefined annotation properties in OWL: Owl:versionInfo, Rdfs:label, Rdfs:comment, Rdfs:seeAlso, Rdfs:isDefinedBy.
- (6) **Ontology update Mapping:** If ontologies evolve, the mapping must evolve too. The *updateMapping* method allows adding new correspondence between two concepts in the mapping.

```

Algorithm Receive ChangeSet: ADD Operation
Require: changeSet:List of Changes,
m:Mapping, O : Ontology
Waiting for Message
For each change c in changeSet do
  Object S, cS
  cS= c.Subject
  Boolean synSearch=O.syntacticSearch(cS)
  Boolean semSearch=O.semanticSearch(cS, S)
  If (c.Operation == ADD) then
    If (synSearch && semSearch) then
      m.updateMapping(S, cS)
    end if
    If (synSearch &&!semSearch) then
      Definition d = calculateDefinition(cS)
      Boolean agree = negotiate(d)
      If agree == false then
        informUser ()
      else
        m.updateMapping(cS, cS)
      end if
    end if
    If (!synSearch && semSearch) then
      O.annotateOntology( cS,S)
      m.updateMapping(S, cS)
    end if
    If ( !synSearch && !semSearch) then
      O.add(cS)
      m.updateMapping(cS, cS)
    end if
  end if
end for

```

Fig. 3. Receiver side Algorithm

Based on the results of the syntactic and semantics search, the agent chooses one of four alternatives and executes related actions summarized below:

TABLE I  
DEPENDENT AGENT ACTIONS

		Syntactic search	
		True	False
Semantic search	True	-Updates the related mapping	-Updates the mapping. -Annotates the corresponding object by the added object.
	False	-Reformulates the change definition. -Negotiates the change definition.	-Adds the new object to the ontology. -Updates the related mapping.

### III. CASE STUDY

In this section, we present an example of communication between services. We limit to an example of two services. But this example can be generalized to a more complex architecture. The running example ontologies come from the domain of Telecom and concern two services. The first service is the SMS billing service ( $S_1$ ), the second one is the Internet billing service ( $S_2$ ).  $S_1$  is described by an ontology  $O_1$  and  $S_2$  is described by an ontology  $O_2$ . Initially, services ( $S_1$  and  $S_2$ ) communicate via a Mapping  $M$ .  $S_2$  adds a new service providing information to the client by SMS. This SMS, called INTERNET\_SMS, contains information about the status of the client account. These SMS is sent by the client service to the client. Then,  $S_2$  adds a new Internet SMS service, which a client can send to another via Internet. We call this new type of SMS: INTERNET\_SMS and we rename the old one INTERNET\_SMS\_INFO.

Following these evolutions, both services have two different interpretations of the concept INTERNET\_SMS. So, communication can not be done properly.

The application of the algorithm in this scenario begins on the Initiator side ( $O_2$  agent). First, changes are listed as shown in Fig.4.

```
changeSet = <
C1=<ADD:Class:INTERNET_SMS:subclassOf SMS>,
C2=<ADD:ObjectProperty:INTERNET_SMS_SENDER:
  <Domain:INTERNET_SMS,Range:SERVICE_CLIENT>>
C3=<ADD:ObjectProperty:INTERNET_SMS_RECEIVER:<Domain:
  INTERNET_SMS,Range:CLIENT>>>
```

Fig. 4. The change set after adding INTERNET\_SMS

Changes are classified according to their relationship with the mapping. All changes are sent to  $O_1$  agent. In the receiver side ( $O_1$  agent), the syntactic and semantic search will return false. Because all added objects are new objects for the ontology  $O_1$ . Thus the agent chooses the fourth case. So,  $C_1$ ,  $C_2$ , and  $C_3$  are added to  $O_1$  and the mapping is updated (i.e.  $mapped(O_1.C_1, O_2.C_1, \dots)$ ). Thus, the Internet service can communicate with the SMS

service using the concept INTERNET\_SMS. However, after adding the new Internet SMS type, the mapping between  $O_1$ .INTERNET\_SMS and  $O_2$ .INTERNET\_SMS becomes wrong. Semantic of INTERNET\_SMS has changed in  $O_2$ , but it kept the same semantics in  $O_1$ . Our algorithm handles this case and  $O_2$  agent generates a list of changes as shown in Fig.5.

```
changeSet = <
C1=<ADD:Class:INTERNET_SMS:subclassOf SMS>>
C2=<ADD:ObjectProperty:INTERNET_SMS_SENDER:
  <Domain:INTERNET_SMS,Range:CLIENT>>
C3=<ADD:ObjectProperty:INTERNET_SMS_RECEIVER:
  <Domain:INTERNET_SMS,Range:CLIENT>>>
```

Fig. 5. The change set after adding the new INTERNET\_SMS

These changes are sent to  $O_1$  agent. The object INTERNET\_SMS in  $O_1$  has a different semantic than INTERNET\_SMS in  $O_2$ . In this case the agent chooses the second case in the algorithm.  $O_1$  agent calculates a definition for INTERNET\_SMS as shown in Fig.6.

```
<<Class:O1.INTERNET_SMS:subclassof
  shortMessage>
<Class:O1.SERVICE_CLIENT:subclassof Thing>
<ObjectProperty:O1.SENDER:<Domain:
  O1.INTERNET_SMS,Range:O1.SERVICE_CLIENT>>>
```

Fig. 6. The new INTERNET\_SMS definition calculated by  $O_1$

$O_1$  agent sends this definition to  $O_2$  agent, which calculates a new definition:

```
<<Class:O1.INTERNET_SMS:subclassof
  shortMessage>
<Class:O1.SERVICE_CLIENT:subclassof Thing>
<ObjectProperty:O1.SENDER:Domain:
  O1.INTERNET_SMS,Range:O1.SERVICE_CLIENT>
<Class:O2.INTERNET_SMS:subclass of SMS>
<ObjectProperty:O2.SENDER:Domain:
  O2.INTERNET_SMS,Range:CLIENT>
<O1.CLIENT↔O2.SERVICE_CLIENT>>
```

Fig. 7. The new INTERNET\_SMS definition calculated by  $O_2$

The last matching, *mapped* (CLIENT, SERVICE\_CLIENT), introduces a conflict in the definition of SERVICE\_CLIENT.

Indeed, as we have already in the mapping that:

*mapped* ( $O_1$ .CLIENT,  $O_2$ .CLIENT)

Introducing in the mapping the following correspondence:

*mapped* ( $O_1$ .CLIENT,  $O_2$ .CLIENTSERVICE)

Implies that:

*mapped* ( $O_2$ .CLIENT,  $O_2$ .SERVICE\_CLIENT)

This operation can generate an error. In this case, the agent sends an alert message to the user. The user can modify the mapping and the ontology  $O_1$  or validate any of the definitions contained in the negotiations exchange.

Our algorithm provides a support for communication between services. In this example, it allows to services ( $S_1$  and  $S_2$ ) to be aware of evolution in other services. Agents take the necessary decisions to maintain this

communication by applying the necessary changes to the mapping and to the ontologies. But if the agent detects a conflict it only informs the user of the problem.

#### IV. IMPLEMENTATION OVERVIEW

The initiator algorithm and the receiver side algorithm have been implemented. Our proposal presents a solution that can be easily used by service designers and developers within their SOA technical infrastructures and SOBA business solutions. As shown in Fig.8, four layers are defined: (1) Service layer: This is outside our architecture, represents services that are described by service ontologies. (2) Mapping layer: Service dependencies are represented by service ontology mappings defined by human designers too using ontology mapping generators. (3) Ontology layer: Service ontologies are defined by human designers using ontology editors, and these service ontologies are then monitored synchronously. (4) Agent layer: Ontology agents are defined following our Ontology Agent Model and algorithms described in the section II.

For each layer we have studied many alternatives to develop our prototype. For storing ontologies, we have studied DAML+OIL, OWL, RDF, RDF (S). We have finally chosen OWL (<http://www.w3.org/TR/owl-features/>). We have chosen to work with Protégé 3.4 (<http://protege.stanford.edu>), but other editors may be used since the *Ontology Agent* is listening to ontology basic file change events before their processing. For ontologies mapping, we have studied LOM, IFMap, OMEN, GLUE, FCA-merge, Prompt (Protégé plugin). Our final choice was for Prompt (<http://protege.stanford.edu/plugins/promp t/prompt.html>) that generates an OWL format for the mapping.

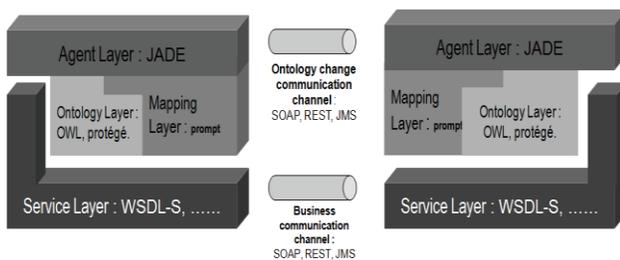


Fig. 8. Implementation overview

Finally, we have studied many alternatives for implementing the *Ontology Agent Model*: AgentBuilder, ASL, FIPA-OS, MOLE, Open Agent Architecture, *RETSINA*, *Zeus*, JADE. And, we have finally chosen JADE (<http://jade.tilab.com/>) to develop our *Ontology Agent Model*. Interactions between agents (i.e. ontology change event messages) are handled through an ontology change communication channel.

#### V. CONCLUSION AND FUTURE WORKS

Algorithms for ontology evolution are not suited for semantic service based application. In these applications, services are described by ontology and collaborate with each other. In this context, each entity has its own semantic and its own interpretation of the exchanged data. This makes communication between services difficult and maintaining this communication is more difficult and complex. We adopt a MAS perspective to model ontologies in ontology-based applications. An agent-based approach aims to provide a flexible way to automate the process of ontology evolution management as much as possible. Moreover, agents which are autonomous and engaged in flexible interactions can perceive changes in the environment and adopt corresponding actions to maintain the mapping between ontologies. Our algorithm covers the basic changes in initiator side and treats the Add operation in the receiver side. We consider the update operation as a combination of deletion and addition operation. Deletion is still being analyzed in our algorithm.

The ontology agent model contains three behaviors: (1) a listener to detect change, (2) a behavior for changes formulation and propagation (initiator side Algorithm), and (3) a behavior for receiving and processing changes (receiver side algorithm). We have developed our model in the agent platform JADE.

The approach could be extended and improved in several ways. There is a need to extend the algorithms to include different types of changes in our algorithms (removal, update, etc...). This paper is thus a first step to lay the groundwork for future research.

#### REFERENCES

- [1] P. Châtel, Toward a Semantic Web service discovery and dynamic orchestration based on the formal specification of functional domain knowledge, ICSSEA, 2007.
- [2] S. Izza, L. Vincent, and P. Burlat, "A Unified Framework for Application Integration : an Ontology-driven Service-oriented Approach". ChinSheng Chen, Joaquim Filipe, Isabel Seruca, José Cordeiro (Eds.): Proceedings of the Seventh International Conference on Enterprise Information Systems, Miami, USA, pp. 165-170, ICEIS 2005.
- [3] A. Haller, J.M. Gomez, and C. Bussler, "Exposing SemanticWeb Service principles in SOA to solve EAI scenarios", Digital Enterprise Research Institute, *WWW, Japan*, May 10\_14, 2005.
- [4] P. Plessers, O. Troyer, and S. Casteleyn, "Understanding ontology evolution : A change detection approach, Web Semantics : Science, Services and Agents", *on the World Wide Web 5* 39-49, 2007
- [5] R. Djedidi, MA. Aufaure: ONTO-EVOAL an Ontology Evolution Approach Guided by Pattern Modeling and Quality Evaluation. FoKS, 2010.
- [6] F. Zablith, " Evolva : A Comprehensive Approach to Ontology Evolution". *In Proceedings of the 6th European Semantic Web Conference PhD Symposium LNCS 5554. eds. L. Aroyo et al., Springer-Verlag, Berlin, Heidelberg, pages 944-948, ESWC, 2009.*

# Intelligent Software Agent Design Issues with Extensions to the Descartes Specification Language

Vinitha H.Subburaj , Joseph E.Urban  
Computer Science Department  
Texas Tech University  
Lubbock, Texas 79409-3104 USA  
{vinitha.subburaj,joseph.urban}@ttu.edu

**Abstract**—Development of intelligent software agents that include learning and reasoning has recently challenged the software industry. The complexity involved in the development of such systems demands a specific methodology be used. To tackle this problem, this paper presents a formal approach to specification for intelligent software agents. The paper describes the different types of agents and the existing methodologies to specify those agent categories. The paper includes an analysis of intelligent software agents, capabilities, characteristics, and design issues. This paper also discusses the Descartes specification language, an executable formal method which was extended to derive the design of intelligent software agents. The extensions made to the Descartes specification language and the appropriateness towards the development of intelligent systems are also stated.

*Keywords*- intelligent software agents; software specification

## I. INTRODUCTION

With the current situation of explosive growth of information around the world, the related technical issues have given rise to the development of agent systems. An agent is an entity that performs the task on behalf of another party. There is currently a wide classification of agents available, out of which intelligent agent is one such category. Development of such intelligent software agents involves a high level of complexity due to the high level of interaction between the agents and learning needed to train a system. This complexity level demands a specific approach to be used while developing intelligent software agents. The problem is to develop a formal methodology that derives the design of a complex intelligent software agent. One approach would be existing agent specification methodologies applied to intelligent software agents.

To address the above issues, this paper presents a formal approach to specify the design of intelligent software agents using an extension to the Descartes specification language. The remainder of this paper is structured as follows. In Section II, related work that has been done in developing agent systems using formal methods is discussed. Section III discusses the research methodology proposed in this paper. A brief

description of the intelligent software agent paradigm and the characteristics of intelligent software agents are also discussed in Section III. The Descartes specification language and its extension in terms of intelligent software agents are described in Section IV. Section V concludes the paper.

## II. BACKGROUND

A distinction between what an agent is and what an intelligent agent becomes is important at this stage of classification. An agent can be viewed as anyone or anything that performs specific tasks for a third party and the representative party gets benefit from it. Whereas, intelligent agents can be viewed as software entities that carryout tasks on behalf of a user or another program with some degree of autonomy with the employment of a user's goal and desires along with it. The current research areas in the field of intelligent software agents have not had agreement on a single, comprehensive definition for intelligent software agents. On the prospective of this research effort, intelligent software agent is defined as a software program that performs user's tasks autonomously and interacts with its environment with a degree of intelligence.

Earlier work has been done in developing reactive systems by extending the Descartes specification language. Due to the volatile nature of reactive agents, the need for formal methods becomes essential. This research effort [6] included transformation procedures that map from the Descartes specification language into agent modeling concepts, AUML specification and design traceability, and also an abstract execution capability of the agent software. Bauer, et al. [3] proposed an Agent UML which is an extension of the Unified Modeling Language. There is a lack of touch and cooperation between the industry and agent oriented software engineering. Agent UML was introduced in order to overcome these differences.

A multi-agent system was developed with the application of a methodology that extends AUML with Petri nets by Bagic [2]. The work focused on analyzing the connecting points between current AUML specifications with concepts of Petri nets. AUML was used as a main modeling tool for multi-agent systems and Petri nets as a formal verification and validation tool. Wooldridge, et al. [7] dealt with several issues relating to

formalisms for the description and development of multi-agent systems. Most of the formalisms are logic based and used in one of three ways: formal philosophy, knowledge representation, or as a basis of programming.

Wooldridge and Ciancarini [8], argued that intelligent agents and multi-agent systems are tools used to manage the complexity of software systems. An agent is a system that enjoys the following properties: autonomy, reactivity, proactiveness, and social ability. Multi-agent systems are seen as an important new direction in software engineering because of the following reasons: natural metaphor, distribution of data and control, legacy systems, and open systems.

A formal specification language known as SLABS for agent based systems was given by Zhu [10]. The specifications for a multi agent system in SLABS consist of specifications for agents and castes. A caste description consists of states, actions, behavior, and environment. The design and implementation of a multi agent specification system called CASA was developed by Geiger and Flake [5]. The clauses of AgentSpeak(L) are used as a base for specification of CASA agents.

### III. INTELLIGENT SOFTWARE AGENT PARADIGM

A clear concept of intelligent agents is obtained initially to proceed any further. What the agent does, its interaction with the environment, the steps involved in building one along with its evaluation needs to be addressed clearly. The methodology will design an intelligent agent system that maps from precepts to concepts. The methodology used in this research effort was a formal method that overcomes the complexities and unforeseen problems during the development of agent systems. Extending the Descartes specification language, such that a formal derivation of the intelligent software agents design is made possible, was the challenging task in the research. The resulting output was evaluated using the IEEE std. 830-1998 characteristics for a good software requirements specification [1] along with added criteria like executable software requirements specification.

#### A. Characteristics of intelligent software agents

Intelligent software agents vary from simple to more complex agents with the degree of characteristics possessed. Simple agents might have one or two characteristics, whereas highly-complex agent systems will have almost all of the characteristics. Characteristics of intelligent software agents were grouped under two large categories [4] of internal and external properties. Internal properties included ability to learn, reactivity, autonomy, and goal-orientedness. External properties included communication, cooperation, and coordination. Characteristics strictly do not follow under one of these two categories.

The following are the intelligent software agent characteristics in the context of this paper's research.

**Autonomy:** Agents should act independently to carry out their tasks in controlling their own actions and states without any direct assistance from human sources or outside.

**Social ability:** Agents should interact with the appropriate environment which includes human and other software agents. Intelligent software agents should work together with other software agents to achieve mutually beneficial complex tasks.

**Responsiveness:** Agents should respond appropriately in a timely fashion to the changes perceived in the environment which includes other agents, humans, and external physical world.

**Proactiveness:** Agents should respond to the environmental changes with a goal-directed behavior.

Along with the common properties that intelligent systems share with other types of agents, knowledge base, reasoning capabilities, and ability to learn are the unique properties that constitute the intelligence of intelligent agents.

**Ability to learn:** Agents should be able to learn from past experience and carryout the tasks accordingly. This characteristic of intelligent agents becomes more important in cases of identifying virtual students in a collaborative learning environment.

**Mobility:** Mobility is the degree to which agents themselves travel through the network. Applications of intelligent software agents are initially designed to be static, and then the mobility will appear gradually in time.

**Communication:** Intelligent software agents access information from third party sources about the current state of the external environment. To access the information requires the ability for agents to communicate with other software agents or other repositories of information. Communication among agents can be in the form of singular request/response or can be complex involving a multiple request/response pattern.

Other intelligent software agent characteristics include cooperative behavior, temporal continuity, collaborative behavior, inferential ability, agent's character, and reasoning.

#### B. Role of intelligent agents

Development in the field of agent systems is taking slow progress, and predicted to take over many of the information intensive tasks in the future. To clearly indicate the basic functionality of an intelligent software agent, consider the following simple example. In a library system, a student can inform the agent that they are interested in a particular area. The intelligent agent would search the database available in digital form on behalf of the user and inform the most relevant information that pertains to the search made by the user. The intelligent agent may be viewed as one similar to a search engine at this point of view, but what differentiates the former from the latter is that, it possesses intelligence, which means that the agent is provided with knowledge of information and the agent retrieves information based on this knowledge.

Intelligent agents get more complex when the level of information and the level of intelligence that an agent possesses increase. The potential benefit of these intelligent systems might range from a private user to a huge industrial giant that involves extensive information processing with learning involved. With these learning aspects of intelligent

software agents along with other base properties of agents being made distinct, the development overhead associated with intelligent software agents is visible.

#### IV. EXTENSIONS TO THE DESCARTES SPECIFICATION LANGUAGE

The Descartes specification language was developed by Urban [6]. This tool was designed to be used throughout the software life cycle. The relationship between the input and the output of a system is functionally specified using this specification language. Descartes defines the input data and output data and then relates them in such a way that output data becomes a function of input data. The data structuring methods used with this language are known as Hoare trees. Consider the following example,

```
address
  street
  city
  country
```

shows the node “address” as having three sub trees with indentation.

These Hoare trees use three structuring methods that are namely direct product, discriminated union, and sequence.

Direct product is the default and provides for the concatenation of sets of elements.

Discriminated union provides for the selection of one element out of a set of elements. Discriminated union is denoted by a plus sign (+) suffixed to the node name.

Consider the following example,

```
agent_state+
  active
  inactive
```

indicates the agent is either in state active or inactive.

Sequence represents zero or more repetitions of a set of elements. Sequence is indicated by an asterisk (\*) suffixed to the node name. Consider the following example,

```
agent_message*
  letter
```

indicates the agent message is a sequence of zero or more occurrences of letter.

By definition of Hoare trees, a sequence node is followed by a subnode. A single node can accommodate a sequence of direct product or a sequence of discriminated union. The length of the sequence can also be indicated as follows,

```
initial(1..3)
  letter
```

illustrates that an initial can range from 1 to 3 letters. As with this case, the lower bound is 1 and upper bound is 3. This indication of length can take different forms as an indication of only lower bound and no upper bound.

In the Descartes specification language, a literal is any string that is enclosed within single quotes. Consider the following example,

```
agent
  'intelligent_agent'
```

wherein intelligent\_agent is a literal.

#### A. Methodology

The methodology introduces several extensions to the Descartes specification language with a motive to effectively represent the characteristics of intelligent software agents. A brief description of the Descartes specification language and the writing of specifications are used as a base for providing the extension to the language. The modules of the Descartes specification language are broadened to support intelligent software agent design and description in terms of their properties. The extensions made will follow top-down modularity to preserve the Descartes specification language’s originality and benefits.

Extended Descartes specification language features for representing intelligent agents are in accordance with Wooldridge intelligent agent definitions [9]. Extensions to the Descartes specification language for supporting reactive agents have been already given by Medina and Urban [6]. The constructs introduced capture the requirements of reactive agents using the Descartes specification language. Purely reactive agents decide on their tasks without reference to their history and simply respond to their environment directly. But, with intelligent agents, a proper balance between the reactivity (ability to perceive it environment) and pro-activeness (goal-directed behavior) should be maintained [9]. Presented with this balance in focus, are the following extensions to Descartes for supporting intelligent agents. In this research effort, extensions to the Descartes specification language possess some knowledge of past and take action based on the past information.

#### B. Agent declaration and state specification

The notion of declaring an agent is similar to the identification of objects in object oriented analysis. Notation developed by Medina and Urban [6] for declaration and organization of agents is used for specifying intelligent software agents. The declaration of agent module is pre-pended with a unary “agent” reserved word. A unique identifier is passed into the module following the “agent” reserved word.

Intelligent software agents, in order to exhibit the property of intelligence, should possess some knowledge about the past. For this concept, an agent module should be encapsulated with its state. The state defines the various possible states that an agent can be in the software system. More clearly, the current state of an agent will give some information and prediction on the previous states and the future states that the same agent will be in.

The reserved word, astate, is used to specify the different possible states an agent can be in. Similarly, the reserved word, estate, is used to specify the different possible environment states. Based on the agent state and environment state the next action will be triggered.

```
agent (FILE_MANAGER)_AGENT
  FILE_MANAGER
  'filem'
astate
  UNINITIATED
```

INITIATED  
ACTIVE  
TERMINATED

**estate**

S0  
S1

*C. Specification of decision function and timer module*

An intelligent agent is distinct from the other agents in the sense that it will not blindly follow a sequence of actions or events to be executed. In accordance with the Wooldridge definition of intelligent agents and their design choices [9], an agent decision function is divided into perception and action subsystems. The idea is that the agent observes the environment and takes action based on the observation. Extended language features permit reservation of “perception” and “action”.

Perception of the environment is specified under the “perception” reserved word. Perception is an important concept with regard to intelligent software agents, as it leads to decision making strategies under a knowledge base. Actions are the set of actions that an agent can process under the perception of its environment. The reserved word “action” is used to derive a specific action that an intelligent software agent performs under a certain circumstance. Perception and action is represented through a series of analysis and synthesis trees. Consider a case study of a file manager system that requires the assistance of an intelligent software agent to monitor a file on its behalf. A sample specification that uses decision function, an extended Descartes construct for the design of such a system is as follows

**decision**

**perception**

outputfile  
filecontent\*  
‘#’  
CHARACTER  
‘#’  
empty\_file  
‘##’  
‘File status:’  
filestatus(1..1)  
CHARACTER

**action**

CP\_FILESTATUS(FILESTATUS)  
FILESTATUS(1..1)  
CHARACTER  
CP\_FILESTATUS\_EQUALS\_('c')  
PRINT\_MESSAGE(“File changed”)  
CP\_FILESTATUS\_EQUALS\_('o')  
PRINT\_MESSAGE(“File overflow”)  
CP\_FILESTATUS\_EQUALS\_('d')  
PRINT\_MESSAGE(“File deleted”)

A timer module is defined using the Descartes specification language to initiate loops of an agent process autonomously. Output from the first cycle of execution is taken as input, and is imagined to have entered the next cycle. The notion of an agent life cycle can help in preserving some information from the past cycles and initiate current events based on the cycle information. The reserved word, cycle, is used to specify and keep a record of the number of cycles an agent has undergone.

**cycle**

‘identifier’

V. SUMMARY

Industrial software systems have started to use agent technologies for the development of real-time systems. The complexity involved in such systems enforces the use of formal methods to support reliable, executable and maintainable design. This paper introduced the application of formal methods with the extended Descartes specification language to support intelligent software agents. From an industrial perspective, the methodology used in this paper will provide executable specifications for an intelligent software agent design that is executable, less prone to errors, and that which satisfies the IEEE std. 830-1998 standard. From an academic perspective, this paper will serve as a strong platform for further research in the field of intelligent software agent design using a formal method. The complex inherent nature of intelligent software agents is handled with the use of a specific approach towards software development.

REFERENCES

- [1] ANSI/IEEE, IEEE std. 830-1998 Recommended Practice for Software Requirements Specification, (ANSI/IEEE), 1998.
- [2] Bagic, Marina., “Formal Infrastructure for Modeling Intelligent Agents with Agent UML and Petri Nets,” On the Move to Meaningful Internet Systems, Volume 3292, 2004.
- [3] Bauer, B., Muller, J., Odell, J., Wooldridge, M., “Agent UML: A Formalism for Specifying Multiagent Interaction,” Proceedings from Agent Oriented Software Engineering (AOSE), Paolo Ciancarini and Michael Wooldridge eds., Springer-Verlag, Berlin, 2001, pp. 91-103.
- [4] Brenner, W., Wittig, H., Zarnekow, R., Intelligent Software Agents: Foundations and Applications, Springer-Verlag New York, Inc., Secaucus, NJ, 1998.
- [5] Geiger, C., Flake, S., “CASA - Structured Design of a Specification Language for Intelligent Agents,” Proceedings of the 5th Asian Computing Science Conference on Advances in Computing Science, 1999.
- [6] Medina, M. A., Urban, J. E., “An Approach to Deriving Reactive Agent Designs from Extensions-to the Descartes Specification Language,” Eighth International Symposium on Autonomous Decentralized Systems, March 21-23, 2007, pp. 363 – 367.
- [7] Wooldridge, M., D’Inverno, M., Fisher, M., Lomuscio, A., Luck, M., De Rijke, M., Ryan, M., “Formalisms for Multi-Agent Systems,” First UK Workshop on Foundations of Multi-Agent Systems (Fo-MAS), Vol. 12, Issue 3, September 1997, pp. 315-321.
- [8] Wooldridge, M., Ciancarini, P., “Agent-Oriented Software Engineering: The State of the Art,” Agent-Oriented Software Engineering, Vol. 1957, 2001, pp 1-28.
- [9] Wooldridge, M., “Intelligent agents: Key Concepts,” from Proceedings of the 9th ECCAI-ACAI/EASSS, AEMAS, 2001, pp. 3-43.
- [10] Zhu, H., “A Formal Specification Language for Agent-Oriented Software Engineering,” Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems, 2003.

# Meta Context for Agent Planning

**Csaba Egyhazy**

Virginia Tech

Computer Science

7054 Haycock Rd., Falls Church, VA 22181, USA

[cegyhazy@vt.edu](mailto:cegyhazy@vt.edu)

## Motivation

In the Procedural Reasoning system (PRS), originally developed at Stanford research Institute by Michael Georgeff and Amy Lansky, context is the precondition of the plan, and is used for checking the current situation so as to determine whether a particular plan, among the various alternative ones, is likely to succeed in handling the event in question. A plan that has a context which evaluates as true given the agent's current beliefs is said to be applicable at that moment in time, and is a candidate for execution [1]. However, the context for a plan may involve information that is not necessarily part of an agent's beliefs. For example, there might be some contextual information related to properties of the agent community the agent belongs to [2]; we call this context "meta context."

In order to organize meta context, we propose using a multi-tiered approach in which the contextual information is segregated and encapsulated in three different levels, the *Local*, *Focal*, and *Zonal*. Each level contains specific kinds of contextual information, thereby pruning the search space and making the process of maintaining contextual information more manageable. Although this approach is not dependant on any particular implementation, it does assume that context is modeled explicitly. Each of the contextual levels is described in detail next.

## Zonal Level

The Zonal Level context represents the context that all agents in the community must share, even if their internal representations differ. Also, the Zonal Level context is the only context that all agents are required to share. As such, this context defines both the necessary and sufficient requirements for membership in the agent community.

The Zonal Level context is comprised of three features:

- Common Knowledge
- Required Behaviors
- Required Protocols

Common Knowledge is the body of knowledge that all agents in the community need in order to cooperate in a plan/schedule. This may include how to access available services, systems messages, or any other information that is deemed necessary. Required Behaviors is the set of most basic behavioral standards that all agents in the community must respect when cooperating. In essence this feature defines the "societal rules" that govern the behavior of every member of the community. Required Protocols is essentially the definition of the most basic language that members of the community can use to communicate with one another.

Since every agent in the community shares the Zonal Level context, it forms the most basic foundation for all agent interactions. In order for agents in a multi-agent system to be social and interact, there needs to be some ground rules on which to form such interactions. Such rules are often built intrinsically into the design of the system. In relatively simple systems, these intrinsic rules may be sufficient to define the entire range of member interactions. However agent communities present such a vast range of possible interactions among a heterogeneous membership, that it is nearly impossible to design the details of all such relationships into the system a priori. What is needed instead is a set of foundational requirements from which complexity can be synthesized. That is what the Zonal Level context provides.

It is by exercising the knowledge, protocols, and behaviors of the Zonal Level context that unfamiliar agents can begin to interact and learn about each other. Through these interactions, agents can discover common higher-level protocols and develop more complex relationships. The details of these relationships are what are stored in the Focal Level contexts.

## Focal Level

The Focal Level consists of the contexts of individual agents relationships with other agents in the community. Here context is built up dynamically as agents in the community interact. The exact contents of Focal Level context is specific to each agent, but must include such information as:

- The nature of the relationship
- Agreed upon protocols
- Appropriate behaviors
- Trust metrics

The nature of the relationship is important because it informs nearly every other aspect of the interactions. For instance it helps to determine which aspects of the interaction history are important, and which behaviors are appropriate or expected. Obviously, it also must play a role in determining the level of trust between the participants. Arguably, the nature of the relationship could be encoded intrinsically in the other aspects of the relationship. But storing it explicitly makes it available as a parameter to algorithms that may be common to many different kinds of relationships. These may or may not apply to a specific plan. Therefore, a mechanism must be determined by which we can mark plans that must include it as part of the plan's contextual information.

The agreed upon protocols are those protocols that were established as a result of the initial Zonal Level facilitated interactions. These may consist of anything from low-level communications protocols to high-level negotiation protocols. These are agreed upon through the course of the relationship, and they define the mechanism by which the agents interact. Of course each of the agents involved is able to support each of the agreed upon protocols, otherwise they would not have been agreed to.

The appropriate behaviors are the relationship specific behavioral algorithms that are appropriate for this relationship. Clearly these are highly dependent on the nature of the relationship. The behaviors that are appropriate when interacting collaboratively with another agent may not be appropriate at all in a competitive or service oriented relationship.

The final piece of the Focal Level context is the trust metric. This is the collection of information and algorithms that define the degree of trust that an agent applies to the entity on the other side of the relationship. In other words the trust metric is what an agent uses to evaluate the risk that the other party may deliberately interfere with the agents ability to achieve its plans. This metric is critical to dealing

with the major difference between agent communities and cooperative multi-agent systems. Specifically, it is the trust metric that allows agents to cope with the fact that some agents with whom they need to cooperate may also be in competition with them. By employing this metric an agent can evaluate how likely another party is to uphold its end of an agreement, or how risky it might be to share important information with that entity.

Because Focal Level context may vary from agent to agent, it cannot be assumed that the Focal Level context of one agent can be shared with another. Furthermore, some aspects of the Focal Level context are not appropriate to share even if sharing is possible. For instance, it doesn't make sense to share an agent's trust metrics with the other members of the relationship. The trust metrics define the level of trust assigned to agent B from agent A, and is unlikely to have an appropriate meaning for agent B assigning trust to agent A. Still, each Focal Level context describes a relationship. That is to say that it describes how two parties interact. So while it is not shared, an agents Focal Level context must be compatible with the corresponding context of the other party to the relationship. By "compatible" we mean that the Focal Level context for each party to a given relationship must:

- Represent the other parties to the relationship
- Define the interface by which all parties to the relationship can interact; and to which all parties to the relationship have agreed.

The first bullet simply indicates that the Focal Level context of parties A and B are not compatible if A's context is not for a relationship with B or if B's context is not for a relationship with A. The second bullet is basically referring to the agreed upon protocols feature of the context. In particular it means:

- The context for each party must contain the same agreed upon protocols
- Each party must be capable of using the specific protocols

This definition of compatibility is very flexible in that it does not require each party to have the same understanding of the nature of the relationship, but only that they agree on how they will communicate. Also, by this definition compatibility should be a self-assembling and self-policing feature. If the Zonal Level context provides the necessary features to begin an interaction, and the relationship and its Focal Level context are built up from the Zonal Level context through negotiation, then the corresponding contexts will naturally be compatible. Additionally, if

the Focal Level contexts are malformed or become incompatible in some other way, the corresponding relation will become either non-functioning or non-productive. By non-functioning we mean that the parties will no longer be able to communicate. By non-productive we mean that incorrect or misunderstood communication will lead to the relationship failing to carry out a plan of one or both of the parties. In either case, the relationship can be terminated and either abandoned or rebuilt from the Zonal Level.

## Local Level

The Local Level is the agent's internal storage of beliefs. These are the internal features that guide the actions of each individual agent in pursuit of their plans. Along the way, there may be occasion for agents to exchange some of this information. In a cooperative multi-agent system this would not present a problem. However, in an agent community, as has been discussed many times before, there is no guarantee that another agent will be cooperative. Even if an agent is cooperating now, that agent may prove to be a competitor later on. With this uncertainty, an agent must be selective about what information it shares and with whom. If it is not selective enough, an agent may find that it has revealed information that causes it to be blocked from achieving its goals. On the other hand, if an agent is too restrictive about sharing information it may not be able to form the cooperative relationships it needs in order to advance its intents [3]. This is where the Focal Level trust metrics come into play.

The purpose of the trust metric is to inform a risk versus reward evaluation of sharing a piece of information with another party. The trust metric uses information gained through a history of interactions to evaluate how likely it is that another entity can be trusted. In order to evaluate the potential risks or rewards of sharing information and participating in a plan, the agent must have understood three critical features of every piece of data:

- Its value
- Its sensitivity
- Its importance

The value of the data is an inverse valuation of the availability of the information. Readily available information is not very valuable, and rare information is more valuable. The sensitivity of information is a measure of how likely it is that it could be used to prevent the agent's plan from succeeding. The importance of a piece of data refers to how important

it is to the agent's plan. This metric is primarily used in evaluating the reward of gaining a given piece of information against the cost, for example, of sharing another piece of information. In addition to these data valuations, an agent must also be able to rank the importance of its plans. This is important to both the risk and reward calculations. For instance sharing risky information (high value and high sensitivity) that relates to a low ranked plan may be worth the reward of receiving high value, high importance information that is necessary for a highly ranked plan.

## References

1. Turner, R.M., Context-Mediated Behavior for Intelligent Agents. *International Journal of Human-Computer Studies*, 1998. 48(3): p. 307-330
2. Sandip Sen, S.S., Stephane Airiau, Teddy Candale, and D.C. Dipyaman Banerjee, Partha Mukherjee, Anil Gursel. *Robust Agent Communities*. in *AIS-ADM*. 2007: Springer-Verlag.
3. Bucur, P.B., Olivier B.. *What Is Context and How Can an Agent Learn to Find and Use it When Making Decisions?* *CEEMAS*. 2005: Springer-Verlag.

# Software Components Search Approaches in the Context of COTS-based Development

Nacim Yanes<sup>#1</sup>, Sihem Ben Sassi<sup>#2</sup>, Henda Hajjami Ben Ghezala<sup>#3</sup>

<sup>#</sup>*RIADI-GDL Laboratory, National School of Computer Sciences, Manouba University  
La Manouba 2010, Tunisia*

<sup>1</sup>*Nacim.yanes@riadi.rnu.tn*

<sup>2</sup>*Sihem.bensassi@isetcom.rnu.tn*

<sup>3</sup>*Henda.bg@cck.rnu.tn*

**Abstract**— COTS-based development is a component reuse approach promising to reduce costs and risks, and ensure higher quality. Searching COTS components to identify potential ones for a later reuse is a critical activity, having to cope with some challenging marketplace characteristics related to its widespread, evolvable and growing nature. In this paper, we present an overview of software component search approaches through the works classifying them. We find that most of them assume an in-house reuse and don't take into account the availability of the net sources of software components. We note also absence of other aspects as classification criteria. We then propose our classification of software components search approaches. Finally, we study the existing COTS components search approaches according to aspects of the proposed classification and the factors influencing the search of such components.

**Keywords**— Software reuse classification, component search, COTS component identification, COTS marketplace.

## I. INTRODUCTION

Prior research on software reuse has focused on searching and identifying software components from large repositories. Over the past fifteen years, a large number of solutions for the search problem have been proposed. However, most of these works have assumed that reuse would take place in-house, with centralized repositories that are out of the COTS reality. The special nature of the COTS component has taken the original concept of reuse into a completely different arena. As consequence, traditional search approaches break down in the context of COTS-based development [1] and efficient means to search and identify COTS components candidates from on-line repositories are still a major software reuse research topic.

The remainder of this paper is structured as follows: section 2 enumerates the different classifications of search approaches identified in the software reuse literature. Section 3 describes our proposal to classify software components search approaches. Section 4 deals with the COTS component search and identification. Finally, the paper ends with a conclusion.

## II. SOFTWARE REUSE CLASSIFICATION AND SEARCH

Software component search constitutes a key element to make development with reusable components more convenient than development from scratch; in fact, as stated by Barringer [2]

“You must find it before you can reuse it”. Over the past fifteen years, a wide range of solutions to the software components search have been proposed and implemented. At different times, based on available software reuse systems and also based on researchers' criteria, software reuse classification and search approaches have been classified differently [3].

Ostertag et al. classify the software component search approaches in three types that are [4]: (1) free-text keywords based approaches, (2) faceted index approaches, and (3) semantic-net based approaches semantically classify and search software components.

Mili et al. group the search approaches into four different types [5]: (1) simple keyword and string match, (2) faceted classification and retrieval, (3) signature matching, and (4) behavior matching.

Bouchachia et al. suggest a different classification [6]. They distinguish between formal and informal search approaches. (1) The formal approaches consist of specification, signature and behavior matching. (2) The informal approaches rely on the use of natural language, which are hypertext, knowledge base and information retrieval approaches.

Sugumaran and Storey combine Ostertag et al. and Mili et al. observations and come out with five types of software components search approaches [7]. The classification and search schemes can be grouped into (1) keyword search, (2) faceted classification, (3) signature matching, (4) behavioral matching, and (5) semantic-based method.

Khayati classifies search approaches into two categories [8]: (1) browsing-based search approaches and (2) query-based search approaches. These latter include four software component search techniques namely internal classification techniques, external classification techniques, structural classification techniques, and behavioral classification techniques.

Maslita and Siobhan suggest two main categories classification of search approaches [9]: (1) descriptive classification approaches with support from natural language. The approaches range from free-text keyword, faceted classification, other classification scheme, and knowledge based. On the other hand, (2) property-based approaches focusing on property of software components are behavioral

sampling, signature matching, specification matching and recommendation approach.

Ramadour and Cauvet propose the following classification [10]: (1) External classification approaches represent the components with an external description. They consist of keyword-based, facets-based and natural language-based approaches. (2) Static classification approaches represent components with a structural description. Two types of descriptions are used to perform matching: signatures (type information) and specification (behavior information). (3) Dynamic classification approaches represent the components with information about its execution results. (4) Browsing search within a library. (5) Combined approaches use two or more approaches presented above.

Ben Khelifa and al. suggest three dimensions to classify software components search approaches [11]: (1) the first dimension groups browsing-based and query-based approaches, (2) the second dimension groups internal and external search approaches, and 3) the third dimension includes structural and behavioral search approaches.

### III. PROPOSED CLASSIFICATION OF SOFTWARE COMPONENTS SEARCH APPROACHES

Although several classifications of software component search approaches were proposed in the literature, we observe minor differences between them. In fact, the majority of works are based on the same criterion to classify search approaches. This criterion is the technique used to index and retrieve software components. On the other hand, none of the proposed classifications distinguish between searching software components from in-house repositories and from on-line repositories. Furthermore, these classifications do not focus on personalization aspect. They do not differentiate between search approaches exploiting information about users during retrieval or query expansion and search approaches which do not take into account knowledge on users requirements.

These observations lead us to propose a new classification of software component search approaches. Our classification is based on the homogenization of previous classifications. In addition, it takes into account the personalization aspect and intra-organizational/extra-organizational aspect. As a consequence, it classifies search approaches according to six aspects:

#### A. Browsing/Interrogation Aspect

This aspect defines the user interaction method with the system. We distinguish two types of search approaches.

1) *Query-based approaches*: they consist in identifying software components which satisfy predefined matching criteria. Its main operation is thus the satisfaction check or matching. The criteria are usually given by an arbitrary user-defined query which is matched against the candidates' descriptors.

2) *Browsing-based approaches*: they consist in inspecting software components candidates for possible extraction, but without a predefined criterion. Its main operation is thus

navigation which determines in what order the components are visited and whether they are visited at all.

#### B. Internal/External Aspect

This aspect focuses on the description model used to characterize software components. It defines two types of search approaches, namely internal and external approaches.

1) *Internal Approaches*: they don't represent software components with an external description. They use information retrieval and indexing technology to automatically extract keywords from code source and documentation such as comments.

2) *External Approaches*: they rely on searching within external descriptions representing software components. They can be subdivided in two types: (1) keyword-based approaches which basically use information retrieval and indexing technology to automatically extract keywords from software documentation and index items with these keywords. The keyword-based approach is simple, but it is limited by lack of semantic information associated with keywords, thus it is not a precise approach. (2) The facet-based approaches which have been introduced by Prieto-Diaz [12]. They are based on experts who extract keywords from components descriptions, and arrange them by facets into a classification scheme, which is used as a standard descriptor for software components. Facet-based approaches are proven to be very effective in identifying software components from repositories, but these approaches are labor intensive.

#### C. Structural/behavioral Aspect

This aspect describes the static and dynamic properties of software components. It groups structural and behavior-based search approaches.

1) *Structural approaches*: Structural search approaches represent software components with a structural description. They can be subdivided in two types: (1) Signature matching approaches which represent software components with a set of signature characteristics. They focus on the type and number of arguments defined for component methods. Signature matching approaches are good because no additional information is needed but it is difficult to map user requirements to code signature. Furthermore, the signature does not guarantee expected component characteristics [9]. (2) Specification matching approaches which are introduced to overcome the problem of signature matching approaches. In fact, specification matching approaches characterize more precisely the semantics of a component rather than to be limited to the signature. They are based on descriptions of the components behavior with the method's pre and post conditions.

2) *Behavioral approaches*: Behavior-based approaches represent software components with information about its execution results. When the required behavior is observed, appropriate components are provided to the user. Software component search approaches based on behavior identify

software components through executing samples of input and a list of the output to be produced [13][14].

#### D. Term/Knowledge Aspect

Term/Knowledge aspect focuses on the semantic information used to search software components. This aspect defines two types of search approaches.

1) *Term-based approaches*: they are limited to the lexical and syntactic level to index and search software components. They don't support any mechanism providing additional knowledge for "understanding" semantics of software components. Term-based approaches are not precise since they do not use semantic information; they often result in too many or too few hits. They may also retrieve components that are entirely unrelated with users needs.

2) *Knowledge-based approaches*: they use semantic information to describe software components and the relationships between them. Knowledge-based approaches use technologies such as ontologies to take into account the semantics of software components and their application domains. They provide greater query flexibility and user satisfaction.

#### E. Intra-organizational/extra-organizational Aspect

The intra-organizational/extra-organizational aspect focuses on the applicability level of software components search approaches. This aspect defines two types of approaches.

1) *Intra-organizational approaches*: they are oriented to intra-organizational environments where the reusing organization has control on the evolution of the functionality and assumptions of the software components.

2) *Extra-organizational approaches*: they take into account the availability of world-wide source of software components; they extend the software reusable library to the World Wide Web.

#### F. Personalization Aspect

The personalization aspect focuses on representation and exploitation of users profiles during search process. It defines two types of approaches.

1) *Non user-centered approaches*: they do not consider users as internal component of search process. They do not exploit information about users during retrieval or query expansion.

2) *User-centered approaches*: they take into account knowledge on users requirements and profiles. In fact, in the last few years, many works have focused on personalization [10]. Several user personalization techniques developed in information research domain have been adapted to components retrieval. Those approaches make explicit the representation of users. It is possible for the user to partially specify his/her requests, which will automatically be enriched by his/her profile before their execution.

## IV. COTS COMPONENT SEARCH AND IDENTIFICATION

As we stated above, software components search approaches have received a good deal of attention in software reuse. A large number of solutions for the search problem have been proposed. However, most of these works have assumed that reuse would take place in-house, with centralized repositories that are out of the COTS components reality. The special nature of COTS component has taken the original concept of reuse into a completely different arena. COTS marketplace nature radically changes the "classical" component repository reuse schema to a global reuse approach where the marketplace acts as a global, dynamic, distributed, and heterogeneous set of repositories; and where nobody has knowledge or control of the available repositories [1]. As consequence, traditional search approaches break down in the context of COTS-based development [1] and efficient means to search and identify COTS components candidates from on-line repositories are still a major software reuse research topic. In this section, we first enumerate factors influencing the COTS component search. Then, we study some proposals addressing the problem of searching COTS components to assess to which extent they are able to satisfy COTS consumers.

### A. Factors Influencing the COTS Component Search

Various factors influence the search and identification of COTS components marketed on the Web. Any approach searching for COTS components should take into account these factors in order to be effective and accomplish its objectives:

1) *Uncontrolled COTS marketplace basis*: in recent years, several on-line commercial repositories with public access, such as [componentsource.com](http://www.componentsource.com)<sup>1</sup> and [componentace.com](http://www.componentace.com)<sup>2</sup>, have started to publish information on available COTS components and to work as a commercial channel for their distribution. These repositories offer continuously their products without any direct control and no one has great influence over the speed, content, or variety of these products [1].

2) *Growing size of the COTS marketplace*: new and improved COTS components and technologies are continuously offered. Consequently, existing market segments offer more and more products, and new market segments are continuously emerging [1]. Mobile technologies are a good example of both situations.

3) *Rapid changes in the COTS marketplace*: new versions of existing products are released every few months. Moreover, market segments frontiers move slightly over the years, making products to offer services that initially were seen as belonging to different segments. For instance, current mail server systems usually provide instant messaging facilities, even video-conferencing services [1].

---

<sup>1</sup> <http://www.componentsource.com>

<sup>2</sup> <http://www.componentace.com>

4) *Lack of standards for COTS descriptions*: currently, the amount of information available about COTS is widespread, vast and still growing. COTS component editors/vendors do not have a standard for describing components, which results in a variety of documentation styles very difficult to compare. This fact does not allow performing automated or at least assisted search [15]. A study conducted in [16] evidenced that the required COTS information is highly incomplete, widespread and unstructured, becoming very difficult to obtain.

5) *Dependencies among COTS*: many dependencies among COTS components exist, either for enabling, enhancing, or complementing their functionality [17]. However, commonly, these dependencies are not explicitly declared. It may result in several over-costing and integration problems [1].

### B. COTS Component Search Approaches

Even though some approaches for searching software components on the Web are proposed, these proposals are generally addressing source code components and consequently they are out of the COTS components reality. Few proposals addressing the COTS component search problem were proposed in the literature. To our knowledge, only two search approaches exist. Table I illustrates the positioning of both these approaches according to our classification.

1) *COTS Search Approach by Sugumaran and Storey*: Sugumaran and Storey propose in [7] a semantic-based approach for retrieving COTS components. This approach makes use of domain models containing the objectives, processes, actions, actors, and an ontology of domain terms, their definitions, and relationships with other domain-specific terms. The user specifies the requirements for the components that he/she is interested in using natural language (nominal or imperative sentences). An initial query is generated and augmented with domain specific information derived from domain models and takes into account the context within which the user is performing the search. Keywords from the user's query are mapped to the domain ontology and appropriate terms derived that are used in the search query. The ontology is also used to expand the query with synonyms to broaden the search, if needed. This approach also considers relationships among components when retrieving them, thus resulting in a consistent set of components.

Although this semantic-based approach provides a natural and flexible way for the user to specify requirements for COTS component search, it presents several limitations. First, it assumes the availability of COTS components descriptions in a centralized repository, which is not the COTS reality. In fact, this approach does not take into consideration the indexing process of COTS components published on heterogeneous on-line repositories. Second, it assumes that the domain model and ontologies exist and are accessible. Third, the authors do not take into account the growing nature of the COTS marketplace. Indeed, they don't mention how their

approach could maintain the relationships between various COTS components so that a consistent set of components can be retrieved. Fourth, the approach does not provide any personalization mechanism for customizing COTS components to meet the specific needs of the user.

TABLE I. POSITIONING OF COTS SEARCH APPROACHES

Aspects		Sugumaran and Storey	Chen et al.
Aspect 1	Browsing		✓
	interrogation	✓	✓
Aspect 2	Internal		
	External	✓	✓
Aspect 3	Structural		
	Behavioral		
Aspect 4	Term		✓
	Knowledge	✓	
Aspect 5	Intra-organizational		
	Extra-organizational	✓	✓
Aspect 6	Non user-centered	✓	✓
	User-centered		

2) *COTS Search Approach by Chen et al*: Chen et al. propose in [18] an hybrid approach to classifying COTS components, which adopts two kinds of classification schema. One classification schema is based on the systematic classification. The other classification schema is based on the faceted classification. Both schemas are defined using the XML Schema. Systematic classification schema is search engine oriented. According this schema, component descriptor offered by the COTS component provider supports users to search for COTS component roughly by using the search engine. Faceted classification schema is retrieval engine oriented. Component provider describes the COTS components based on this schema and offers the component descriptors that facilitate user to retrieve components accurately with the aid of the retrieval engine. The process of acquiring the suitable components contains two phases: search and retrieval phases. Firstly, users search the candidate COTS components roughly with the aid of search engine. And then they retrieve components accurately among the candidates by using the retrieval engine. That is to say, these two kinds of engines establish a two-level query mechanism together. When describing the component referring to hybrid classification, the component provider must generate the descriptors based on two classification schema respectively. As a result, there exist two descriptors based on different schema for each component. One descriptor based on the systematic classification contains the basic information used for rough search. The other descriptor based on the faceted classification contains the information about basic information and domain information, which is helpful for accurate retrieval.

However, this approach also presents some limits. First, it does not use semantic information to describe COTS components and the relationships between them. Thus, this search approach is not precise; it could often result in too many or too few hits. It may also retrieve COTS components that are completely unrelated to users needs. Second, this

approach depends entirely on the involvement of COTS components providers. As a matter of fact, the number of COTS components indexed, in this approach, depends on the component descriptors offered by the COTS component providers. Third, authors of this approach do not mention which kind of information characterizes COTS components. Fourth, similarly to the approach proposed by Sugumaran and Storey, it does not provide any personalization mechanism allowing the representation of users as internal component of search process and consequently customizing COTS components results according to the user domains of interest.

#### V. CONCLUSION AND FUTURE WORK

In this paper, we first proposed a classification of software components search approaches. Our classification is based on the homogenization of previous classifications identified in the literature. In addition, it takes into account the personalization aspect and intra-organizational/extra-organizational aspect. We then studied some proposals addressing the problem of searching COTS components to assess their effectiveness. We concluded that these proposals still remain insufficient to the identification step of the CBD process.

We are now focusing on the development of an intelligent, user-centered approach for searching COTS components overcoming the studied search approaches limitations and taking into consideration aforementioned factors. Our initiative is intended to manage an evolvable COTS component ontology for the unification of heterogeneous COTS descriptions and application domains ontologies to represent and store COTS specific knowledge. We also anticipate integrating, in our search approach, a user model representing the preferences and interest domains of COTS component users so as to fit as much as possible their needs.

#### REFERENCES

- [1] Ayala, C.P.: Systematic Construction Of Goal-Oriented COTS Taxonomies, PHD thesis (2008)
- [2] Barringer, H., Cheng, J. H., Jones, C. B.: A Logic Covering Undefinedness in Program Proofs. In: *Acta Informatica*, 21(3), pp. 251 – 269 (1984)
- [3] Haining Yao Letha Etkorn.: Towards A Semantic-based Approach for Software Reusable Component Classification and Retrieval. In: *ACM Transaction on Software* (2004)
- [4] Ostertag, E., Hendler, J., Prieto-Díaz, R., Braun, C.: Computing Similarity in a Reuse Library System: an AI-based approach. In: *ACM Transaction on Software* (1992)
- [5] Mili, R., Mili, A., Mittermeir, R.T.: A Survey of Software Storage and Retrieval, *Annal of Software Engineering*, vol. 5, no. 2, pp. 349-414 (1998)
- [6] Bouchachia, A., Mittermeir, R. T., Pozewaunig, H.: Document Identification by Shallow Semantic Analysis, lecture notes, in computer science, pp. 190 – 202 (2000)
- [7] Sugumaran, V., Storey, V.: A Semantic-based Approach to Component Retrieval, *The Data Base for advances in Information Systems*, Volume 34, N°3, (2003)
- [8] Khayati, O. : Modèles Formels et Outils Génériques pour la Gestion et la Recherche de Composants, PHD thesis (2005)
- [9] Maslita, A.A., Siobhán N.: Retrieving Software Component using Clone Detection and Program Slicing, the University of Sheffield, Memorandum CS-07-03 (2007)
- [10] Ramadour, P., Cauvet, C.: An Ontology-based Reuse Approach for Information Systems Engineering. In: *IEEE International Conference on Signal Image Technology and Internet Based Systems* (2008)
- [11] Ben Khalifa, H., Khayati, O., Hajjami Ben Ghezala, H.: Une Technique Structurelle Comportementale pour la Recherche de Composants. In: *Second International Conference SIIIE* (2009)
- [12] Prieto-diaz, R.: Implementing Faceted Classification for Software Reuse (Experience Report). In: *12th International Conference on Software Engineering*, Nice, France, IEEE Computer Society Press, pp. 300–304 (1990)
- [13] Podgurski, A., Pierce, L.: Retrieving Reusable Software by Sampling Behavior. In: *ACM Transactions of Software Engineering and Methodology*, pp. 286 – 303 (1993)
- [14] Hall, R. J.: Generalized Behaviour-based Retrieval. In: *International Conference on Software Engineering ICSE93*, Baltimore (1993)
- [15] Cechich, A., Réquillé-Romanczuk, A., et al.: Trends on COTS Component Identification and Retrieval. In: *Proceedings of the International Conference on COTSBased Software Systems*. IEEE Computer Society (2006)
- [16] Bertoa, M.F., Troya, J.M., Vallecillo, A.: A Survey on the Quality Information Provided by Software Component Vendors”. In *Proceedings of the 7th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering* (2003)
- [17] Franch, X., Maiden, N.: Modelling Component Dependencies to Inform their Selection”. In: *Proceedings of the 2nd International Conference on COTS-Based Software Systems*. Lecture Notes in Computer Science. Volume 2580/2003 pp. 81-91 (February 2003)
- [18] Chen, H., Ming, Z. Ying, S.: An Approach to Manage and Search for Software Components. In: *5th WSEAS International Conference on Applied Computer Science*, pp358-363. IEEE press, Hangzhou, China (2006)

# Architecture-centric development and evolution processes for component-based software

Huaxi (Yulin) Zhang, Christelle Urtado, Sylvain Vauttier  
LGI2P / Ecole des Mines d'Alès – Nîmes – France

{Huaxi.Zhang, Christelle.Urtado, Sylvain.Vauttier}@mines-ales.fr

**Abstract**—Component-based development focuses on component reuse and composition: abstract components (as wished) must be searched for and matched to existing component (as found). This search and reuse activity greatly impacts software development and evolution processes. Unfortunately, very few works propose adaptations of traditional software engineering processes and no existing ADL yet permits to describe the resulting development artifacts. This paper proposes architecture-centric processes for the development and evolution of component-based software. Architecture-centric development produces descriptions for architecture specification, architecture configuration and component assembly. The paper shows how Dedal, a three-level ADL, can be used to support the consistent description of these three artifacts. The paper then shows how these descriptions can be used during a controlled architecture-centric evolution process that helps build, test and record versions of component-based software. This tackles the well-known issues of architecture erosion and drift that denote mismatches between the different architecture definitions.

## I. INTRODUCTION

Component-based software engineering (CBSE) is the major technical response to the increase in software system complexity and the ever growing need to decrease development time and cost without giving up quality. It promotes a reuse-based approach to define, implement and compose loosely coupled independent software components into whole software systems [1], [2]. Software architectures can be described at three development stages: architecture specification, architecture configuration and instantiated component assembly. Conformance between these descriptions must be guaranteed top-down from an abstract description level to the next, more concrete one. Such descriptions could also be used to control software evolution by propagating changes bottom-up. Surprisingly, no architecture description language (ADL) proposes such a detailed description for architectures that covers the artifacts produced during the component-based development cycle. Most ADLs, such as Wright [3], [4], C2SADEL [5], [6], Darwin [7], focus on modeling one or two architecture descriptions levels. Furthermore, no ADL is rich enough to serve as the support of a complete evolution process for component-based software. This paper first presents an architecture-centric development process, based on intensive component reuse. It defines how architectures can be gradually defined, thanks to three-leveled architecture definitions which capture the design decisions at each step of the process. More specifically, explicit architecture specifications provide

a means to effectively integrate the reuse of components as part of the development process. A controlled architecture-centric evolution process is then presented. The aforementioned three-leveled architecture definitions are used to check the consistency of changes and to manage the propagation of changes when architecture definitions are versioned to prevent architecture drift and erosion [8]. The features of Dedal – our proposed ADL – are specifically designed to support these two processes.

The remaining of this paper is organized as follows. Section II defines the proposed architecture-centric development process and its associated architecture descriptions. Section III presents how the Dedal ADL supports the three proposed architecture descriptions. Section IV depicts the context of software evolution in CBS before Sect. V defines a controlled architecture-centric evolution process and its support with Dedal. Section VI concludes and draws future work directions.

## II. ARCHITECTURE-CENTRIC DEVELOPMENT FOR CBS

### A. An architecture-centric development process

Component-based software development is characterized by its implementation of the “reuse in the large” principle. Reusing existing (off-the-shelf) software components therefore becomes the central concern during development. In the context of component-based software development, traditional software development life-cycles have to be adapted [2]. Figure 1 illustrates our vision of such a development life-cycle which is classically divided in two: the component development life-cycle (sometimes referred to as component development *for* reuse), which is not detailed here, and the component-based software development cycle (referred to as component-based software development *by* reuse) that describes how previously developed software components can be used for new software development (and how this reuse process impacts the way software is built).

Our component-based software development life-cycle deliberately is an architecture-centric development process focusing on the produced architecture artifacts (architecture descriptions as models of the software) for each development step. In this component-based software development life-cycle, software is considered to be produced by the reuse of components that have previously been stored and indexed in a component repository. After a classical requirement analysis step, architects establish the abstract architecture specification.

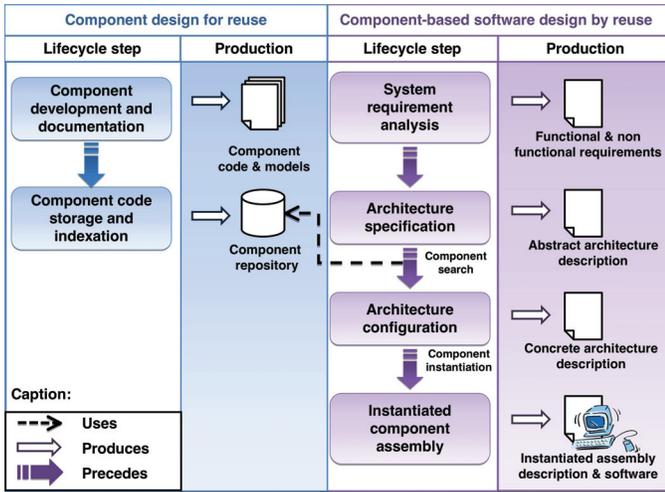


Fig. 1. Component-based software development process

They define which functionalities should be supplied by components, which interfaces should be exported by components, and how interfaces should connect to build a software system that meets the requirements. In a second step, architects create architecture configurations that define the sets of component implementations (classes) by searching and selecting from the component repository. Abstract component types from the architecture specification then become concrete component types in architecture configurations. In a third step, configurations are instantiated into component instance assemblies and deployed to executable software applications.

The claim of this paper is that **an architectural description should correspond to each of the three steps (specification, configuration and assembly)** of the component-based software development process. These three point of views on architectures are necessary to reflect the architect’s design decisions and should be expressed using an adequate ADL.

### B. Development cycle coverage by existing ADLs

Software system architectures [9] gather design decisions on system. They are expressed using an ADL. In the syntax of most ADLs, architectures are usually described by two complementary views: an architecture specification where a class of systems is described as composed of component classes and connector classes, and an architecture configuration where a system instance is described as composed of component instances and connector instances.

Systems can also be described at various steps of their life-cycles. To our knowledge, no ADL really includes this time dimension. Some works such as UML [10] or Taylor *et al.* [9] implement or describe close notions. UML makes it possible to describe object-oriented software at various life-cycle steps but this capability is not transposed in their component model. Taylor *et al.* distinguish two description levels for architectures at design and programming time, respectively called perspective and descriptive architectures.

Garlan *et al.* [11] points out the importance of three levels (called task, model and runtime layers) for dynamic software evolution management but, as far as we know, do not propose any ADL or metamodel to concretely implement them. Other existing ADLs such as C2SADEL, Wright, Darwin, Unicon [12], SOFA2.0 [13], Fractal ADL [14] and xADL2.0 [15] do not either (see Table I).

As a conclusion, we found no research work that enables to model the three levels of software systems as architectural descriptions which correspond the development stages.

ADL	Specification	Configuration	Assembly
C2SADEL	✓	✓	×
Wright	×	✓	×
Darwin	×	✓	×
Unicon	×	✓	×
SOFA 2.0	×	✓	×
Fractal ADL	×	✓	×
xADL 2.0	×	✓	✓

TABLE I  
EXPRESSIVENESS OF EXISTING ADLs

### C. Example of a Bicycle Rental System

Figure 2 shows the example used throughout the paper: the architecture specification of a bicycle rental system (BRS). A *BikerGUI* component manages a user interface. It cooperates with a *Session* component which handles user commands. The *Session* component cooperates with the *Account* and *Bike&Course* components to identify the user, check the balance of its account, assign him an available bike and then calculate the price of the trip when the rented bike is returned. In the following, we will use a part of this system (*BikeCourse* and *BikeCourseDB* component roles and their connection) to illustrate our concepts and ADL syntax.

### III. OVERVIEW OF THE DEDAL ADL

In this section, we present elements of the syntax of the Dedal ADL [16] we propose to describe the artifacts produced at each of the three stages of component-based software development.

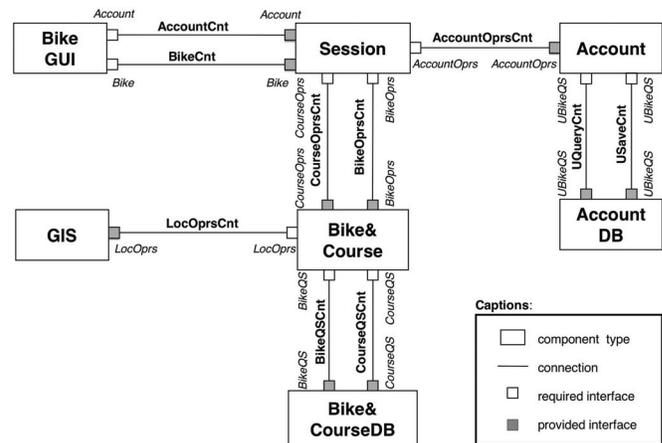


Fig. 2. BRS abstract architecture specification

## A. Abstract Architecture Specifications

**Abstract architecture specifications** (AASS) are the first level of software architecture descriptions. They provide a generic definition of the global structure and behavior of software systems according to previously identified functional requirements. They are specified during the design step of software and serve as a basis to create concrete architecture configurations. These architecture specifications are abstract: they do not identify concrete component types that are going to be instantiated in the software system. They only describe “ideal” component types from the application point of view. Abstract architecture specifications can be compared to perspective architectures as denoted by Taylor *et al.* [9].

In Dedal, an AAS is composed of a set of component roles and a set of connections. The architecture version number and versioning information are also part of the abstract architecture specification description (see Sect. V). Figure 3 provides an example of the AAS for the BRS. For readability reasons, this description represents only a small part of the BRS AAS depicted in Fig. 2.

```
specification BRSSpec
component_roles
  BikeCourse; BikeCourseDB
connections
  connection BikeQSCnt
    client BikeCourse.BikeQS
    server BikeCourseDB.BikeQS
  connection CourseQSCnt
    client BikeCourse.CourseQS
    server BikeCourseDB.CourseQS
version 2.0;
pre_version 1.0;
by additionGISList;
```

Fig. 3: AAS of the BRS (partial)

**Component roles** model abstract component types in that they describe the roles components should play in the system. A component role lists the interfaces (both required and provided) the component should expose<sup>1</sup>. As component roles are abstract component specifications, Dedal describes them outside architecture specifications, so as they can be reused from a specification to another (which would not be possible if they were embedded). Figure 4 shows the description of the *BikeCourse* component role.

## B. Concrete Architecture Configurations

**Concrete architecture configurations** (CACs) are the second level of system architecture descriptions. They result from the search and selection of real component types (component classes) from a component repository. These component classes must match abstract component descriptions from the architecture but need not be identical; compatibility is sufficient. CACs describe the architecture from an implementation

<sup>1</sup>Dedal further includes the description of architecture dynamics using behavior protocols that are not presented here due to space limitations [16].

```
component_role BikeCourse
required_interfaces BikeQS; CourseQS; LocOprs
provided_interfaces BikeOprs; CourseOprs
```

Fig. 4: BikeCourse component role

viewpoint (by assigning component roles to existing component types). CACs correspond to descriptive architectures as denoted by Taylor *et al.* [9].

CACs list the concrete component and connector classes which compose a specific version of a software application. The architecture of a given software is thus defined by a unique AAS and possibly several CACs (each of which must conform to the AAS). This means that each component or connector class used in an architecture configuration must be a legal implementation of the corresponding component role or connection in the architecture specification. The configuration version number and versioning information are also part of the concrete architecture configuration description (see Sect. V). An excerpt of the BRS configuration derived from the specification presented on Fig. 2 and 3 is described in Fig. 5.

```
configuration BRSSConfig
implements BRSSpec (2.0)
component_classes
  BikeTrip (1.0) as BikeCourse;
  BikeCourseDBClass (1.0) as BikeCourseDB
version 2.0;
pre_version 1.0;
by additionStationDataList;
```

Fig. 5: A possible CAC for the BRS

**Component classes** used in the CAC are listed and each of them is associated to the component role it implements. Component classes can either be primitive or composite. *Primitive component classes* are implemented by a single class. *Composite component classes* are coarser grained components which implementation is a configuration. Component classes can have (externally visible) attributes so as to be able to express constraints on component instance states.

**Conformance** between an AAS and a CAC is a matter of conformance between component roles and the component classes that supposedly implement them. Many conformance relations could be defined, from stricter to very loose ones. On the one hand, we defend that reused components need not be exactly identical to specifications because being too strict in this matter might seriously decrease the number of reuse opportunities. On the other hand, it is expected from a conformance relation that it enables verifications that guarantees good chances that the thought component combination will execute. The rule of the thumb that can be used is that concrete components must provide at least what the specification declares it provides and require less than what the specification already requires.

## C. Instantiated Software Component Assemblies

**Instantiated software component assemblies** (ISCAs) are the third level of architectures. They result from the instan-

tiation of the component classes from a configuration. They provide a description of runtime software systems and gather information on their internal states thus enabling the record of state-dependent design decisions [17].

ISCAs list the component and connector instances that compose a runtime software system, the attributes of this software system, and the assembly constraints the component instances are constrained by. The assembly version number and versioning information are also part of the instantiated software component assembly description (see Sect. V). Figure 6 gives the description of a software assembly that instantiates the BRS architecture configuration of Fig. 5.

```
assembly BRSAss
instance_of BRSConfig (2.0)
component_instances
  BikeTripCl as BikeCourse;
  BikeCourseDBClassCl as BikeCourseDB
assembly_constraints
  BikeTripCl.currency=="Euro.";
  BikeCourseDBClassCl.company==
  BikeTripCl.company;
version 2.0;
pre_version 1.0;
by additionStationDataInsList;
```

Fig. 6: ISCA for the BRS

**Component instances** document the instances of components that constitute the runtime software system. They are instantiated from the corresponding component classes of the architecture configuration. They might contain constraints on components' attributes that reflect design decision that impact component states (attribute values).

**Assembly constraints** define conditions that must be verified by attributes of some component instances of the assembly, to enforce its consistency. Such assembly constraints are not mandatory. Assembly constraints are illustrated on the example of Fig. 6 where the value of the *currency* attribute of component *BikeTripCl* is fixed to the "Euro" value. These constraints are very simple and do not yet enable the expression of alternatives, negation, nor the resolution of possible conflicts.

**Conformance** between a CAC and an ISCA is quite straightforward. All component instances of the assembly must be an instance of a corresponding component class from its source configuration (and reciprocally). Conformance also includes the verification that attribute names used in an assembly constraint of some component assembly pertain to the component classes the components of the assembly are instances of. For example, the assembly constraint *BikeTripCl.currency="Euro."* of Fig. 6 entails that the *BikeTrip* component class (from which *BikeTripCl* is instantiated) must possess a *currency* attribute.

#### IV. CONTEXT OF ARCHITECTURE-CENTRIC EVOLUTION

##### A. Requirements of architecture-centric evolution

Component-based software development has been widely studied during recent years, while there has been less effort to

study component-based software evolution. *Software evolution* is defined as the collection of all programming activities intended to generate a new version of some software from an older operational version [18]. In the context of component-based development, a software evolution process must be re-invented as claimed by [19], [20].

To our opinion, software evolution should possibly be triggered from any architecture level of component-based software (specification, configuration or assembly). Indeed, depending on contexts, evolution can occur during specification or design (e.g. if the evolution management process is integrated into some development environment), or directly at runtime (e.g. if the evolution management process is to be used by some autonomic software) [21]. For example, the specification of some software architecture can be required to evolve to meet new requirements, thus adding a new component role to provide new functionalities. The architect might also want the configuration to evolve at design time, for example to replace some component class by another compatible one that has better qualities (e.g. cheaper, more efficient, better maintained or safer). Changes might also occur at runtime as for example when some component fails and must be replaced by another one that provides similar (or sometimes only acceptable replacement) functionality. Changes, either they be triggered from the specification, the configuration or the assembly description levels, must nonetheless be managed adequately for all three description levels to be kept consistent with one another. This entails architecture re-engineering or architecture re-factoring.

##### B. Evolution support in existing ADLs

Evolution in existing ADLs typically characterizes by five facts. Firstly, few ADLs enable to *describe changes*, except C2SADEL and Darwin which both use change transactions that apply on the runtime system. They have no formal description of changes as first class information in ADL syntax. Secondly, evolution can only be *triggered from the configuration level* (e.g. C2SADEL, Darwin, Wright). As these ADLs do not describe running assemblies, they cannot enable changes to be triggered from this level. Thirdly, some *consistency checks* are performed (e.g. C2SADEL, Wright), but none of them checks all consistencies of architectures: name, behavior, interface, interaction and refinement. Then, *change propagation* is limited. Evolution processes should handle all three description levels of software architectures. Some change at one of the three levels should be propagated to the other two in order to maintain all descriptions consistent and prevent architecture drift and erosion. This propagation should possibly be directed top-down (from specification to assembly) or bottom-up (from assembly to specification). To our knowledge, the few existing ADLs that support change propagation (e.g. xADL2.0) only direct it in a top-down manner. This is adapted to traditional software development and evolution but could be improved in the case of open and dynamic component software (as autonomous systems). At last, *versioning* components and architectures is not frequent except SOFA2.0 and MAE [22].

ADL	Consistency checking	Evolution test	Change propagation	Versioning
C2SADEL	Refinement	×	×	×
Wright	Name, interaction, deadlock	×	×	×
Darwin	State	×	×	×
SOFA2.0	Behavior	×	×	State-based
xADL2.0	×	×	Horizontal (topdown)	×
MAE	Subtyping	Perfective	✓	Change-based

TABLE II  
EVOLUTION SUPPORT IN EXISTING ADLS

```

change additionStationData
time dynamic
level configuration
operation addition
artifact component_class is StationData
purpose perfective
origin given

```

Fig. 7: Change description example

MAE uses configuration management to control component versions, but provides no version support for whole architecture configuration or specifications. In conclusion, most ADLS provide a limited support for evolution as shown in Table II.

## V. ARCHITECTURE-CENTRIC EVOLUTION FOR CBS

In this section, we present our vision of a component-based software evolution process based on Dedal. Architecture evolution can be triggered by changes at any of the three representation levels. Moreover, both top-down (re-factoring) and bottom-up (re-engineering) change propagation are supported. Classically, evolution operations at a representation level are controlled in order to enforce their conformance with upper (more abstract) representation levels. Conversely, changes are propagated to lower representation levels in order to update them and maintain their consistency with upper representation levels. Our approach allows bottom-up evolution too, in which transitional non-conform architectures can be created to experiment new solutions [16]. Combined top-down and bottom-up evolution prevents architecture erosion and drift [8].

### A. First class change expression

In order to be able to trace the evolution of software architectures, changes should be explicitly described in the same language as for architectures. Furthermore, having them described modularly makes them reusable. This can be summed up saying that **changes need to be first class entities**. Dedal implements this principle. Changes from a version to the next (delta) are described as a list of change operations. Figure 7 gives the example of the *additionStationData* change description.

The effects of changes on architectural elements (component roles, component classes, component instances, architectures, configurations, assemblies) might result in their versioning. Dedal records two pieces of information for each version: the *versionID* that identifies versions and the *pre\_version* link that constitute the structure of the version tree. Figure 5 gives an example of such versioning information for the *BRSConfig* architecture configuration.

### B. Architecture-centric evolution process

The three levels of architecture descriptions constitute the underlying logical foundation of software evolution. Evolution can be initiated at any of the three levels of software architecture. The **evolution process** decomposes into three stages (see Fig. 8). Evolution planning analyzes the impact of change and checks its consistency in each architecture description. Evolution implementation prepares, tests the change and implements it in the implementation environment. Evolution re-engineering propagates changes to other levels and versions software if necessary. This evolution process is controlled by both an architecture evolution management module and an implementation evolution management module.

Consistency is an internal property of an architecture description, which intends to ensure that its elements do not contradict one another [9]. The aim of **consistency checking** is to predict whether changes induce inconsistencies inside and among the three levels of a given architecture. We define *intra-level consistency* that checks name inconsistency, interface inconsistency, attribute inconsistency, interaction inconsistency and *inter-level consistency* that checks mapping inconsistency. If changes preserve consistency, the thought evolution will be permitted. If not, it will either be forbidden or trigger the derivation of a new architecture version for which consistency will be ensured.

If inconsistencies are detected, the evolution process either forbids the thought evolution or consider the thought evolution as being part of a new architecture version. This step is called **change propagation** [23]. The new architecture version will be derived and its content inferred so as to be consistent with

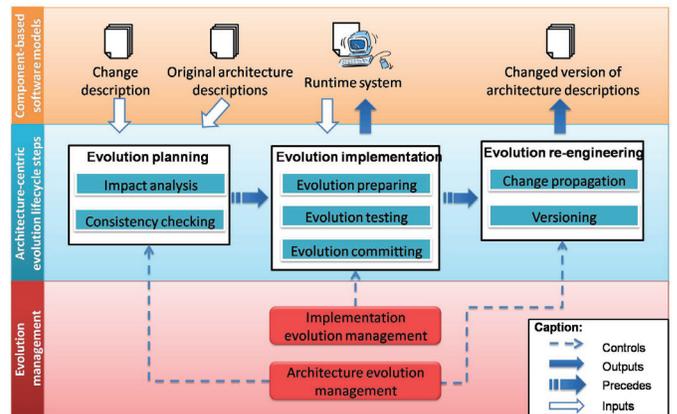


Fig. 8. Architecture evolution process for component-based software

the thought change. The information necessary to derive new versions on each of its levels are extracted from lower to higher levels.

**Versioning** in Dedal relies on a changed-based version model which maintains version trees for the three descriptions of components (roles, classes and instances) and for whole specifications, configurations and assemblies. In each level, versions record three types of information: version ID, previous version ID and change operations. Fig. 9 shows an example of such a version tree for the BRS.

## VI. CONCLUSION

Dedal enables the explicit and separate representations of architecture specifications, configurations and assemblies. Architecture design decisions can thus be precisely captured and traced throughout the component-based development process. Evolution may be initiated at any description level. Consistency is checked and changes are propagated (top-down and bottom-up) to the other description levels so as to maintain the architecture descriptions up-to-date and consistent. This provides a controlled support for component-based software evolution that prevents architecture drift and erosion.

Dedal and the process presented in this paper have been implemented in a tool that validates the feasibility of the approach. It already allowed us to manually experiment evolution scenarios on small examples and gave us preliminary experimental feedback on the expressiveness and adequateness of the proposed ADL. Our implementation is an extension of Julia, an open-source java implementation of the Fractal<sup>2</sup> component model. Dedal-based component architectures can be described and visualized through multiple synchronized views: using a box-based architectural view, using an XML-based syntax for Dedal or using the BNF-based syntax for Dedal presented in this paper. Preliminary experiments have been run on the software architecture of customizable electronic music

<sup>2</sup> <http://www.objectweb.org>

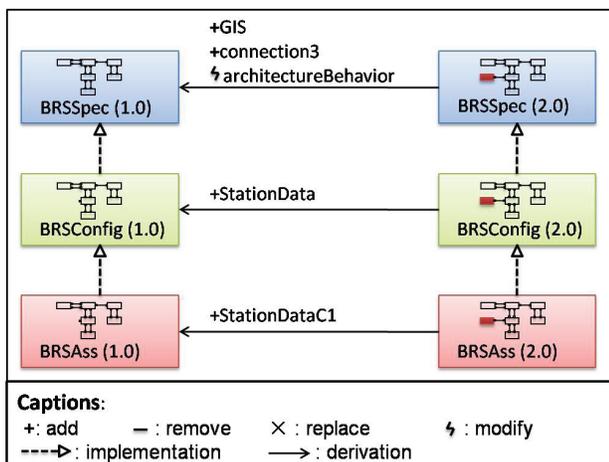


Fig. 9. Version of the BRS architecture

instruments. A perspective for this work is to experiment it to manage component-based software product lines.

## REFERENCES

- [1] I. Sommerville, *Software Engineering*, 8th ed. Addison Wesley, 2006.
- [2] M. R. V. Chaudron and I. Crnkovic, *Software Engineering; Principles and Practice*. John Wiley & Sons, 2008, ch. Component-based Software Engineering, pp. 605–628.
- [3] R. Allen, D. Garlan, and R. Douence, “Specifying dynamism in software architectures,” in *Proc. of the Wkshp on Foundations of Component-Based Software Engineering*, Zurich, Switzerland, September 1997.
- [4] R. Allen, R. Douence, and D. Garlan, “Specifying and analyzing dynamic software architectures,” in *Proc. of the Conf. on Fund. Appr. to Soft. Engineering*, Lisbon, Portugal, March 1998, pp. 21–37.
- [5] N. Medvidovic, “ADLs and dynamic architecture changes,” in *Joint Proc. of the 2nd Int’l software architecture Wkshp and Int’l Wkshp on multiple perspectives in software development on SIGSOFT ’96 Wkshps*, San Francisco, USA, 1996, pp. 24–27.
- [6] N. Medvidovic, D. S. Rosenblum, and R. N. Taylor, “A language and environment for architecture-based software development and evolution,” in *Proc. of the 21st Int’l Conf. on Software Engineering*, Los Angeles, USA, May 1999, pp. 44–53.
- [7] J. Magee and J. Kramer, “Dynamic structure in software architectures,” *SIGSOFT Softw. Eng. Notes*, vol. 21, no. 6, pp. 3–14, 1996.
- [8] D. E. Perry and A. L. Wolf, “Foundations for the study of software architecture,” *ACM SIGSOFT Software Engineering Notes*, vol. 17, no. 4, pp. 40–52, 1992.
- [9] R. N. Taylor, N. Medvidovic, and E. M. Dashofy, *Software Architecture: Foundations, Theory, and Practice*. John Wiley & Sons, January 2009.
- [10] G. Booch, J. Rumbaugh, and I. Jacobson, *Unified Modeling Language User Guide, (2nd Ed.)*. Addison-Wesley, 2005.
- [11] D. Garlan, B. Schmerl, and J. Chang, “Using gauges for architecture-based monitoring and adaptation,” in *Proc. Working Conf. on Complex and Dynamic Systems Architecture*, Brisbane, Australia, December 2001.
- [12] M. Shaw, R. DeLine, D. V. Klein, T. L. Ross, D. M. Young, and G. Zelesnik, “Abstractions for software architecture and tools to support them,” *IEEE TSE*, vol. 21, no. 4, pp. 314–335, 1995.
- [13] T. Bures, P. Hnetyka, and F. Plasil, “Sofa 2.0: Balancing advanced features in a hierarchical component model,” in *Proc. of the 4th Int’l Conf. on Software Engineering Research, Management and Applications*. Seattle, USA: IEEE, 2006, pp. 40–48.
- [14] E. Bruneton, T. Coupaye, M. Leclercq, V. Quéma, and J.-B. Stefani, “The fractal component model and its support in java: Experiences with auto-adaptive and reconfigurable systems,” *Softw. Pract. Exper.*, vol. 36, no. 11-12, pp. 1257–1284, 2006.
- [15] E. M. Dashofy, A. van der Hoek, and R. N. Taylor, “A comprehensive approach for the development of modular software architecture description languages,” *ACM TOSEM*, vol. 14, no. 2, pp. 199–245, 2005.
- [16] H. Y. Zhang, “A multi-dimensional architecture description language for forward and reverse evolution of component-based software,” Ph.D. dissertation, Montpellier II University, France, April 2010.
- [17] M. Shaw and D. Garlan, *Software architecture: perspectives on an emerging discipline*. Prentice-Hall, 1996.
- [18] M. M. Lehman and J. C. Fernandez-Ramil, “Towards a theory of software evolution - and its practical impact,” in *Proc. Int’l Symposium on Principles of Software Evolution*, 2000, pp. 2–11.
- [19] S. S. Yau, J. S. Collofello, and T. M. MacGregor, “Ripple effect analysis of software maintenance,” in *Software engineering metrics I: measures and validations*. McGraw-Hill, Inc., 1993, pp. 71–82.
- [20] K. H. Bennett and V. T. Rajlich, “Software maintenance and evolution: a roadmap,” in *Proc. of the Int’l Conf. on Software Engineering – Future of SE track*, 2000, pp. 73–87.
- [21] N. Medvidovic, “Architecture-based specification-time software evolution,” Ph.D. dissertation, University of California, Irvine, 1999.
- [22] R. Roshandel, A. V. D. Hoek, M. Mikic-Rakic, and N. Medvidovic, “Mae—a system model and environment for managing architectural evolution,” *ACM TOSEM*, vol. 13, no. 2, pp. 240–276, 2004.
- [23] C. Urtado and C. Oussalah, “Complex entity versioning at two granularity levels,” *Information Systems*, vol. 23, no. 2/3, pp. 197–216, 1998.

# Conflict Analysis in Commercial Off-The-Shelf (COTS) Based Development

Hamdy Ibrahim<sup>1</sup>

Tom Wanyama<sup>1</sup>

Armin Eberlein<sup>2</sup>

Behrouz H. Far<sup>1</sup>

<sup>1</sup>*Department of Electrical and Computer Engineering University of Calgary, Canada*

<sup>2</sup>*Department of Computer Science & Engineering, American University of Sharjah, UAE  
{hibrahi, twanyama, far, eberlein}@ucalgary.ca*

## Abstract

*COTS-based development (CBD) in particular the evaluation and selection of COTS candidates is a challenging task. One major challenge is the possible conflict between stakeholders, which can occur during the evaluation and selection of COTS products. This conflict is inevitable since stakeholders have different needs and preferences. In this paper, a conflict analysis method for identifying conflicts during the definition of evaluation criteria and the elicitation of stakeholders' preferences towards these criteria is proposed. Identifying conflicts at these two early stages helps to save or reduce time, effort, and resources required to resolve them at COTS selection decision stage, to understand rational behind these conflicts, and better managing them. A software tool that supports the proposed method is developed. A case study is conducted to demonstrate how the method and its tool are used to identify conflicts.*

**Keywords:** Conflict Analysis, Conflict Resolution, COTS evaluation and selection, Negotiation.

## 1. Introduction

COTS-based development (CBD) is defined as the development of new software systems using one or more Commercial Off-The-Shelf (COTS) products [1]. The use of CBD offers several potential benefits such as reducing cost, development time and effort, while improving the quality of the final software product, and offering functionality to stakeholders which might not have been requested initially but which is still beneficial [2]. However, the evaluation and selection of COTS candidates is a non-trivial task and associated with several challenges, such as the involvement of multiple stakeholders who may have conflicting objectives, needs, preferences, and constraints as they may have different backgrounds, experiences and may come from a variety of disciplines (e.g. users, developers, designers, domain experts and managers) [3,4].

Conflict, in the context of CBD, refers to: (1) possible mismatches between software requirements and COTS features or (2) possible opposition between stakeholders' needs and/or interests. Both types of conflicts may compromise satisfaction of stakeholders and therefore success of the software project [5,6]. Conflict analysis is the first step in the conflict management and resolution process. It helps stakeholders to gain a deeper and better understanding of either the context in which they participate or the dynamics inherent in their relationships. It includes the identification of conflict, its parties, and their rational [7].

In this paper, we propose a conflict analysis method for identifying possible conflicts and their parties at three stages: (1) evaluation criteria definition stage; (2) preference elicitation stage; and (3) COTS ranking stage. A case study is conducted to demonstrate how the proposed method is used to identify conflicts and show its benefits.

The rest of the paper is organized as follows. Section 2 discusses the related work. In section 3, the proposed conflict analysis method is presented. Section 4 discusses how our method is used in a case study while section 5 concludes the paper and discusses possible future work.

## 2. Related Work

There are several COTS evaluation and selection methods which verify the quality and suitability of COTS candidates with respect to software requirements and business needs [8-10]. Some research, such as [5,6,11], address mismatches between software requirements and features provided by COTS candidates. Furthermore, [12] presented a framework for identifying and resolving interoperability mismatches between COTS components forming the software system. Other researchers, such as [13], studied the problem of architectural mismatches during system composition.

The majority of COTS evaluation and selection methods address conflict between stakeholders' needs and preferences in an ad hoc manner [3]. In [3], we proposed a multi-agent model to help stakeholders identify conflicts and evaluate resolution options during the COTS

evaluation and selection process. In this model, stakeholders define their own evaluation criteria (i.e. win conditions), then evaluate and rank COTS candidates based on these criteria. Once the evaluation is completed, the final ranking of COTS candidates is compared to other stakeholders' ranking. If the rankings differ, a conflict arises and a negotiation process is initiated to resolve it. The negotiation process starts with identifying several agreement options, analyzing them considering win-conditions for all stakeholders, and refining stakeholders' win-conditions until an agreement is reached with which all stakeholders are satisfied. The weaknesses of the multi-agent model are as follows: (a) It relies on the ability of stakeholders to define their own evaluation criteria. However, not all stakeholders may have enough experience to perform this successfully and the evaluation criteria definition process requires the cooperation of all key stakeholders. (b) Conflicts are identified at the final stage of the COTS evaluation and selection process (i.e. after completing the evaluation and ranking of COTS candidates). This means that it is late and difficult to figure out the reasons behind these conflicts and the resolution cost in terms of time, effort, and resources usually is high. (c) Conflicts are mainly identified based on a lack of consensus between stakeholders about the final ranking of COTS candidates. This is expected since each stakeholder defines his/her own evaluation criteria considering his/her own needs and interests which may conflict with other stakeholders' needs. This means that there is a need to look for conflicts early while defining the evaluation criteria.

So far there has been very limited research to identify and resolve conflicts associated with the COTS evaluation and selection process.

### 3. Conflict Analysis Method

The method proposed in this research aims to identify conflicts during the early phases of the COTS evaluation and selection process. The method is mainly an extension of our previous multi-agent negotiation model and can be used to overcome the previous model's weaknesses [3]. Figure 1 shows how the method is used during COTS evaluation and selection process which usually starts with defining an initial list of evaluation criteria considering software requirements and features provided by potential COTS candidates. Figure 2 summarizes conflict analysis activities. This paper focuses on evaluation criteria definition (L1) and stakeholders' preferences elicitation (L2) stages activities since L3 stage activities have already been covered in [3]. Some concepts used within the proposed method are based on the work presented in [14,15]. The proposed method works as follows:

Assume that there are:

- a. **N-stakeholders** participating in the analysis process and defined by a set called " $S$ ":

$$S = \{Sh_1, Sh_2, \dots, Sh_n\} \quad (1)$$

- b. **M-evaluation** criteria are defined by a set called " $E$ ":

$$E = \{E_1, E_2, \dots, E_m\} \quad (2)$$

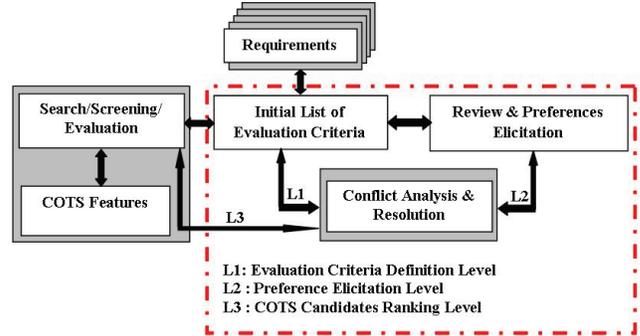


Figure 1: Conflict Analysis Method Within the Evaluation Process

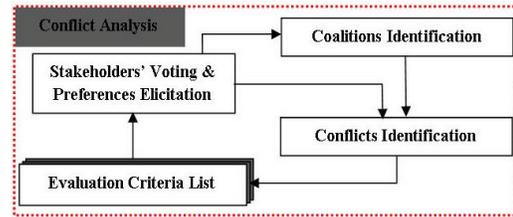


Figure 2: Conflict Analysis Activities at L1 & L2

#### 3.1. Conflict Analysis Activities at L1

1. The objective at L1 is to resolve conflicts arising between stakeholders who have not agreed on a set of evaluation criteria.
2. Once an initial list of evaluation criteria is defined or a new criterion is introduced into the criteria list, each stakeholder is asked to vote on these criteria. Stakeholders can choose between two kind of votes:
  - i. The first voting choice ( $VC_1$ ):

$$VC_1 = \{-1, 0, 1\} \quad (3)$$

Where:

**-1:** stakeholder is against the criterion.

**0:** stakeholder is neutral which means that the stakeholder is neither for nor against the criterion.

**1:** stakeholder is in favour of the criterion.

- ii. The second voting choice ( $VC_2$ ) reflects how much the stakeholder supports the criterion.

$$VC_2 = \{Strongly Agree, Agree, Somehow Agree, Border, Somehow Disagree, Disagree, Strongly Disagree\} \quad (4)$$

3. A voting function ( $V$ ) is defined as follows:

$$V(Sh_i, E_j) = \{(k, l) : k \in VC_1 \ \& \ l \in VC_2\} \quad (5)$$

*Example:*  $V(Sh_1, E_1) = (1, Strongly Agree)$

4. Once the voting process is completed, votes analysis process is invoked to identify whether there are

conflicts and whether there are coalitions between stakeholders. Considering votes belonging to  $VC_i$ , the following relation (R) is defined between two stakeholders ( $Sh_i, Sh_j$ ) on criterion ( $E_k$ ):

$$R(Sh_i, Sh_j, E_k) = \begin{cases} 1 \xrightarrow{\text{if}} V(Sh_i, E_k) * V(Sh_j, E_k) = 1 \\ 0 \xrightarrow{\text{if}} V(Sh_i, E_k) * V(Sh_j, E_k) = 0 \\ -1 \xrightarrow{\text{if}} V(Sh_i, E_k) * V(Sh_j, E_k) = -1 \end{cases} \quad (6)$$

Where "\*" refers to the multiplication or product operator.

5. There are three cases:

a. **Agreement**

When  $R(Sh_i, Sh_j, E_k) = 1$ , stakeholders i and j agree on the criterion  $E_k$  and they form a coalition.

b. **Neutrality**

When  $R(Sh_i, Sh_j, E_k) = 0$ , at least one stakeholder has no opinion or preference (neutral) about the criterion  $E_k$ .

c. **Conflict**

When  $R(Sh_i, Sh_j, E_k) = -1$ , stakeholders i and j have different opinions and disagree on the criterion  $E_k$ . Therefore they are in a conflict about criterion  $E_k$ .

6. Therefore:

If

$$R(Sh_i, Sh_j, E_k) = 1 \ \& \ R(Sh_i, Sh_m, E_k) = -1 \quad (7)$$

Then

$$R(Sh_j, Sh_m, E_k) = -1$$

This means  $Sh_i$  and  $Sh_j$  are in a coalition and a conflict exists between this coalition and  $Sh_m$ .

7. Votes belonging to  $VC_2$  are used to determine the degree of conflict representing how much discrepancy there is between two votes. The degree of conflict determines its priority. If one stakeholder strongly agrees and another strongly disagrees about a criterion, this conflict needs to be resolved before a conflict in which one stakeholder somehow disagrees and another somehow agrees about a criterion.

8. A conflict analysis graph is developed to visualize stakeholders' votes and to determine conflicts. The conflict analysis graph (see figure 3) consists of nodes each of which representing a stakeholder. Solid lines between stakeholder nodes are used to represent a conflict. However, dotted lines are used to represent agreements. Figure 3 shows an example of a conflict analysis graph. There are 6 stakeholders. Three coalitions (i.e.,  $\{(Sh_1, Sh_6), (Sh_2, Sh_5), (Sh_3, Sh_4)\}$ ) can be identified.

9. After identifying conflicts, a negotiation process is initiated to resolve them [3]. Once negotiation is completed, there are two sets. The first set includes the resolved conflicts. The second set includes the non-resolved conflicts associated with the reasons behind negotiation failure. A second attempt to resolve these conflicts will be initiated later at L2 and L3 stages.

### 3.2. Conflict Analysis Activities at L2

1. The objective at L2 is to resolve conflicts arising between stakeholders who cannot agree which criteria are the most important. For example, two stakeholders may agree about the existence of a specific criterion. However, one stakeholder considers it as the most important criterion while the second considers it as the least one in his criteria list. Therefore, if these two stakeholders evaluate COTS candidates without resolving this kind of conflict, they will rank candidates differently (e.g. one stakeholder ranks candidate 1 first, While another stakeholder ranks candidate 2 first).

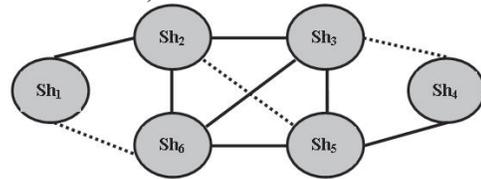


Figure 3: Conflict Analysis Graph

2. Now each stakeholder performs pair-wise comparisons between evaluation criteria to determine weights that reflect their importance for stakeholders.

3. Each stakeholder develops a pair-wise comparison matrix such as the matrix used in the analytic hierarchy process (AHP). The matrix cells represent the relative importance of each evaluation criterion with respect to other criteria. Saaty's scale is used to represent the relative importance values [16]. Then, the matrix is used to estimate the weights and a consistency check is performed to be sure the pair-wise judgments are consistent. If the judgments are inconsistent, the stakeholder will review the pair-wise comparison matrix and modify the values in the matrix to reduce the inconsistency. More details about how weights are estimated and how the consistency check is done can be found in [17].

4. Once stakeholders completed the pair-wise comparisons and weights are calculated, results will be analyzed to identify conflicts. Subsequently, the negotiation process described in [3] is invoked to resolve the identified conflicts.

### 3.3. Conflict Analysis Tool

We developed a tool supporting our proposed method. Figure 4 shows its top level user interface. There are six menu items. The "File" item includes functions for creating, saving, and listing project as well as exiting from the tool. The "Stakeholders" item includes functions for defining who is involved in the specific project. the "Evaluation criteria" item is used to define and modify evaluation criteria. The "Conflict Analysis" item covers the voting process (i.e. L1) and criteria weighting process (i.e. L2). The "Negotiation Process" item includes

functions supporting stakeholders during the negotiation to resolve conflicts. Finally, the "Help" item provides assistance with the usage of the tool. Details about how the tool is used in a case study is covered in section 4.

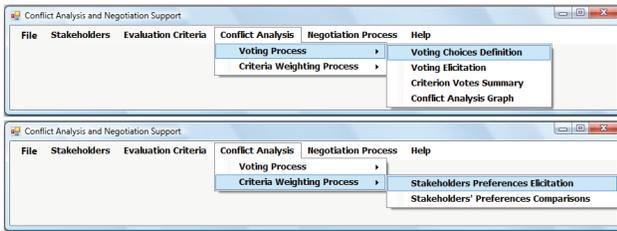


Figure 4: Tool Interface

#### 4. Case Study: Digital Library System

The proposed conflict analysis method and its supporting tool are used during evaluating and selecting a COTS candidate to automate a library system. Three people (two have experience in library science and automation and the third has experience in software development) participated in the case study. Figure 5 shows a list of 24 evaluation criteria used in the case study.

ID	Name
1	Platform Configuration
2	Logging on/off
3	User Account Maintenance
4	Set-up/Installation
5	Vendors
6	User Documentation
7	Licenses Contract
8	Performance
9	Security
10	Membership/borrower functions
11	Catalogue
12	Items Acquisition
13	Multiple-branch Library System
14	Circulation
15	Availability
16	Reliability
17	Maintainability/Modifiability
18	Portability
19	Interoperability
20	Library Capacity
21	Accessibility
22	System generated usage statistics and re.
23	User Interface
24	Queries and Reports

Figure 5 : Evaluation Criteria List

The tool is used as follows:

##### 4.1. Project Setup Phase

During this phase, a detailed information about the project (e.g. name, budget, deadline), stakeholders (e.g. name, contact, domain experience) involved in the evaluation and selection process, and a list of evaluation criteria defined by key stakeholders is recorded. For example, Figure 6 shows the interface used to input stakeholders information.

##### 4.2. Voting Process Phase

This phase includes L1's conflict analysis activities. In this phase, stakeholders vote on defined evaluation criteria. Each stakeholder needs to determine whether s/he is favorable, neutral, or against the criteria as well as

determine how much s/he supports them. Furthermore, s/he should include reasons behind his/her opinion. The interface used to input the voting information is shown in Figure 7. Two methods for analyzing elicited votes are used. The first method results in positive-negative bar charts shown in Figure 8. The positive chart means that a stakeholder favors the criterion and its height represents how much the stakeholder supports it. A negative value means that s/he is against the criterion and its height represents how much s/he is against it. The chart is divided into three intervals corresponding to the three stakeholders.

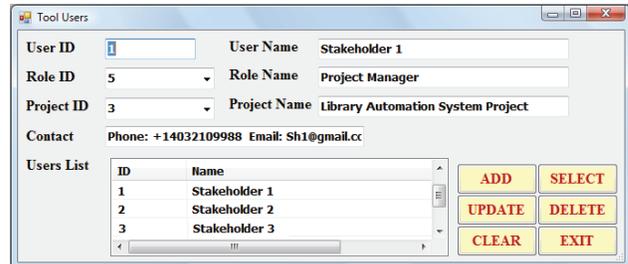


Figure 6 : Stakeholder Information Form

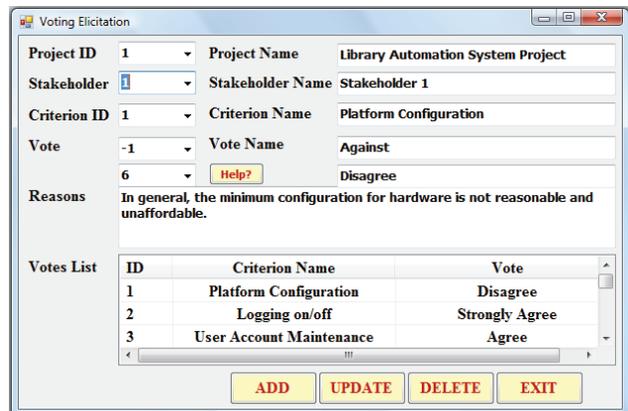


Figure 7 : Voting Information Form

Considering Figure 8, it is clear that for the platform configuration criterion (i.e. first bar charts in each interval) there is a conflict between stakeholder 1 and stakeholders 2 and 3. The same result is confirmed by the conflict analysis graph shown in Figure 9.

##### 4.3. Criteria Weighting Process Phase

This phase includes L2's conflict analysis activities. The module shown in Figure 10 is used by Stakeholders to elicit their preferences of evaluation criteria. Six evaluation criteria have been selected to demonstrate how preferences are elicited. The criteria selection is based on selecting the criteria with which several conflicts are related. Each stakeholder uses the module to perform pair-wise comparisons of the evaluation criteria. The pair-wise comparison values are used to estimate weights for the criteria. The criteria are ranked based on their weights. The criterion that has the highest weight represents the

most important one to stakeholder. If stakeholders assign different weights and ranks to criteria, then conflicts exist.

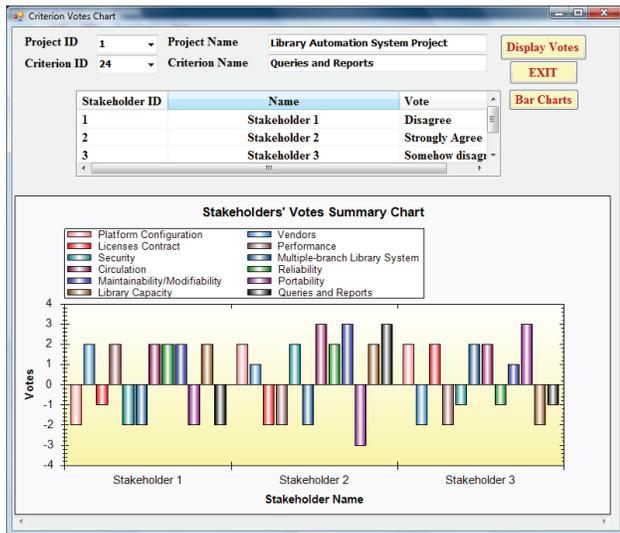


Figure 8 : Stakeholders' Votes Summary at L1

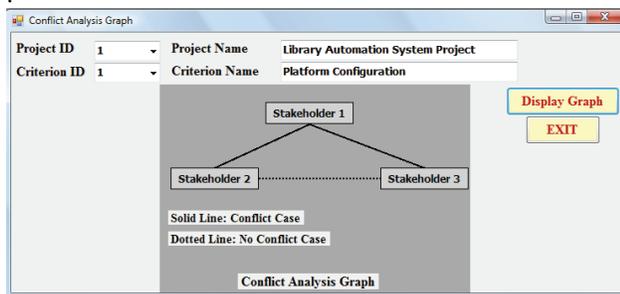


Figure 9 : Conflict Analysis Graph Sample

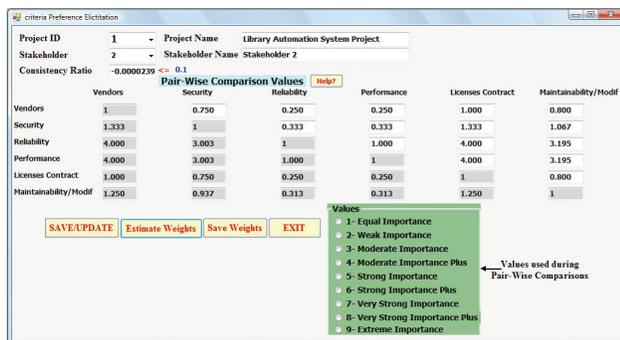


Figure 10 : Pair-Wise Comparisons Module

Figure 11 shows a comparison between criteria weights for the three stakeholders. As shown in this figure, the vendors and license contract are the most important criteria for stakeholder 1 and the reliability and performance criteria are the most important criteria for stakeholder 2. However, security and maintainability are the most important criteria for stakeholder 3. Table 1 summarizes the criteria ranking based on their weights.

It is clear that there is a conflict between stakeholder 1 and stakeholders 2 and 3 with respect to vendors and license criteria. However, there is a conflict between the three stakeholders with respect to the reliability criterion. Other examples for conflicts are highlighted using the same color. There is a need to resolve these conflicts before evaluating COTS candidates. Otherwise, conflicts will arise during the evaluation and selection of COTS candidates.

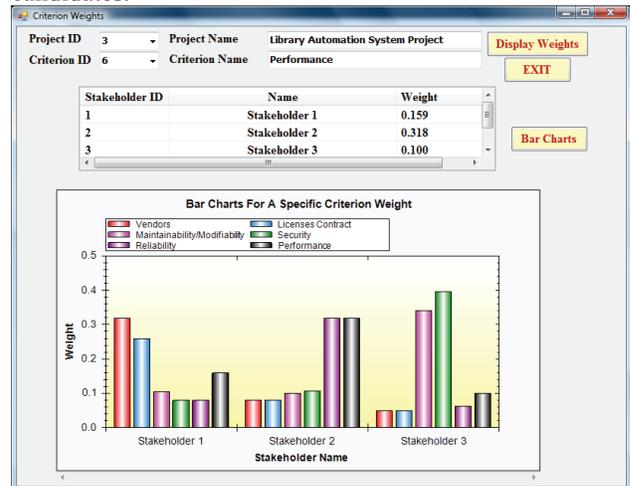


Figure 11 : Criteria Weights Comparisons at L2

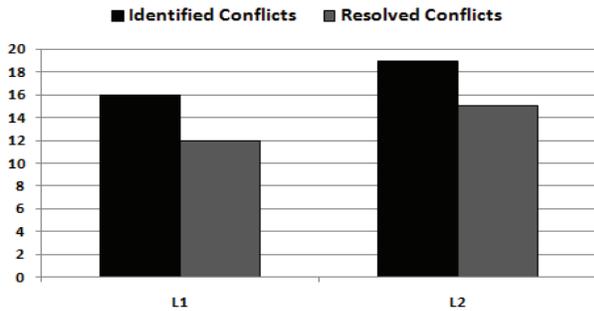
Table 1: Criteria Ranking For 3 Stakeholders

SH/ Rank	1	2	3
1	Vendors	Reliability	Security
2	License	Performance	Maintainability
3	Performance	Security	Performance
4	Maintainability	Maintainability	Reliability
5	Security	Vendors	Vendors
6	Reliability	License	License

#### 4.4. Results and Discussions

Figure 12 summarizes the number of conflicts identified and resolved at L1 and L2 stages. Sixteen and nineteen conflicts have been identified at L1 and L2 respectively. Using the negotiation model in [3], twelve (0.75%) and fifteen (0.79%) conflicts have been resolved at L1 and L2 respectively. At both stages, conflicts are related to both technical (i.e. functional and non-functional) and non-technical (e.g. vendors and license issues) criteria. However the majority of conflicts were related to non-functional criteria such as performance and security. Resolving conflicts related to system functionalities was straightforward and agreement was reached by giving each stakeholder a chance to explain his point of view regarding the criteria and trying to make trade-offs between the criteria. However resolving conflicts related to either non-functional and non-technical criteria was not a trivial task since not all stakeholders completely understood the impact of such

criteria on the system and its operation. Resolving conflicts at L1 helps during the resolution process at L2 even when stakeholders did not reach an agreement and were unable to resolve the identified conflicts because stakeholders had learned more about conflicts, who was involved, what their opinions were, and what reasons were behind not reaching an agreement. This information helped to expect some conflicts identified at L2 and determine their rational in advance.



**Figure 12 : Conflicts Summary at L1 and L2**

## 5. Conclusion and Future Work

In this paper we proposed a method to identify conflicts early at evaluation criteria definition and stakeholders' preferences elicitation stages. The proposed method was used in a case study to demonstrate how both the method and its tool are used to identify conflicts at these two stages. Considering the case study results, we conclude the following:

1. In COTS evaluation and selection process, conflicts between stakeholders are inevitable even when they initially agreed on the software requirements that are used along with COTS features to define evaluation criteria. This is because requirements in COTS-based development are volatile in nature and a continuous change to evaluation criteria list usually happens. This leads to the necessity of getting continuous feedback about the criteria and identifying possible conflicts between stakeholders with respect to these criteria.
2. The proposed method aims at identifying conflicts related to the existence of criteria and criteria ranks.
3. Identifying conflicts at evaluation criteria definition and preferences elicitation stages offers many benefits such as;
  - a. Understanding the criteria and their impact on the system being developed.
  - b. Resolving conflicts early usually needs less time, effort and resources.
  - c. Even when stakeholders cannot resolve a set of conflicts, It is still beneficial since the information about these unresolved conflicts helps in expecting other conflicts, determine their rational in advance, and better management of their consequences.

Our future research will focus on integrating the research presented in this paper with our previous model

[3] to develop a complete method for conflict analysis and resolution which will be supported by a software tool.

## References

- [1] Franch X., Torchiano M., "Towards a Reference Framework for COTS-Based Development: A Proposal", Proc. of the second international workshop on Models and processes for the evaluation of off-the-shelf components, Int. Conf. on Software Engineering, USA, 2005, pp. 1 - 4.
- [2] Cavanaugh B. P. and Polen S. M., "Add Decision Analysis to Your COTS Selection Process", The Journal of Defense Software Engineering, April 2002.
- [3] Tom Wanyama, Behrouz H. Far, "Negotiation Coalitions in Group-Choice Multi-Agent Systems," Proc. of the 5th Int. Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS'06), Hakodate, Japan, pp. 408-410.
- [4] Alves C. and Finkelstein A., "Challenges in COTS decision-making: a goal-driven requirements engineering perspective", SEKE 2002, pp. 789-794.
- [5] Alves C. , Finkelstein A., "Investigating conflicts in COTS decision-making", International Journal of Software Engineering and Knowledge Engineering, Vol. 13, No. 5, 2003, pp. 1-21.
- [6] Alves C., Franch X. , Carvallo J. P., and Finkelstein A., "Using Goals and Quality Models to Support the Matching Analysis During COTS Selection", ICCBSS 2005, pp. 146-156.
- [7] Karn J., "An Ethnographic Study of Conflict in Software Engineering Teams", Journal of Information, Information Technology, and Organizations Volume 3, 2008, pp. 105-133.
- [8] C. Alves and J. Castro, "CRE: A Systematic Method for COTS Selection", Proc. Of XV Brazilian Symposium on Software Engineering, Rio de Janeiro, Brazil, October 2001.
- [9] D. Kunda, "A social-technical approach to selecting software supporting COTS-Based Systems", PhD Thesis, Department of Computer Science, University of York, Oct. 2001.
- [10] Burgues X., Estay C., Franch X., Pastor J. A., and Quer C., "Combined Selection of COTS Components", Proc. Of The First International Conference on COTS-Based Software Systems (ICCBSS'02), Orlando, Florida, 2002, pp. 54-64.
- [11] Mohamed A., Ruhe G., Eberlein A., "MiHOS: an approach to support handling the mismatches between system requirements and COTS products", Journal of Requirements Engineering, Vol. 12, No. 3, July 2007, pp. 127-143.
- [12] Bhuta J. and Boehm B., "A Framework for Identification and Resolution of Interoperability Mismatches in COTS-Based Systems", International Conference on Software Engineering Proc. of the Second International Workshop on Incorporating COTS Software into Software Systems: Tools and Techniques, 2007, pp. 2-7.
- [13] Gacek C., "Detecting Architectural Mismatches During System Composition," PhD dissertation, University of Southern California, 1998.
- [14] Pawlak Z., Skowron A., "Rough Sets and Conflict Analysis", E-Service Intelligence 2007, pp. 35-74.
- [15] Skowron A., Ramanna S., and Peters J. F., "Conflict Analysis and Information Systems: A Rough Set Approach", LNCS Vol. 4062/2006, pp. 233-240.
- [16] Saaty T. L., "Decision Making for Leaders", Third edition, RWS Publications, 2001.
- [17] Ibrahim H., Far B. H., Eberlein A., "Weighted criteria hybrid model for ranking commercial off-the-shelf products", Proc. of the IASTED Int. Conf. on Software Engineering SE 2009, Feb. 17-19, 2009 Innsbruck, Austria, pp.49-56.

# Achieve Semantic-based Precise Component Selection via an Ontology Model Interlinking Application Domain and MVICS

Chengpu Li, Xiaodong Liu and Jessie Kennedy

School of Computing  
Edinburgh Napier University  
Edinburgh, UK  
{c.li, x.liu, j.kennedy} @napier.ac.uk

**Abstract**— With the continuous increases in component varieties and size, it becomes more challenging to find components perfectly applicable for an application. A major obstacle for wider and smoother component reuse is the lack of automated and effective approaches to component specification and retrieval. This paper presents a domain-integrated ontology-based approach to holistic and adaptation-aware component specification and search. The work explores the possibility of combining the MVICS model with domain ontology, with financial domain as a case study. Such a combination enhances the function and application scope of the MVICS model by bringing more domain semantics into component specification and retrieval. The approach and its prototype tool support three distinctive features integrally, including Semantic-based Dynamic Component Retrieval, Flexible Interlink with Domain Model, and Result Component Profile. Furthermore, the inter-linkage between MVICS and domain models has resulted in improved retrieval precision and scope. The effectiveness of the approach and tool has been evidenced by user feedbacks via wide web use.

**Keywords**—Automated component repository and retrieval; ontology-based component specification; domain ontology; linkage between ontologies.

## I. INTRODUCTION

At present Component-Based Development (CBD) fails to reach its full potential due to two reasons: firstly the lack of effective and automated methods for holistically and semantically specifying and retrieving existing components that precisely match user requirements [9]; secondly the specification model is either too domain specific to bear enough applicability or too generic to generate precise enough retrieval result. It is clear that approaches that are able to integrate the generic and domain-specific specification models in an easy and flexible way are highly needed. As we previously identified, the ontology in the existing approaches has too simple and/or monolithic structure and few relationships to deal with the specification and retrieval of modern components [5]. To resolve the above problem, an ontology-based approach is developed to achieve holistic and semantic-based component specification and then automatic and precise component retrieval. As a foundation, a Multiple-Viewed Interrelated Component Specification ontology model

(MVICS) for component specification and repository was developed. Although the MVICS provides an ontology based architecture to specify components in a spectrum of perspectives, it still fails to integrate knowledge of component based software engineering with application domain knowledge, and therefore is immature for real-life use.

In this paper, we focus on exploring the possibility of integrating the MVICS model with domain ontology models because any generic component specification model cannot practically survive without considering application domain. A financial domain related software system ontology model is established as a case study. Such integration enhances the function and application scope of the MVICS model by bringing more domain semantics into component specification and retrieval. The component retrieval method is now not only semantic-based but also dynamic and domain related by introducing the dynamic fiducial class weight for MVICS and the domain model. The result of retrieval includes the result component names, their accurate relevance rating, search information in both sub models of MVICS and the domain model and unsatisfied discrepancy, all are presented in a revised Result Component Profile. Another improved feature of the proposed approach is that the effect of possible component adaptation is also extended to specific domain related components, which enables a more systematic and holistic view in component specification and selection.

A prototype tool is then expanded to accommodate the above improvement on MVICS approach towards domain-integration. The component repository is built to accommodate components from financial domains. Extensive user feedbacks have been received based on case studies and a web-based version of the prototype. The new approach and tool has been evidenced to be more effective for the problem.

The remainder of the paper is organized as follows: Section 2 discusses related work with critical analysis. Section 3 introduces the Multiple-Viewed and Interrelated Component Specification ontology model briefly. Section 4 describes the financial domain related ontology and its linkage with MVICS. Section 5 describes holistic, dynamic and domain-related component retrieval. Section 6 describes the prototype tool and

case study. Section 7 evaluates the approach and prototype based on practical use. Finally, section 8 presents the conclusion and future work.

## II. RELATED WORK

So far, the traditional component specification and retrieval approaches have been suffering from lower recall and precision [7][8][11][12], and are rather limited in accommodating semantics of user queries and domain knowledge. To solve these problems, ontology has been thus introduced to help understand the semantics of components, and domain model is used to capture application domain specific knowledge. In this section, we analyzed some typical work on component retrieval and repository with a focus on ontology-based domain knowledge related approaches, as follows:

When component-based software engineering (CBSE) became a research hot topic, the issue of providing specification of components through ontological models was noticed [4]. However, the research of ontology (information system) therein was not mature, and their ontological models were rather limited.

With the development of ontology technology, many approaches employed the ontology and domain model in the process of the component specification and retrieval. Sugumaran's approach [9] introduced domain ontology to apply the additional knowledge to augment or revise a user initial query. However, the domain ontology used in his approach is less sophisticated, and covers limited semantic information. Liu Quan's approach [6] presented a component description scheme with OWL language, and proposed an ontology based retrieval system architecture. Clearly Liu's scheme is not ontology based and its semantics are not computer-recognizable. In addition, he did not detail the information of domain ontology and the rules to link the ontology to user query. Braga [2][3] addressed the interoperability between component repositories, but only focused on the business components, leaving other kind of components (such as UI component, controller component and IT function component) undiscussed.

To summarize, although ontology-based domain model technologies have improved component specification and retrieval, the existing approaches still have the following limitations: i) ontology has simple and/or monolithic architecture and few relationships; ii) ontology models are either too domain specific to bear enough applicability or too generic to generate precise enough retrieval result; iii) lack of methods to link domain ontologies into generic component ontology.

## III. MULTIPLE-VIEWED INTERRELATED COMPONENT SPECIFICATION ONTOLOGY MODEL (MVICS)

A holistic ontology model of component specification will provide the foundation for effective semantic matching in the component retrieval and improve the precision of component retrieval substantially. The MVICS ontology model has a pyramid architecture, which contains four facets: *function model*, *intrinsic model*, *context model* and *meta-relationship*

*mode*. Each of the four models specifies one perspective of a component and as a whole they construct a complete spectrum of semantic-based component specification. All the four models are extracted from the analysis of CBSE knowledge and have extension slots for specific application domains. The first three models can be viewed as sub-ontology models, each of which describes one facet of component specification. The *meta-relationship model* is used to store four types of inter-relationships among the classes of the first three models. Meanwhile, OWL DL [1] is adopted to define the classes, individuals and relationships of the above models. A detailed description of the MVICS model is presented in [5].

## IV. INTEGRATING DOMAIN KNOWLEDGE OF APPLICATION DOMAINS INTO MVICS

A financial domain related software system ontology was built for the purpose of a case study. This ontology mainly covers the basic financial operation software systems, and was built to collaborate with the MVICS to support the ontology-based financial domain component search. To obtain the semantic meanings between IT functions and financial domain operation, two mechanisms were established to link the financial domain ontology with the MVICS, namely Association Link (AssL) and Aggregation Link (AggL).

### A. Financial Domain Related Ontology

Each class in this ontology represents one software system or module that carries out a financial operation. Superior-subordinate relationships have been used to describe the affiliations of the functions of these systems or modules. The top level classes include Asset Management Systems, Payments & Transfers Systems, Risk Management Systems and so forth. Their subordinates and the subsequent sub-subordinates and so forth constitute their sub classes and sub-sub-classes, till the most specific function units at the bottom level.

### B. Ontology Linkage Method

Different from traditional ontology mediation, the connection between MVICS and financial domain related ontology neither uses the method of ontology merging to create a new ontology, nor uses the method of ontology mapping to make the same or similar ontologies to establish contacts. Our approach enhanced the function and application scope of the component retrieval by linking MVICS to different domain specification ontology with two mechanisms: Association Link (AssL) and Aggregation Link (AggL).

In our approach, for those classes in domain ontology which can be viewed as sub classes of a MVICS class, we call them "Association class". AssL is used to link these financial classes with their super class counterparts in MVICS. The Association classes in financial domain represent specific financial operations, which are viewed as a specialization of their MVICS super classes in financial domain.

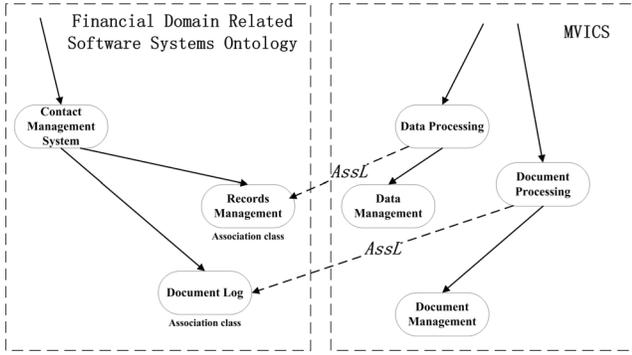


Figure 1. An example of Association

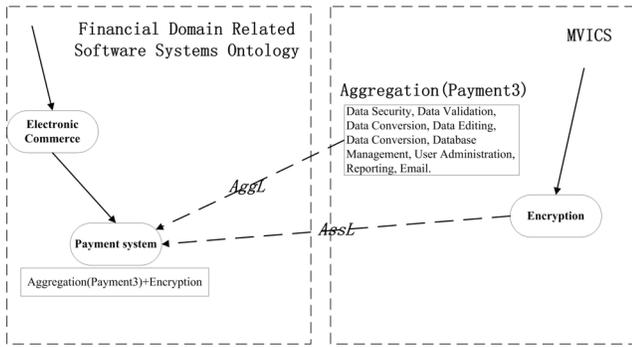


Figure 2. An example of Aggregation

Figure 1 shows an example of AssL. The class *Document Log* in financial domain ontology can be viewed as a sub-class of the class *Document Processing* in MVICS, because logging is a specific of document processing in financial sector, therefore the *Document log* link to *Document Processing* with AssL.

An Aggregation in MVICS is defined as a set of MVICS classes which work together to implement a larger function. Apart from the classes in the financial domain that can be linked in the way of AssL, other financial operation modules are more comprehensive and multifunctional. In this case, we first establish a set of reusable Aggregations in terms of the IT function in MVICS, to represent the function of financial domain operations. These reusable Aggregations are the function units of MVICS with minimum intersection of reusable MVICS functions. Each Aggregation is viewed as a reusable unit oriented to different financial operations. To link a domain model with MVICS, an AggL is defined as a link from a domain class to an Aggregation in MVICS. Classes in the financial domain ontology are linked to MVICS through AssL, where a counterpart super/subclass relation exists, and/or AggL, where a domain class is composed of one or a set of Aggregations in MVICS.

An example of AggL is shown in Figure 2. In total, class *Payment System* in financial domain ontology has the following ten functions which are already defined in MVICS, including *Data Security, Data Validation, Data Conversion, Data Editing, Data Conversion, Database Management, User Administration, Reporting, Email*.

*Administration, Reporting, Email* and *system Encryption*. The first nine functions are composed into an Aggregation (*Payment 3*) in MVICS, and linked with AggL to class *Payment System*. In addition to the above nine functions, the class *Payment System* has an extra function *Encryption*, and this is expressed with an AssL from class *Payment System* to the MVICS class *Encryption*. Thus, the whole function of class *Payment System* is expressed by a combination of the AssL and AggL.

## V. HOLISTIC, DYNAMIC AND DOMAIN-RELATED COMPONENT RETRIEVAL

“Holistic” here refers to that the MVICS is a comprehensive component specification model, and the approach considers a spectrum of respects in component specification and retrieval, such as adaptive component matching and result component profiles. In addition, the financial case study enhanced the function and application scope of the MVICS model and improved the Retrieval Method and Component Profile as follows.

### A. Dynamic Class Weight Calculation

A method for basic class weight calculation is first proposed in [5], however our later findings show that the calculation has to be adjustable at run time because the constant evolution of components in function and QoS. Weight of class ( $W$ ) is defined as the foundation for calculating the precision of result components. In each sub-model of MVICS, every class is given a weight to calculate the relevance of each search result. The weight assignment are formally defined as  $W = (1+X)^n$ , where  $n$  is the level of the layer in which the class locates,  $X = 0.5$  for a class in the function model,  $X = 0.3$  for one in intrinsic model,  $X = 0.2$  for class in the context model [5].

The fiducial weights ( $X$ ) of the classes in each model are given based on our experience. According to the testing data collected from the user, the fiducial weights will be updated dynamically after every 100 groups of user keywords are obtained. Each group of the keywords will be recorded and classified by the facets of the MVICS. The rules of dynamic fiducial class weight assignment are: the more frequently the keywords are used in a facet, the heavier fiducial weight of this facet is. Let  $N$  represent the occurring times of the keywords in a facet, the subscript  $f, i, c$  indicate the related facet (function, intrinsic or context). The weight assignment rules are defined as follows:

$$X_f = 0.5 \times \frac{N_f}{N_f + N_i + N_c} \quad X_i = 0.3 \times \frac{N_i}{N_f + N_i + N_c} \quad X_c = 0.2 \times \frac{N_c}{N_f + N_i + N_c} \quad (1)$$

Moreover, some users may wish only to be influenced by the views of the users in their own user group. Hence, the following rules are proposed to assign fiducial class weight for this purpose, i.e., in support of user group oriented component search, which can further improve the precision of search results. The superscript  $R, E, A$  indicate which user group is

related, namely software engineering researchers, software engineers and amateurs.

$$X_f^{R,E,A} = 0.5 \times \frac{N_f^{R,E,A}}{N_f} \quad X_i^{R,E,A} = 0.3 \times \frac{N_i^{R,E,A}}{N_i} \quad X_c^{R,E,A} = 0.2 \times \frac{N_c^{R,E,A}}{N_c} \quad (2)$$

The weight of a search path ( $W_p$ ) in the MVICS is the sum of the weight of the classes included in it.

The weight of financial domain ontology classes are given on the basis of MVICS class weights. As mentioned in section 4, AssL and AggL were developed to link the financial domain ontology to MVICS. For the class linked with a MVICS class via AssL, its weight is the same as the MVICS class it linked to. For a domain class linked with MVICS through AggL, its weight is the sum of weights of classes contained in the Aggregation. The  $W_p$  in the financial domain ontology is the weight of the matched class.

### B. Domain Related Retrieval Algorithm and Precision Calculation

Based on linkage between MVICS and the financial domain ontology, the component search algorithm was refined. The financial domain keywords are identified as Function Keywords (FK). At the same time, the search tool records the information of the keywords, which will be used to refine the fiducial weights.

The prototype tool will then search the Function Keywords, together with Intrinsic Keywords (IK) and Context Keywords (CK) in the OWL files of MVICS and financial domain ontology. Meanwhile, it will record the search path of every keyword and then calculate the path weight by summing up every class weight in this path for MVICS matched class and by recording the class weight for financial domain ontology matched class. The search tool will record the components that link to the result class. After retrieval, a set of records is obtained for each keyword, which includes the result component name, the search path and its weight. The match precision of a result component ( $P_c$ ) is calculated by the following dynamic fiducial weight, domain related formula, which is refined base on a previous unified formula proposed in [5]:

$$P_c = \frac{\sum_{r=1}^a W_p FK_r}{\sum_{i=1}^i W_p FK_i} \times X_f + \frac{\sum_{r=1}^b W_p IK_r}{\sum_{j=1}^j W_p IK_j} \times X_i + \frac{\sum_{r=1}^d W_p CK_r}{\sum_{k=1}^n W_p CK_k} \times X_c \quad (3)$$

## VI. PROTOTYPE TOOL AND CASE STUDY

A prototype tool has then been expanded to accommodate the above improvement on MVICS approach towards domain-integration. A component repository is built to include components from financial domains. To evaluate both the approach and the prototype tool, a case study based on

financial domain has been conducted with real-life scenarios of component search.

### A. The Prototype Tool

The prototype tool has a simple user interface: A text area for the input of search keywords. The first column of option buttons lists the available domain ontology; the second column spreads out user-oriented options (i.e. Researcher, Engineer and Amateur). And a black panel for showing the summary of search results.

In this tool, the MVICS model, the domain ontology and the related Association classes and Aggregations files are implemented in OWL and located in corresponding OWL files. Figure 3 shows a system overview of the tool. User requirements are refined by the synonym operator first, the corrected and OWL defined keywords are mapping with the classes both in the MVICS ontology and domain related software system ontology. At the same time, the refined keywords are recorded by the requirement recorder. These records are used as primary data to update the fiducial weights of the MVICS classes by the following dynamic class weight assignment method.

On completion of the initial ontology-based searching, if the tool finds some matched classes and their instances (called "result components"), the precision calculator will further calculate the precision of each result component; if there is no matching class, the tool will then search for available adaptive methods or assets in adaptive component matching part. The proposed approach accommodates the impact of adaptation in the specification and selection of matching components. This

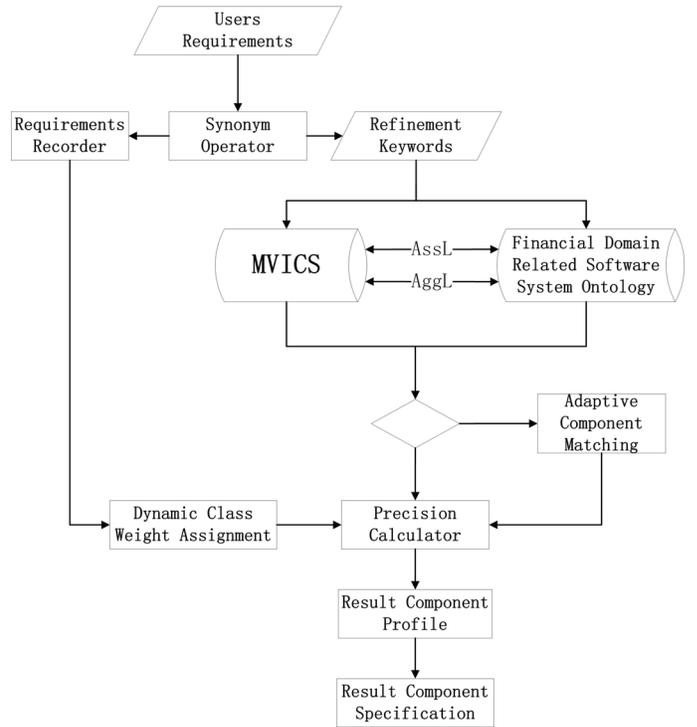


Figure 3. System overview of the prototype tool

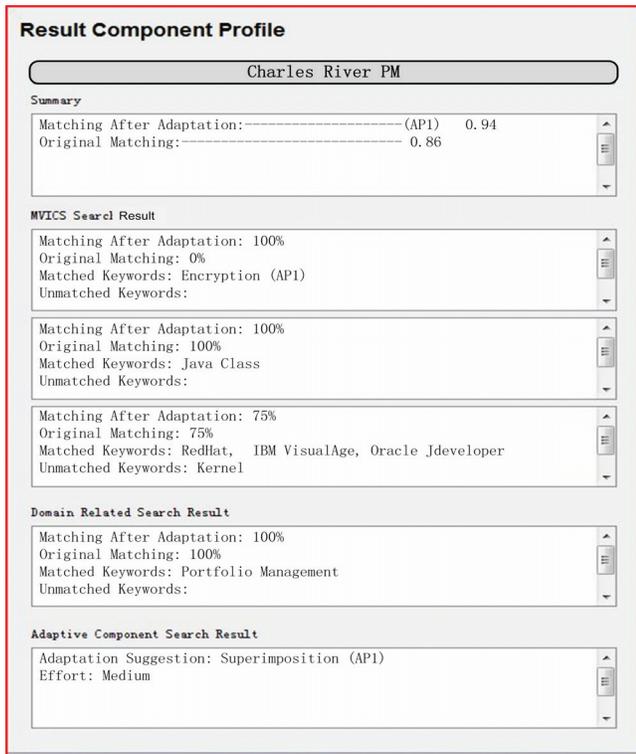


Figure 4. The Result Component Profile

unique feature will allow a more systematic and holistic view in component specification and selection. In MVICS and domain ontology, the adaptive components are linked to a class via an adaptation method or assets if the component becomes relevant to that class after adaptation with that method or asset. These methods or assets are defined as classes or instances in the MVICS and the financial domain ontology. After the adaptive component matching, the search results also go through the following precision calculator to compute the searching precision.

In contrast to most existing approaches, which present only the name and precision of the result component, our prototype tool provides a holistic Result Component Profile to help the user make the best decision in component selection. The revised profile includes the brand new domain related search result for user inspection.

### B. Case Study

A retrieval scenario of developing an encrypted portfolio management system has been studied. We presume a user plans to search for relevant components with the following requirements: *Portfolio management, Encryption, Java Class, RedHat, Kernel, IBM VisualAge and Oracle Jdeveloper*.

Because the portfolio management system is a kind of financial domain related system, the user clicks the financial domain option button, and leaves user-oriented option buttons blank for original precision calculation. The search tool implements the keywords follows the process of prototype tool

as mentioned in section VI. The search engine searched the keywords one by one in function, intrinsic and context sub-model of MVICS, and then financial domain ontology. Afterwards, the search engine search available adaptive methods or assets in adaptive component files.

The names of the result components and their precisions (calculate by the precision calculator) are displayed on the right show panel of the interface. When a result component is highlighted, its search result profile will pop up, as shown in Figure 4. The result component name “Charles River PM” is shown on the top button which can be further clicked to full structured textual specification. Below the component names, it is a search summary, including the original search precision (0.86) and the after adaptation search precision (AP1-Adaptation Path 1, precision is 0.94). The keywords matching information in MVICS model is listed in the middle of the profile. It comprises the keywords matching percentage in each sub model of the MVICS (original or after adaptation), the matched and unmatched keywords. As the shown example, the matching percentages in the three sub models of the MVICS are 100%, 100% and 75% after adaptation; the percentages are 0%, 0% and 75% originally. The keywords “Encryption”, “Java Class”, “RedHat”, “IBM VisualAge”, “Oracle” and “Jdeveloper” are matched in the corresponding sub model respectively. The keyword “Kernel” fails to match in the context model. The following domain related search result display is new in the profile. It shows the search information of the domain related keywords which are collected with help of the linkage between the MVICS and the financial domain ontology. The only domain related keyword “Portfolio Management” is matched in the financial domain ontology model. Therefore, the percentages of original and after adaptation are all 100%, and on unmatched keywords. The available adaptation method(s) or asset(s) with their efforts are specified in the bottom adaptive component search result box.

## VII. EXPERIMENTS AND ANALYSIS

To verify the linkage between MVICS and domain ontology model, the prototype tool was transformed to a web application and published on the site <http://ceres.napier.ac.uk/staff/chengpu/index.asp>. This is a comprehensive project website, including project introduction, component specification, component publish and management, online tool test and questionnaires. Six hundred components were selected from component sale websites, with possible adaptation assets developed, and then were populated into a component repository. To compare with traditional component retrieval approach and other domain ontology-based approach, a SQL database search tool and a domain ontology-based component search tool were built, respectively. The domain ontology-based search tool uses the same financial domain ontology for refining the user requirements and specifying the component without MVICS. Software engineers, researchers and amateurs can use the applications and make comments via a questionnaire. To help the users to compare the function of

## VIII. CONCLUSION AND FUTURE WORK

The objectives of the research are attained by integrating domain knowledge into the MVICS ontology-based approach as an interlinked domain ontology model. This extension solves the component mismatch problem via both holistic, semantic-based and adaptation-aware component specification and retrieval and extending the search precision and scope by the linkage with domain ontology.

The extended MVICS approach supports dynamic and user group oriented retrieval by adjusting the fiducial facet weights. The presented domain linkage method improves the function and application scope of the component retrieval by linking MVICS to different domain specification ontology. With the linkage, the MVICS model is refined and the repository is extended. A financial domain related software system ontology is built as a case study to explore the possibility of integrating the MVICS model with domain ontology.

Our case studies and user feedbacks have shown that the approach and the tool are promising in tackling the drawbacks in component specification and selection. In the future, we plan to improve the MVICS approach by developing a mechanism for the evolution of MVICS model and its linkage with domain models.

## REFERENCES

- [1] Baader, F. (2007) "The Description Logic Handbook: Theory, Implementation, and Applications". *Cambridge University Press*.
- [2] Braga, R., Mattoso, M., and Werner, C. (2001). "The use of mediation and ontology technologies for software component information retrieval", *SSR 2001*, Page(s): 19-28.
- [3] Braga, R., Werner, C., and Mattoso, M. (2006) "Odyssey-Search: A multi-agent system for component information search and retrieval". *Journal of Systems and Software* 79(2): Page(s): 204-215.
- [4] Cmkovic, I., Hnich, B., Jonsson, T. and Kiziltan, Z. (2002). "Specification, implementation, and deployment of components", *ACM 45(10)* Page(s): 35-40.
- [5] Li, C., Liu, X., Kennedy, J. (2010). "A Holistic Semantic Based Adaptation-Aware Approach to Component Specification and Retrieval". In *Springer LNCS, Tunisia: Springer Berlin / Heidelberg*.
- [6] Liu, Q., Jin, X. and Long, Y. (2007). "Research on Ontology-based Representation and Retrieval of Components". *8th ACIS International Conference*, Page(s): 494 - 499.
- [7] Ostertag, E., Hendler, J., Prieto-Diaz, R. and Braum, C. (1992). "Computing Similarity in a Reuse Library System: An AI-based Approach". *ACM Transactions on Software Engineering and Methodology*, Vol. 1(3), Page(s): 205-228.
- [8] Prieto-Diaz, R., Freeman, P. (1987). "Classifying Software for Reuse". *IEEE Software*, Vol. 4(1), Page(s): 6-16.
- [9] Sugumaran, V., Storey, V. (2003). "A Semantic-Based Approach to Component Retrieval", *The Database for Advances in Information Systems*, Vol. 34(3).
- [10] Yao, H., Letha, E. (2004). "Towards A Semantic-based Approach for Software Reusable Component Classification and Retrieval". *ACMSE'04*.
- [11] Zaremski, A.M., Wing, M. (1993). "Signature Matching: A Key to Reuse". *Software Engineering Notes*, Vol. 18, No. 5, Page(s): 182-190.
- [12] Zaremski, A.M., Wing, M. (1995). "Specification Matching of Software Components". *Software Engineering Notes*, Vol. 20, No. 4, Page(s): 6-17.

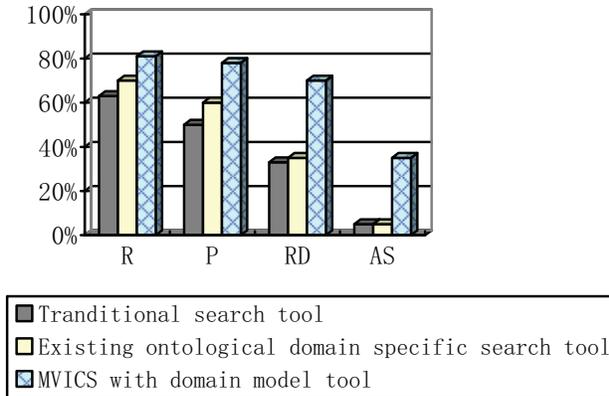


Figure 5. The level of satisfaction of the MVICS with domain model tool, existing ontological domain specific search tools and traditional search tools

different approaches, several financial domain related reuse scenarios with their matching components are provided. Users can test the search results by using given scenarios or producing their own.

The following procedures are suggested for comparison:

- 1) Using the provided scenarios and the corresponding result components ( $R_1$ ), or proposing their own scenarios and realizing the suitable component ( $R_1$ ) manually.
- 2) Using the SQL database search tool, existing ontological domain specific search tool and MVCIS plus domain model tool to search the same requirements and receive a set of search results ( $R_2$ ), ( $R_3$ ), ( $R_4$ ) respectively.
- 3) Comparing  $R_2$ ,  $R_3$  and  $R_4$  with  $R_1$  respectively, and then fill out a questionnaire regarding how well each search was performed according the four criteria: Recall of the financial domain component search (R), Precision of the financial domain component search (P), Result Display (RD), and Adaptation Suggestion (AS).

So far, 127 users with 5 years (average) software development experiences had tested the tool. The results are given in Figure 5. It's clear that there is much improvement in R, P, RD and AS in the domain-related MVICS search tool. The linkage between MVICS and financial domain ontology provides a mechanism to represent both generic (CBSE) and domain specific (financial domain) component knowledge in a hierarchical structure and builds relationships to add more semantic meaning, which helps find the most related components. Furthermore, the precision of result components obtained from the financial ontology classes can be calculated on the basis of MVICS classes through the linkage. This should lead to improved R and P. Compared with the SQL database search and the domain ontology-based component search, the RD and AS are new for component specification and retrieval. The MVICS approaches improve substantially in these two aspects.

# A Top-Down Method for Secure SOA-based B2B Processes

Mostafa Madiesh and Guido Wirtz  
Distributed Systems Group, University of Bamberg  
Feldkirchenstraße 21, 96052 Bamberg, Germany  
{mostafa.madiesh | guido.wirtz}@uni-bamberg.de

## Abstract

*Designing secure B2B processes is a challenging task. The business partners have to agree on a common data format and meaning as well as on the security requirements each partner has to fulfill. In this paper we propose a top-down method to design secure B2B processes in the context of service oriented architectures (SOA). In this method, the Web Services Choreography Description Language (WS-CDL) is used to describe the global behavior of a B2B process, which serves as contractual basis for the collaboration between business partners. The global security requirements are represented in a separate XML-based descriptor document. The local behavior for the individual business partners can be generated from the WS-CDL specification using the abstract Web Services Business Process Execution Language (WS-BPEL). WS-BPEL is used in this step solely to describe externally observable message exchange behavior of each business partner involved, without revealing their internal implementation. These generated abstract WS-BPEL processes are the starting point for implementing new local business processes. The local security requirements for each business partner can be generated from the global security descriptor document. Finally, the implemented WS-BPEL processes are deployed in a suitable WS-BPEL engine.*

**Keywords:** B2B, SOA, WS-CDL, WS-BPEL, BPMN

## 1. Introduction

Nowadays globalized business world puts high demands in terms of flexibility and adaptability on companies and their business processes. Companies are typically required to extend their processes to interact with business partners which renders *Business-to-business (B2B) processes* in which different business partners interact with each other to achieve a common goal, more or less the standard situation. Unfortunately, process planning and organization gets even harder than inside a single enterprise. Often, there are incompatible rules and platforms at each partner, e.g., different software environments and data formats. Global control of the entire B2B process at one single partner is almost always not acceptable; there even may be no detailed knowledge about the process as a whole at a single partner. In fact, each of the

partners has detailed knowledge about her own part but only some insight in those she interacts with. The importance of B2B processes, however, requires a common understanding among the partners about the reliability and security context of the overall processes. This puts an additional burden on the design and implementation of B2B processes.

In this situation, suitable abstraction techniques are essential to determine which information should only be known internally and how much information has to be published among partners to ensure an overall successful collaboration. Hence, each B2B process has (at least) two aspects: a global process view and different local process views. In the global process view, interactions between the business partners and the dependencies between these interactions should be described in a manner that no information about the internal process logic of business partners is included. At this level, security issues which are of global concern have to be, at least declaratively, specified. In the local process view each business partner involved describes the interactions shared with other business partners, and additional internal steps between these interactions, which are invisible to the other business partners. At this level, security issues have to be implemented, too.

As B2B processes cross enterprise borders, standards and well-accepted models and tools for describing and implementing processes are indispensable. For the underlying basic technology, standards for implementing SOAs like WSDL [1] for service descriptions and the Web Services Business Process Execution Language (WS-BPEL) [2] for describing local processes at each partner are a widely accepted choice. For the more abstract and global level there have been a lot of proposals over the last years ranging from a level of high technical detail re-using WS-BPEL over choosing a still technical but more specific language for the global perspective like the Web Services Choreography Description Language (WS-CDL) [3] to using process specifications that are more close to the business domain like, e.g., the Business Process Modeling Language (BPMN) [4] or ebXML BPSS (ebBP) [5]. The former approaches make life hard when discussing business processes with non-IT experts whereas the latter leave a large gap between abstract specifications and suitable process implementations which is especially hard to overcome if security and quality-of-service issues are involved [6].

Over the last years, our distributed systems group has been engaged in a wide range of research trying to tackle these problems using business interaction standards like, e.g., RosettaNet [7], reference architectures [8] and pattern to integrate B2B processes [9] as well as transformation-based approaches using different description levels like ebBP [10] and WS-CDL [11].

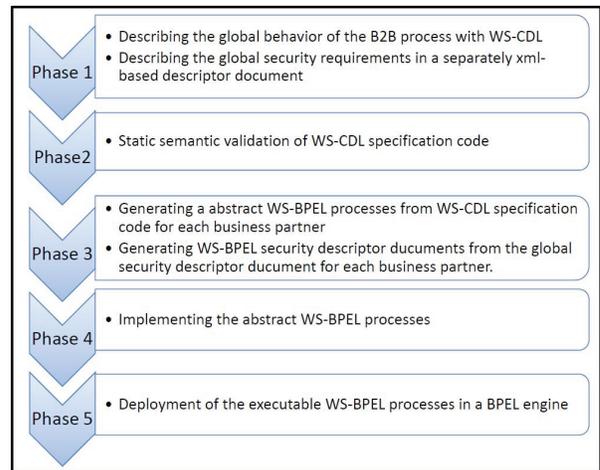
In this paper we propose a refined *top-down method to design secure B2B processes* that has evolved from the basic ideas of [11] with a special focus on respecting security issues during the whole design process. WS-CDL is used to capture the global behavior of a B2B process, which serves as contractual basis for the collaboration between business partners. Because BPMN as a graphical notation is well accepted for designing processes, process descriptions can be made in a BPMN style that is transformed into WS-CDL automatically. The global security requirements are represented during the design process by means of a separate XML-based descriptor document that is also generated from a more human-readable annotation. The externally visible skeleton of the local behavior for the individual business partners is generated from the WS-CDL specification using abstract WS-BPEL. WS-BPEL is used in this step solely to describe the externally observable message exchange behavior of each business partner involved, without revealing their internal implementation. These generated abstract WS-BPEL processes are the starting point for implementing new local business processes at each partner. Most of the local security requirements for each business partner can be generated from the global security descriptor document. Finally, the implemented WS-BPEL processes are deployed in a suitable WS-BPEL engine.

The remainder of this paper is structured as follows. Section 2 discusses some closely related work. Section 3 describes and illustrates the proposed method in more detail. Tool support for the top-down Method is sketched in section 5. Finally, section 6 summarizes the main results and discusses routes for ongoing and future work.

## 2. Related Work

There are lots of approaches to design processes in a service-oriented setting; [12] provides a valuable first overview of a spectrum of methods and techniques used. We concentrate on proposals more closely related to our work in the following.

An informal approach to design inter-organizational workflows based on a workflow setting is presented in [13]. This approach specifies the common public workflow with a so-called 'workflow net', which serves as a contract between the organizations involved. Afterwards, the workflow net is partitioned over the organizations involved and serves as the basis for creating a private workflow at each organization. In [14], a three level approach is presented to map RosettaNet Partner Interface Processes (PIPs) to WS-BPEL processes whereas [15] transforms business transactions into BPEL.



**Fig. 1. Top-Down method.**

[16] also uses a top-down approach to model global behaviors of composite web services using so-called conversation protocols specified by a realizable Büchi automaton enriched with linear temporal logic (LTL) as a basis for formal verification. Finally, the implementation of each business partner involved in the conversation protocol is synthesized from the Büchi automaton via projection.

In [17], starting from a model of service interaction that is created using the global modeling language called *Lets Dance*, an algorithm is defined for determining if a global model is locally enforceable and another algorithm is presented for generating local models from global ones. The generated local models are WS-BPEL processes.

In [18] a framework is presented for securing BPEL compositions using WS-Security and WS-Policy. The main components of this framework are the process container implemented by a set of aspects in AO4BPEL, an aspect oriented extension to BPEL, the security service and the deployment descriptor. In this framework the notion of policy-based process deployment is introduced to check the compatibility of the security policies of the composition and its partners at deployment time.

In [19] an approach is presented to check WS-CDL documents statically. This check deals with those constraints appearing in CDL documents, which cannot be captured by its XML Schema totally. Abstract machines are used to represent the constraints and the corresponding checking algorithms implement a static check of CDL documents.

In contrast to these approaches, our work puts its focus on using SOA standard description languages and to provide integrated tools to generate and/or to check consistent process descriptions which leads to an overall 'compatible behavior' respecting security issues.

## 3. The Top-Down Method

The top-down method (figure 1) comprises of five phases:

**Phase 1:** *Describing the global behavior of the B2B process*

```

<?xml version="1.0" encoding="UTF-8"?>
<security-dd>
<requirements>
<requirement name="req-1" type="usernameToken-signature">
<applyTo>
<interaction name="BuyerRequestsQuote"
path="package/choreography/interaction"
applyToRequest="true"
applyToResponse="true"/>
</applyTo>
</requirement>
<requirement name="req-2" type="usernameToken">
<applyTo>
<interaction name="CheckFails"
path="package/choreography/choice/interaction"
applyToRequest="true"
applyToResponse="false"/>
</applyTo>
</requirement>
</requirements>
</security-dd>

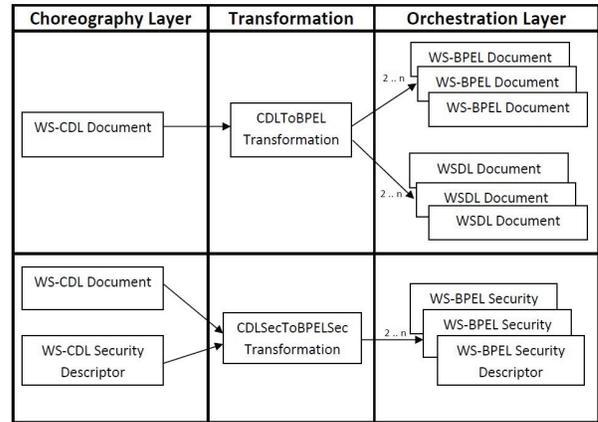
```

**Fig. 2. WS-CDL Security Descriptor**

The business partners involved agree on a common global behavior, which serves as a contract for the overall B2B process, and the security requirements each partner has to fulfill. This common global behavior is provided in WS-CDL. The agreed security requirements are captured in a separate XML-based descriptor document, because WS-CDL does not support describing of security requirements. The security requirements supported in our method are *usernameToken*[20], *signature*[21] and *encrypt*[21]. The *usernameToken* requirement supports the authentication of business partners and the *signature* requirement demands the integrity of exchanged messages between business partners. The *encrypt* requirement is to support the confidentiality of exchanged messages between business partners. Figure 2 shows an example of a security descriptor document. This document has 1 to n requirement elements; each requirement element has *name* and *type* attributes. The *type* attribute may be valued by one of the combinations: *usernameToken*, *signature*[21], *encrypt*[21], *usernameToken-signature*, *usernameToken-encrypt*, *usernameToken-signature-encrypt* and *signature-encrypt*. The requirement element has a sub-element called *applyTo* using a sub-element called *interaction element* specifying to which interaction element of the WS-CDL specification document the security requirement is applied to. This can be done using the *name* and the *path* attributes of the interaction element. The attributes *applyToRequest* and *applyToResponse* are used to specify if the security requirement is applied to the request, to the response or both.

**Phase 2: Static semantic validation of WS-CDL specification**

A static validation of the semantic of the WS-CDL code created in the first phase is performed next. Here, the con-



**Fig. 3. Generating abstract WS-BPEL**

straints mentioned in the WS-CDL specification [3] that can not be captured by WS-CDL XML Schema are determined and validated afterwards.

**Phase 3: Generation of abstract WS-BPEL processes**

After ensuring that the WS-CDL specification is valid, the following transformations are performed (see figure 3):

1. **CDLToBPEL:** The input of this transformation is the WS-CDL document and the output consists of two or more *abstract* WS-BPEL processes and two or more WSDL documents; afterwards there is an abstract WS-BPEL process and its associated WSDL document for each business partner involved in the WS-CDL specification. An abstract WS-BPEL process is non-executable and describes only the public behavior of a business partner which contains the interactions with the other business partners.
2. **CDLSecToBPELSec:** The input of this transformation is the WS-CDL document and its associated security descriptor document created in the first phase; for each business partner the output consists of a security descriptor document for the secure web service provided by the business partner itself and a security descriptor document for each secure web service invoked from a business partner.

The security descriptor documents generated in this phase are *Apache Rampart module* documents. This *Apache Rampart* module [22] is the *Axis2* module that implements the WS-Security functionality based on *Apache WSS4J* [23], an Open Source implementation of the OASIS Web Services Security specification [21]. Using the *Rampart* module, web services can be secured for authentication, integrity and confidentiality, so that exchanged SOAP messages are secured according to specifications in the WS-Security stack, whereas the security requirements are not included in the WSDL document but attached in the header of exchanged soap messages. Figure 4 shows an example for a generated security descriptor document. Each

```

<service>
  <module ref="rampart" />
  <parameter name="InflowSecurity">
    <action>
      <items>UsernameToken</items>
      <passwordCallbackClass>
        password callback class is to be specified
      </passwordCallbackClass>
    </action>
  </parameter>
  <parameter name="OutflowSecurity">
    <action>
      <items>UsernameToken</items>
      <user> username is to be specified </user>
      <passwordCallbackClass>
        password callback class is to be specified.
      </passwordCallbackClass>
      <passwordType>PasswordDigest</passwordType>
    </action>
  </parameter>
</service>

```

**Fig. 4. WS-BPEL Security Descriptor**

security descriptor document generated has two parameter types: the first type is *inflowSecurity* that specifies which security action, i.e., *usernameToken*, *signature* or *encrypt*, have to be applied on receiving a message. The second type is *outflowSecurity* specifying which security action has to be applied if a message is to be sent.

Occasionally, there are elements in the generated security descriptor documents that cannot be initialized in this phase but have to be taken care of in the implementation phase. For instance, in figure 4 there are two elements that cannot be not handled completely in this phase: the *user* element specifying the username of the business partner just as the *passwordCallbackClass* element that specifies which Java class has to be executed in order to retrieve the password of the business partner associated with the username have to be filled in during the implementation phase.

**Phase 4: Implementing the abstract WS-BPEL processes**

Each business partner has to implement his generated abstract WS-BPEL process by adding private behavior and details of execution that are hidden in the abstract WS-BPEL process. In addition the elements of the WS-BPEL security descriptor documents, that are not yet specified, must be specified in this phase. Furthermore, the address element in the generated WSDL document must be specified, too. In this phase, it is crucial for each business partner that the

implementation of the abstract WS-BPEL processes, i.e. the behavior of the refined executable WS-BPEL process, is kept consistent with the behavior of the corresponding abstract WS-BPEL process.

To validate that an executable WS-BPEL process is consistent with an abstract WS-BPEL process, one can adopt the method proposed in [24]. In this method, the abstract as well as the executable WS-BPEL process are transformed into so-called Workflow nets (WF-net)[25] which is a class of the Petri net including special places modeling the sending and receipt of messages. After the transformation, the WF-net of the abstract WS- BPEL process and the executable WS-BPEL are transformed into a communication graph (c-graph) [26]. Then, the c-graph of the executable WS-BPEL process and the c-graph of the abstract WS-BPEL process are compared to verify compatibility rules, like, e.g., that the executable WS-BPEL process accepts at least those messages the abstract WS-BPEL process accepts. If the c-graph of the executable WS-BPEL process simulates the c-graph of the abstract WS-BPEL process, then the executable WS-BPEL process is a consistent implementation the abstract WS-BPEL process.

**Phase 5: Deployment of the executable WS-BPEL processes**

After implementing the abstract WS-BPEL processes and ensuring that all executable WS-BPEL processes are consistent with the abstract WS-BPEL processes, the executable WS-BPEL processes are deployed in a suitable WS-BPEL engine. Then the set of interacting executable processes is executed. We have selected the Apache ODE engine [27] to deploy the executable WS-BPEL processes, since it is open source and it supports the desired security requirements.

**4. Implementation and Tool Support**

The top-down development process for secure B2B processes is supported by a new tool that has been developed in Java. In order to design a secure B2B process using this tool, the user begins with creating a *graphical model* of the common global behavior of the process using the Business Process Modeling Notation [4] (see figure 6). We have selected BPMN as the graphic model of choice, since it is already in wide-spread use for describing choreographies in terms of interconnected interface behavior models and it is an easy understandable graphical notation for business process modeling. The security requirements are specified by annotating the graph when adding a new interaction (see figure 5). Out of the BPMN graphic model a WS-CDL document and its security descriptor document (XML) are automatically generated (see figure 7). Then out of the generated WS-CDL document it's security descriptor document and abstract WS-BPEL process, it's associated WSDL as well as it's security descriptor documents are automatically generated for each business partner involved in the WS-CDL specification (see phase 3).

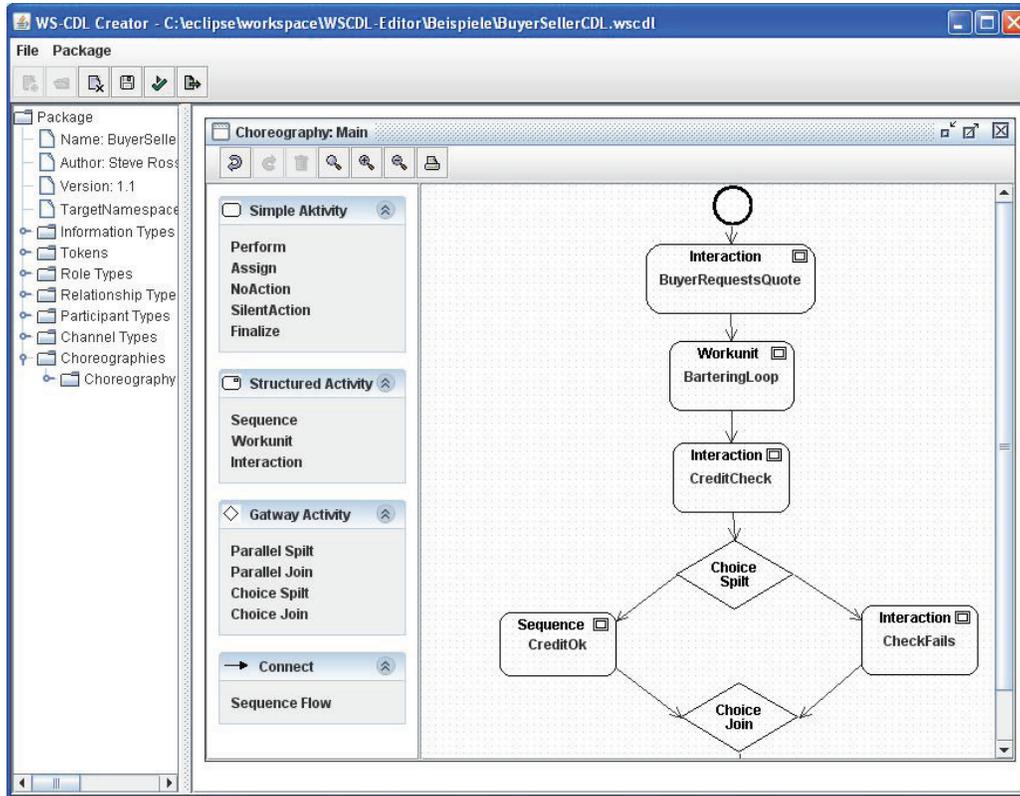


Fig. 6. B2B process design tool

The screenshot shows the WS-CDL Creator interface with two XML documents open. The top document is 'D:\Desktop\BuyerSellerCDL.xml' containing the WS-CDL code. The bottom document is 'D:\Desktop\security-dd.xml' containing the Security Descriptor Document code. Callouts point to the respective code blocks.

```

<choreography name="Main" root="true">
  <relationship type="tns:BuyerSeller"/>
  <relationship type="tns:SellerCreditCheck"/>
  <relationship type="tns:SellerShipper"/>
  <relationship type="tns:ShipperBuyer"/>
  <variableDefinitions>
    <variable name="Seller2ShipperC" channelType="tns:Seller2ShipperChannelType" roleTypes="SellerRoleType"/>
    <variable name="Seller2CreditChkC" channelType="tns:Seller2CreditCheckChannelType" roleTypes="SellerRoleType"/>
    <variable name="DeliveryDetailsC" channelType="tns:2BuyerChannelType" roleTypes="BuyerRoleType SellerRoleType ShipperRoleType"/>
    <variable name="barteringDone" informationType="tns:BooleanType" roleTypes="BuyerRoleType SellerRoleType"/>
    <variable name="Buyer2SellerC" channelType="tns:Buyer2SellerChannelType"/>
  </variableDefinitions>
  <interaction name="BuyerRequestsQuote" channelVariable="tns:Buyer2SellerC" operation="requestForQuote" align="false">
    <participate relationshipType="tns:BuyerSeller" fromRoleTypeRef="tns:BuyerRoleType" toRoleTypeRef="tns:SellerRoleType"/>
    <exchange name="quoteRequest" informationType="tns:RequestForQuoteType" action="request">
      <send/>
    </exchange>
  </interaction>
</choreography>

```

```

<security-dd>
  <requirements>
    <requirement name="req-1" type="usernameToken-signature">
      <applyTo>
        <interaction name="BuyerRequestsQuote" path="package/choreography/interaction" applyToRequest="false" applyToResponse="false"/>
      </applyTo>
    </requirement>
    <requirement name="req-2" type="usernameToken">
      <applyTo>
        <interaction name="CheckFails" path="package/choreography/choice/interaction" applyToRequest="false" applyToResponse="false"/>
      </applyTo>
    </requirement>
    <requirement name="req-3" type="encrypt">
      <applyTo>
        <interaction name="SellerSendChannel2Shipper" path="package/choreography/choice/sequence/interaction" applyToRequest="false" apply
      </applyTo>
    </requirement>
  </requirements>
</security-dd>

```

Fig. 7. Generating the WS-CDL document and its security descriptor document

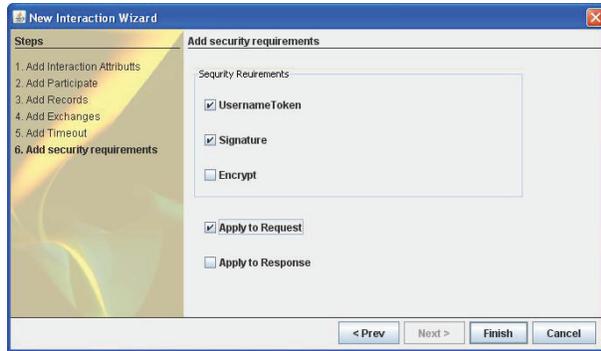


Fig. 5. Specifying the security requirements

## 5. Conclusion and Future Work

We presented a top-down method to design secure B2B processes using SOA standards as implementation technology. The design uses BPMN to describe the global behavior of the B2B process that is transformed into WS-CDL and analyzed afterwards. As WS-CDL does not support the description of security requirements, the global security requirements are annotated to the process graph and represented in a separate XML-based descriptor document. This makes it easy to adapt the same global process to different environments by simply using a more or less strict security descriptor document. Out of these specifications, for each business partner involved, an *abstract WS-BPEL process* as well as an XML-based security requirements document is automatically generated. Afterwards, each business partner implements the detailed local *executable WS-BPEL process* for herself.

Using this method and the tool support for generating most of the coordination code needed, the gap between high-level descriptions of business processes and security requirements and their implementation in a SOA setting is closed at least for top-down style development scenarios.

In order to validate our method we currently conduct different case studies using our tools. First results seem promising and prove that the tool supports the top-down process adequately. However, more real-life size case studies are needed to estimate the overall benefit of the method in an industrial setting. So, ongoing and future work is dedicated to conduct more rigorous case studies for B2B integration in real life settings. Moreover, as processes normally are designed completely new from scratch, combining the top-down method presented here with bottom-up and mixed-style integration scenarios is the next step to go.

## References

- [1] Roberto Chinnici, Jean-Jacques Moreau, Arthur Ryma, and Sanjiva Weerawarana. Web services description language (wsdl) version 2.0 part 1: Core language. W3C, March 2006.
- [2] Alexandre Alves et al. Specification: Business Process Execution Language for Web Services version 2.0. Tech. rep. 2.0, OASIS, January 2007.
- [3] W3C. *Web Services Choreography Description Language*. W3C, 1.0 edition, November 2005.

- [4] Stephen A White. *Business Process Modeling Notation, V1.2*. OMG, January 2009.
- [5] OASIS. *ebXML Business Process Specification Schema Technical Specification*. OASIS, 2.0.4 edition, December 2006.
- [6] Andreas Schönberger and Guido Wirtz. Taxonomy on consistency requirements in the business process integration context. In *Proceedings of 2008 Conf. on Software Engineering and Knowledge Engineering (SEKE'2008)*, Redwood City, California, USA, 1.-3. July 2008. Knowledge Systems Institute.
- [7] Andreas Schönberger and Guido Wirtz. Using Webservice Choreography and Orchestration Perspectives to Model and Evaluate B2B Interactions. In *The 2006 Intern. Conf. on Software Engineering Research and Practices (SERP'06)*. CSREA Press 2006.
- [8] Christoph Pflügler, Andreas Schönberger, and Guido Wirtz. Introducing partner shared states into ebBP to WS-BPEL translations. In *Proc. iiWAS2009, 11th Intern. Conf. on Information Integration and Web-based Applications & Services*, 14.-16. December 2009, Kuala Lumpur (Malaysia). ACM.
- [9] Helge Hofmeister und Guido Wirtz. Applying service-orientation through a reference architecture. *Journal of Systemics, Cybernetics and Informatics*, 6(1):80–90, June 2008.
- [10] Andreas Schönberger, Thomas Benker, Stefan Fritzemeier, Matthias Geiger, Simon Harrer, Tristan Kessner, Johannes Schwalb, and Guido Wirtz. QoS-enabled business-to-business integration using ebBP to WS-BPEL translations. In *Proc. of the IEEE SCC 2009 Intern. Conf. on Services Computing, Bangalore, India*. IEEE, September 2009.
- [11] Mostafa Madiesh and Guido Wirtz. A Top-Down Method for B2B Process Design Using SOA. In *2008 Intern. Conf. on Software Engineering Research & Practice (SERP 2008)*.
- [12] Michael P. Papazoglou and Willem-Jan Van Den Heuvel. Service-oriented design and development methodology. *International Journal of Web Engineering and Technology*, 2(4):412–442, 2006.
- [13] Wil M. P. van der Aalst and Mathias Weske. The P2P Approach to Interorganizational Workflows. In *CAiSE '01: Proceedings of the 13th International Conference on Advanced Information Systems Engineering*, pages 140–156, London, UK, 2001. Springer-Verlag.
- [14] Rania Khalaf. From RosettaNet PIPs to BPEL processes: A three level approach for business protocols. *Data Knowl. Eng.*, 61(1):23–38, 2007.
- [15] Birgit Hofreiter, Christian Huemer, Philipp Liegl, Rainer Schuster, and Marco Zapletal. Deriving executable BPEL from UMM business transactions. In *IEEE SCC*, pages 178–186, 2007.
- [16] Xiang Fu, Tevfik Bultan, and Jianwen Su. A Top-Down Approach to Modeling Global Behaviors of Web Services. In *Proc. of WS on Requirements Engineering and Open Systems (REOS)*, Monterey, CA, September, 2003.
- [17] Johannes Maria Zaha, Marlon Dumas, Arthur H. M. ter Hofstede, Alistair P. Barros, and Gero Decker. Bridging global and local models of service-oriented systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 38(3):302–318, 2008.
- [18] Anis Charfi and Mira Mezini. Using aspects for security engineering of web service compositions. In *ICWS '05: Proceedings of the IEEE International Conference on Web Services*, pages 59–66, Washington, DC, USA, 2005. IEEE Computer Society.
- [19] Lei Zhou, Hanyi Zhang, Tao Wang, Chuchao Yang, Zheng Wang, Meng Sun, and Geguang Pu. Static check of WS-CDL documents. In *SOSE '08: Proc. of the 2008 IEEE Intern. Symposium on Service-Oriented System Engineering*, pages 142–147, 2008.
- [20] OASIS. *UsernameToken Profile 1.1*, February 2006.
- [21] OASIS. *Web Services Security*. OASIS, 1.1 edition, February 2006.
- [22] Apache Software Foundation. Apache Rampart. <http://ws.apache.org/rampart/>, February 2010.
- [23] Apache Software Foundation. Web Services Security for Java. <http://ws.apache.org/wss4j/>, July 2009.
- [24] Axel Martens. Consistency between Executable and Abstract Processes. In *Proceedings of Intl. IEEE Conference on e-Technology, e-Commerce, and e-Services (EEE'05)*, Hong Kong, March 2005. IEEE Computer Society Press.
- [25] W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.
- [26] Axel Martens. On Usability of Web Services. In *Proceedings of Ist Web Services Quality Workshop (WQW 2003)*, Rome, Italy, 2003.
- [27] Apache Software Foundation. Apache ODE. [ode.apache.org/](http://ode.apache.org/).

# Supporting Software Process Improvement in Very Small Entities through a Template-based Guide

Miguel Morales Trujillo,  
Guadalupe E. Ibargüengoitia  
Graduate Science and Engineering  
Computing, UNAM  
Mexico City, Mexico  
{migmor, gig}@ciencias.unam.mx

Francisco J. Pino  
IDIS Research Group – Electronic  
and Telecommunications  
Engineering Faculty.  
University of Cauca  
Popayán, Colombia  
fjpino@unicauca.edu.co

Mario Piattini  
Alarcos Research Group – Institute  
of Information Technologies &  
Systems.  
University of Castilla-La Mancha  
Ciudad Real, Spain  
Mario.Piattini@uclm.es

**Abstract**— When a VSE decides to implement a software process model, the first step is to choose a reference model that is appropriate to its working environment and possibilities, which must then be adjusted to its specific needs. We believe that it would be simpler to adapt the reference model to the needs of the VSE through the use of templates, which would permit the correct integration of the activities proposed in the model and of those that the company has already been carrying out, thus narrowing the gap between the chosen model and its adoption. This paper presents a template-based guide as an alternative through which to implement a process reference model. The templates are developed to capture and concentrate the results of the activities stated in the reference model, thus creating standardized products that meet the appropriate requirements needed to increase the companies' maturity and capability. In accordance with the activities recommended in the COMPETISOFT reference model, we propose templates as an alternative means of generating a product associated with each activity. The templates are made up of individual items - atomic template-units - each of which is supported by the Unified Process and PMBoK methodologies. Our intention is to incorporate the processes into the VSEs, with the support of templates that are manageable, easy and understandable, taking into account the inherent limitations of small businesses.

**Keywords:** *Template-based, VSE, software process model, software process improvement, process reference model, COMPETISOFT, ISO/IEC PDTR 29110*

## I. INTRODUCTION

Software Process Improvement (SPI) is a controlled and guided effort whose objective is to improve process sets in a software development organization, and particularly its capability and maturity. All initiatives to implement an SPI must follow a methodological framework that will provide certain activities leading to the desired improvement.

Very small entities (VSEs), which consist of less than 25 employees, are fundamental to a country's economy. In countries such as the US, Brazil, Canada, China, India, Finland, Ireland, or Hungary, small companies represent up to 85 percent of all software organizations [1] [2].

Nevertheless, if VSEs are to constantly develop and improve, they must adopt software process reference models. However, there is a widely held belief that process reference models are too expensive, require a lot of time and are intended for larger enterprises (more than 100 employees) thus making them difficult for VSEs to acquire [6] [7].

According to [20] other reasons for this phenomenon include: the VSEs remain unaware of these methodologies; in addition, the proposals are difficult to apply owing to the large investment of time, money and resources involved in an SPI. What is more, the proposals are not suitable for the particular characteristics of this type of companies.

In an effort to help VSEs adopt processes in everyday work, process reference models such as COMPETISOFT [3] [4] and ISO/IEC PDTR 29110 [5] were created. These models simplify the required tasks and help to achieve the software engineering practices correctly. However, complex issues still remain and a considerable amount of time is needed to assimilate them.

At the moment of capturing the results of the activities executed, the organization confronts the difficulty of how to represent them in a tangible product. An alternative to the implementation of SPI, in accordance with the process reference models, is a template-based proposal.

The use of a set of templates makes it easier to adopt a reference model [13]. In support of this strategy, this paper describes a template-based guide for SPI implementation in accordance with COMPETISOFT. We propose templates as an alternative means of generating a product associated with each activity recommended by the process reference model.

The use of a template helps to reduce the gap in the knowledge of models, since the template is ready to capture the results of the activities to be executed, and there is no need to resort to any other additional source of knowledge. This signifies that the template acts as an explicit guide and a result container for each activity, always following the required regulations.

This paper is organized as follows: the background is presented in Section II, Section III demonstrates the templates and their characteristics, Section IV describes two case studies and the final section discusses conclusions and future work.

## II. BACKGROUND

Reference models such as COMPETISOFT and ISO/IEC PDTR 29110 are already available for VSEs. They define groups of processes, which later group the activities needed to achieve the desirable improvement. However, these models tend to focus on merely stating the activities and products that are generated by their implementation. We offer a template-based approach which follows the COMPETISOFT framework, in which the template guides the activities and captures their results to generate standard and regulation-oriented products that will boost an organization's maturity and process capability.

COMPETISOFT emerged with the objective of increasing competitiveness among small Latin-American software enterprises, along with providing a common framework that would be easily adjusted to the specific needs of any small enterprise with the purpose of establishing an evaluation and certification mechanism that operates specifically for VSEs.

By seeking to adapt the model to the organization rather than vice versa, COMPETISOFT consistently integrates basic and essential elements in order to ease its implementation. The reference model is divided into three levels: Top Management, Management and Operations, thus reflecting the basic organization division of a VSE. Top Management is consequently in charge of strategic planning and continuous review within the organization; Management is responsible for providing resources, processes and projects, together with supervising the accomplishment of the organization's goals; and finally, Operations is where Project Management, Software Development and Maintenance processes are concentrated.

The intention of the Operations level is to ensure that every software development project that the organization is running is accomplished within the constraints of time, cost and quality [17]. The processes responsible for accomplishing these objectives are: Project Management, the person in charge of finishing the processes on time and within the estimated cost; Software development, which ensures that the product meets the requirements and eliminates defects during early stages of development through verification and validation, and Software Maintenance, which is required after the product is released. Owing to the vital role that the Operations level plays in an organization, this paper focuses on introducing the template-based SPI for these groups of processes.

Having defined the scope of our template-based proposal, we then established an object oriented methodology to control the templates' structure and development. In this case the templates are based on the Unified Process (henceforth referred to as UP) [8] [9] and the Project Management Body of Knowledge (henceforth referred to as PMBoK) [10], since these regulations are applicable to any software development process.

We chose the UP to guide and control the templates since, owing to its similarity with COMPETISOFT, both propose an iterative and incremental development and are highly adaptable to any organization's specific needs, thus

motivating the achievement of high quality standards. Moreover, the fact that UP is accepted as a good reference by the software industry makes it an excellent alternative for the templates' objectives.

As for PMBoK, this is a clearly defined standard for project management, which falls into nine knowledge areas and embraces the best practices, in which the information referring to project administration is standardized in a consistent manner, and that knowledge is then applied to software engineering.

In the following section of the paper, we present the characteristics and development of the templates that are expected to be an explicit guide towards how to make process improvement activities and their fulfillment feasible.

## III. TEMPLATES AND THEIR CHARACTERISTICS

The use of templates as process improvement guides seeks to make a significant contribution to an organization by using them as a tool [18] [19] to reduce the gap between the VSE and the reference model.

The development of the templates began with those that would be used in the Project Management processes, followed by those that would be used in Development and Maintenance. Product generating and non-generating activities were identified for each of the processes. For those activities that have an associated product, the respective templates provide a technique to accomplish them. For example, our proposal is to represent the capture of requirements by means of a Use-Case Diagram; a *requirements workshop* is suggested to create a user interface prototype, and the use of a *dependency graph* is proposed to develop the integration test plan.

In order to bring the activities to a successful conclusion and to achieve their goals, we chose to extract good practices from PMBoK for the Project Management and from the UP for the Software Development and Maintenance processes. A mapping between RUP and PMBoK is presented in [11]; this mapping proves that there are no contradictions or fundamental incompatibilities between the two standards, and that their practices can therefore coexist without abnormalities. It is thus possible to correlate both the PMBoK and the UP guidelines.

The decision to choose UP for Software Development and Maintenance is mostly owing to the fact that those processes are guided by use-cases and they focus on the architecture. In addition, and according to the conclusions reached in the research conducted in [12], the scope of coverage between those two processes is largely attained in accordance with the ISO/IEC 15504 qualification [15]. The ISO/IEC 15504 regulation provides a model for process evaluation and the coverage scope is determined by establishing the minimum requirements that the processes must fulfill.

However, we discovered that some of the products suggested by COMPETISOFT are not taken into account by the UP, for instance, Integration Test Plan and Report. In COMPETISOFT an entire development phase is designated for this activity. For that reason, in some cases we opted to

follow industry examples that were adapted to the objectives of this project.

The main feature of the templates is the modularity of the entities of which they are composed. These entities can be divided into two types: general entities and process entities. The general entities are common to all the templates and focus on describing the following aspects: Introduction, Overall description, Rationale for decisions and Supporting information. The process entities will capture all the information generated by the activities described in the reference model.

These entities will be referred to as *atomic template units*, and each *atomic unit* contains a technique with which to generate its content. Altogether, they represent a detailed picture of the step-by-step creation of a product that would meet all the requirements and would fit in suitably with the rest of the products. The division of templates into atomic units facilitates traceability of the software elements during their life-cycle. An example of this could be the trace connecting client's requirements, use-cases and classes in order to implement them, where each component can be easily traced and associated at any point of development.

Every process component is chosen in accordance with COMPETISOFT requirements, always ensuring that the template is simple and self-contained. The Testing template can serve as an example, in which the atomic unit Testing Strategy is responsible for the tests that are not considered by the methodology, but are required by the organization. Each template is assisted by an *Atomic Unit Diagram* which serves as a guide for its correct use.

To summarize, one of the key features of the templates is the modularity of their process entities which allows them to be easily adapted to different reference models.

#### A. Project Management Templates

The first process subgroup concerns the Project Management (PM), which is in charge of the establishment and systematic fulfillment of the activities needed to achieve the time and cost goals of a project. The Project Management activities produced four templates; three of these are products that are repeatedly generated during the process: *Traceability Matrix*, *Acceptance Document* and *Monitoring Report*. These documents were therefore standardized and included those specific elements from each activity that are considered to be good practices by PMBoK. The fourth template, Project Plan, describes the information related to the project, which includes and records PMBoK techniques.

It is worth pointing out that we attempted to make the filling out of the Project Management templates as intuitive and natural as possible, since these templates are used several times throughout the process.

Figure 1 shows the atomic unit Version Control Strategy which, in accordance with PMBoK techniques, defines the way in which the changes will be managed during the development, maintaining the baselines' integrity. Moreover, another part of the Project Plan template is the Change Request Form, which is essential for keeping a record of project changes.



### 16. Software configuration management

[Method for managing software configuration; must include a mechanism to request changes as well as the structure and rules to use the repository.]

Version control strategy											
Software configuration identifier	[establish what will be stored in the repository; provide a baseline for classifying products and documents]										
Software configuration condition	[provide information about the products in the repository; listing the products stored, their location and condition]										
Auditor	[a person responsible for verifying the existence and condition of the products previously established to generate. The auditor should collaborate in each audit that the execution of the products has been fulfilled successfully]										
Frequency of audits	[establish within which period the software configuration will be audited]										
Requested changes	<table border="1"> <thead> <tr> <th>Product name</th> <th>Description</th> <th>Impact</th> <th>Petitioner</th> <th>Condition</th> </tr> </thead> <tbody> <tr> <td colspan="5">[list of the changes requested along the process]</td> </tr> </tbody> </table>	Product name	Description	Impact	Petitioner	Condition	[list of the changes requested along the process]				
	Product name	Description	Impact	Petitioner	Condition						
[list of the changes requested along the process]											
Policies	[agreements on the use of the repository; lay down who can store, modify and access the repository]										

Change Request Form	
Project:	Date:
<b>Product information</b>	
Name:	
Backup location:	
<b>Change information</b>	
<b>Change description</b>	
<b>Change benefit</b>	
<b>Change impact</b>	
<b>Change condition</b>	
Resolution date:	
APPROVED <input type="checkbox"/> REJECTED <input type="checkbox"/>	
Name and signature of the resolver:	

Figure 1. PM Project Plan template fragment.

#### B. Software Development Templates

The products required by Software Development (SWD) processes are incorporated in one template for each of the phases proposed by COMPETISOFT. This produces seven elements, of which the *Requirement Specification* comes first. This template includes the atomic unit specially designed for modeling the requirements by means of use-case technique, this atomic unit being the most important part of the document. The life-cycle traceability of each element begins with the registration of every requirement. The System Testing Plan and the User Interface Prototype should be included in the same document to cover every use-case.

The *Analysis, Design, Construction, Integration, Test and Closure* templates were also designed.

As an example of these templates in Figure 2, we can observe the elements that make up the Logical View (point 4), where a description of the view (4.1), along with the definition of the architecture (4.2), is provided by using packages. The Deployment View (6) is where a deployment diagram (6.2) needs to be provided for its presentation. Finally, the Data View (7.) includes the instructions necessary to define aspects concerning data persistence (7.1.) and represent them in a diagram (7.1.1.). All of the above is based on and adapted from the UP Design Phase techniques.

It is necessary to mention that the atomic units could be reorganized into subsets to meet the needs of other reference models.



4. Logical View

4.1 Overview

[This section describes in general the decomposition of the model by means of package hierarchy.]

4.2 Packages of architecture

[Every package should include a name, a brief description and a diagram with the most important classes it contains.]

4.2.1 <Package Name 1>

4.2.1.1 Brief Description of the Package

[The description should state briefly the main purpose of the package.]

4.3 Design Classes

[For each class contained in the package, there should be included a detailed class, a description of the responsibilities of the class as well as its attributes and operations.]

4.3.1 <Detailed class diagram>

5. Process View

5.1 Overview

[This section describes the system decomposition into processes and process groups, organized according to the interactions between them. It describes the principal means of communication used in these interactions.]

6. Deployment View

6.1 Overview

[This section should describe the software network configuration for its execution. It should indicate the nodes (computers, routers, etc.) necessary for the proper functioning of the software, and interconnections between nodes (bus, LAN, etc.) Ideally it should include a mechanism or protocol of node communication.]

6.2 Deployment model

[Distribution of the identified nodes.]

7. Data View

7.1 Overview

[This section should include a description of the data storage mechanism of the system. This section is optional in case if there is little or no persistent data or the translation between the design model and the data model is trivial.]

7.1.1 <Database Diagram>

Figure 2. SWD Design template fragment.

C. Software Maintenance Templates

The Software Maintenance (SWM) process established by COMPETISOFT is an agile process [16]. In order to carry it out successfully, we abstracted two necessary elements: the *Modification Request* template, from which the maintenance team participation is requested, and the *Intervention* template in which a detailed description necessary to perform the requested modification is provided.

These templates have the same structure as the Project Management templates, and are meant to be user-friendly, owing to the fact that the *Modification Request* could be filled out by a system user and not by a software developer, as is shown in Figure 3. In this figure, we can observe that each entity of the template appears to be easy to abstract and that the information obtained would be useful for the maintenance team. Each atomic unit represents an explicit guide consisting of direct questions for the template user, who must give an immediate, clear and sufficiently complete answer.



Modification Request

<b>Priority</b>	[Normal / Urgent]
<b>Error/Change description</b>	[This section should answer questions like these: Who experienced the error? When did the error occur? How did the error occur? These questions must be answered by the system user who gets the error occurred. The purpose of this section is to determine actions that triggered the error.]
<b>Occurrence environment</b>	[This section should include a list of the features presented in the system environment at the time of the error submission.]
<b>Affected aspects</b>	[This section should list the issues that are affected by the described defect or the occurrence of the error.]
<b>Requested by</b>	[Name of the person making the request, in case of being other than the user who experienced the error include the name of the user.]
<b>Diagnosis and potential solutions</b>	[In this section possible solutions to the error should be proposed as well as error diagnosis, ie establishing the nature of the error by observing the signals causing it.]

Figure 3. SWM Modification Request template.

D. Atomic Unit Diagrams

In order to provide a graphic representation of the implementation of the templates during the process, diagrams were generated, in which three important elements were conjoined: roles involved, list of activities described in the process reference model and the template's atomic units in-use. Figure 4 presents the atomic unit diagram related to the Project Plan template. On the left-hand side of the figure there is a graph showing the activities described in the Project Management process taken from the COMPETISOFT reference model. (Sequence A. 1.)

In this diagram, the roles (columns) are related to each activity by means of swim lanes. The template's atomic unit associated to a corresponding activity of the process is positioned in each lane, therefore resulting in a clear and consistent mapping between the process reference model and the templates. The objective of the aforementioned diagram is that of meeting specific needs of the activities.

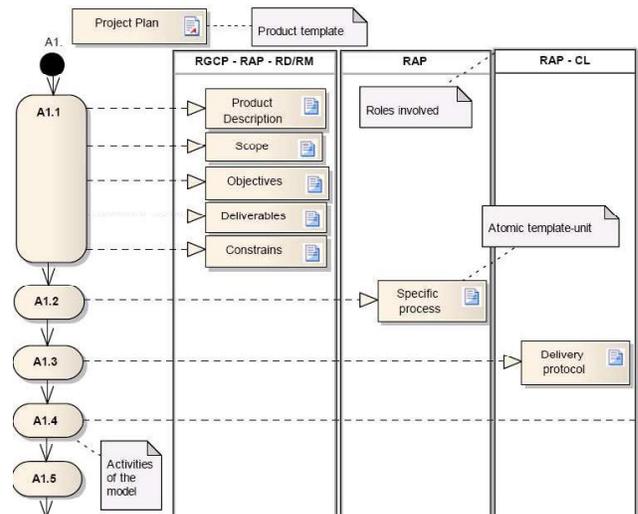


Figure 4. PM Project Plan Atomic Unit Diagram fragment

### E. Template Adjustment to ISO/IEC 29110-5-1

Taking advantage of the modular structure of the templates, a set of atomic units was chosen to be adjusted to ISO/IEC 29110-5-1, with the objective of designing templates that would be part of each deployment package [14] proposed by this future regulation.

Figure 5 shows the content that resulted from gathering together suitable atomic units in order to complete the Software Requirements Analysis deployment package from ISO/IEC 29110-5-1. This quick and easy adjustment to the deployment package is an example of the benefits of choosing to structure the templates from atomic units. The modular aspect of templates is meant to make process implementation more flexible and user-friendly.

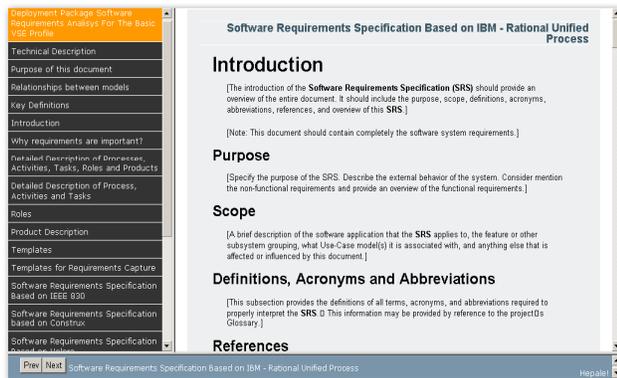


Figure 5. Content of Software Requirements Analysis deployment package template.

## IV. CASES STUDY

The templates designed for the Project Management and Software Development processes were used in the development of three software products related to an academic environment. The first case is the application of templates to a tool development project called HEPALE! (Educational Tool for the Adoption of Standards from its acronym in Spanish) [21], which consists of two modules, both of which are in charge of integrating distance learning standards to spread COMPETISOFT and ISO/IEC 29110 models. Moreover, this tool obtains useful statistics which can be analyzed to provide means to improve educational offerings and to contribute to the dissemination of reference models at an international level. The other use of templates was conducted during the creation of a repository, which contains all the changes needed to manipulate learning objects and e-learning principles in its structure. The architecture of the repository is service-oriented and implements web services to manipulate, search and store deployment packages. The repository not only makes the interchange of contents among different Learning Administration Systems easier, but also facilitates their publication.

The HEMOSIST case study was developed outside the academic environment and with a real client. This system was designed for the National Institute of Cardiology to

manage the database records of Hemodynamic patients. In order for the system to be easily upgraded, there was an essential need for proper documentation and for a software process reference model that could be followed.

The development team was composed of 5 people who had never followed a model and who adopted the template-based proposal to guide their development according to COMPETISOFT and to meet the quality requirements. The team confirmed that the application of templates assisted in following the model, reducing the effort and time to adopt it, which permitted them to focus on the development of the system to a greater extent. At the end of the development, the system was delivered on time and with the quality standard required, whilst the organization obtained a full system documentation, leaving it ready for an upgrade or maintenance process. At present the system is successfully in use at the Institute.

As an example of the use of the templates, the document resulting from the implementation of the Analysis template during the development of HEMOSIST is shown as follows. Figure 6 demonstrates the outcome of applying the techniques proposed for the Architecture Representation atomic unit, and a diagram, goals and system architecture restrictions are presented.

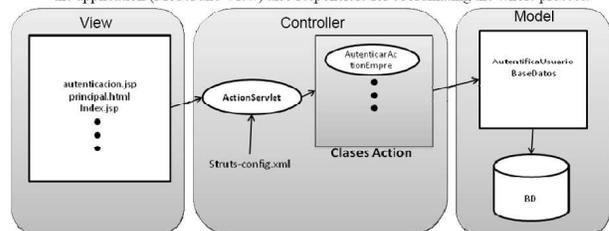
Author: Jakost  
Version: 1.8  
Date: 12/11/2009



### 2 Architecture representation

Due to the fact that this is a web application, we choose the Model View Controller (MVC architecture) since it is the best option for this type of applications. The following is the introduction of each of the layers:

- **The View:** It is in charge of generating responses to be sent to the client. Its implementation will be carried by a .jsp page that will display a dynamic content.
- **The Model:** It is where the business logic of the application comes, including data access and manipulation. The model consists of a set of business components enabling their reuse and decoupling.
- **The Controller:** All requests made are directed to the controller, whose task is to determine what action to take for each of the requests, and invoke other components of the application (Model and View) also responsible for coordinating the whole process.



Struts is a framework that implements the abovementioned architecture. This framework demands the creation of *Action* type classes which are responsible for managing different types of requests that arrive from the client. For example, the class *Authentication.Action* calls the model component in charge to handle the user authentication.

On the Model side the classes that handle the connection to the database and business logic are presented, this will require the use of DAO and VO patterns. Type VO classes are representations of the database tables and DAO classes are those that permit transactions with the database.

#### 2.1 Architecture overview

For the development, Netbeans IDE version 6.7.1 will be used, which provides a complement to UML diagramming and several web application servers, from which GlassFish version 2.1. has been chosen.

For the coordination of development team and easy integration testing the Subversion version manager will be utilized. A server has been installed in the computer containing the repository.

Figure 6. Example of use of the Architecture Representation atomic unit.

## V. CONCLUSIONS AND FUTURE WORK

Approximately 80% of software development organizations are categorized as VSEs, that is, they have 25 or less employees. This overwhelming majority leads to the necessity for a reference model which can be adapted to VSEs rather than vice versa. Bearing in mind the aforementioned aspects, a template-based guide was proposed to assist in the adoption of reference models with the aim of ensuring process improvement without forgetting the organizations' real working environment and real needs. In this paper we have presented the template-based guide designed to introduce and follow COMPETISOFT. Among the guide's characteristics are flexibility and ease-of-adjustment to the organization's real needs provided by the modular structure of templates. Moreover, these templates can be adapted and used with other reference models, such as ISO/IEC CD TR 29110-5-1, thanks to the use of process entities in atomic units.

Some of the positive comments and aspects mentioned by the people who have used the templates are: (i) The templates facilitate and show the best course to follow when carrying out process activities according to a reference model; (ii) The templates include all the aspects that are necessary and are required by a reference model, and it is therefore difficult to overlook any of them; (iii) The templates serve as fill-in forms and incorporate all the resulting information, which allows a complete, organized and standardized documentation to be obtained at the end of a process; (iv) The use of atomic units ensures the traceability of elements and gives the template remarkable flexibility and usefulness since the developer is free to decide whether or not to use each atomic unit; and (v) The way in which the atomic units are constructed, and particularly those of Project Management and Software Maintenance, reduces the time and effort needed to fill out forms, making it simple and easy for any user.

As future work we shall consider the following: (i) To integrate the template-based proposal into all previous work done with regard to COMPETISOFT, thus making the templates accessible and available for use to any software organization; (ii) To obtain a larger number of applications of the templates in order to evaluate, improve and refine their content; and (iii) To continue a widespread industrial implementation in real software development organizations.

## ACKNOWLEDGMENTS

This work has been funded by the following projects: ARMONÍAS (JCCM of Spain, PII2I09-0223-7948), INGENIO (JCCM of Spain, PAC08-0154-9262), PEGASO/MAGO (MICINN and FEDER of Spain, TIN2009-13718-C02-01), PAEP and the mixed-grant scholarship program of CONACYT.

## REFERENCES

- [1] Qualidade no Setor de Software Brasileiro-Pesquisa 2001, Ministry of Science and Technology (MTC), 2001; <http://ctjovem.mct.gov.br/index.php/content/view/34854.htm> 20/02/2010.
- [2] Software Industry Statics for 1991-2005, Enterprise Ireland, 2006; [www.nsd.ie/hm/ssii/stat.htm](http://www.nsd.ie/hm/ssii/stat.htm), 15/01/2009.
- [3] Oktaba, H.; García, F., Piattini, M., Pino, F., Alquicira, C., Ruiz, F. "Software Process Improvement: The COMPETISOFT Project". IEEE Computer, October, 2007. Vol. 40(10), pp. 21-28.
- [4] COMPETISOFT, <http://alarcos.inf-cr.uclm.es/Competisoft/>, 20/02/2010.
- [5] ISO/IEC 29110 Draft version, BER-005 29110 TR VSEP Overview WD2 15CL.
- [6] Richardson Ita, Gresse von Wangenheim Christiane, "Why Are Small Software Organizations Different?", IEEE Software, January/February 2007. Vol. (40). pp.18-20.
- [7] Laporte, C. Y., Alexandre, S., O'Connor, R., "A software engineering lifecycle standard for very small enterprises", in R. V. O'Connor et al. (Eds.): EuroSPI 2008, CCIS 16, pp. 129-141
- [8] Kruchten P., The Rational Unified Process. An Introduction. 2nd edition. Addison Wesley. 2000."
- [9] Jacobson I., Booch G., Rumbaugh J., The Unified Software Development Process, Pearson, 2000
- [10] A Guide to the Project Management Body of Knowledge, Third Edition, Project Management Institute, USA, 2004
- [11] Charbonneau, S. "A mapping between RUP and the PMBok", <http://www.ibm.com/developerworks/rational/library/4721.html#author,01/05/2009>
- [12] Cánepa K., Dávila, A., "Mapeo de los Procesos de RUP respecto a MoProSoft", JIISIC 08, pp. 139-146
- [13] Kong, B., Luo, X., Jiang, Y., Mao, M. "Template-based software process improvement", Journal of Information and Computational Science 1 (2), pp. 175-180
- [14] Laporte, C., Alexandre, S., Renault, A., "Developing International Standards for Very Small Entities", IEEE Computer 41(3): 98-101 (2008).
- [15] van Loon, H. (2007a) "Process Assessment and ISO 15504", Springer.
- [16] Pino, F., Triñanes, G., García, F. y Piattini, M. "Agil MANTEMA: Una metodología de mantenimiento de software para pequeñas organizaciones". JISBD 08, pp.171-182
- [17] Oktaba, H., Piattini, M., Pino, F., Orozco, M. Julia., Alquicira, C., COMPETISOFT: Mejora de Procesos de Software para Pequeñas y Medianas Empresas y Proyectos, Ra-Ma, 2008
- [18] Word Template Index, <http://homepages.fh-giessen.de/~hg14540/SQM/projekt/rup/wordtmpl/index.htm>
- [19] Rational Unified Process: Artifact Overview [http://www.ts.mah.se/RUP/RationalUnifiedProcess/process/artifact/ovu\\_arts.htm](http://www.ts.mah.se/RUP/RationalUnifiedProcess/process/artifact/ovu_arts.htm)
- [20] Pino, F., Hurtado, J., Vidal, J., García, F., Piattini, M. "A process driving Software Process Improvement in small Organizations". ICSP 2009, 342-353
- [21] Cruz, R., Morales, M., Morgado, M. Oktaba, H., Ibargüengoitia, G., Pino, F., Piattini, M. "Supporting the Software Process Improvement in Very Small Entities through E-learning: the HEPAL! Project". ENC 09, in press.

# Scrum and Plan-driven Process Integration and its Impact on Effort Estimation

Nelio Alves<sup>a</sup>, William Carvalho<sup>b</sup>, Edgard Lamounier<sup>b</sup>

<sup>a</sup> Federal Institute of Triangulo Mineiro, Brazil

<sup>b</sup> Federal University of Uberlandia, Brazil

e-mail: nelio@iftriangulo.edu.br, william@facom.ufu.br, lamounier@ufu.br

**Abstract** – This paper analyses how the effort estimation of software development projects is impacted by integrating Scrum practices in traditional plan-driven process. A case study was carried out in a company in which several projects were analyzed according to a set of metrics. We made a comparison between a set of projects executed by a plan-driven process and another set of projects executed by the same plan-driven process streamlined using Scrum practices. We discuss the case study findings on effort estimation, which was characterized by recurring project overestimation on those projects in which Scrum practices were introduced.

## 1. Introduction

Agile method [3] adoption is an interesting choice to improve productivity in software industry [24]. It is attracting a great portion of software industry [2, 21]. Those methods incorporate agile practices like close customer involvement, short release cycles, collaborative team participation, fixed delivery schedules, and fast response to change requests.

However, agile adoption implies paradigm changes that affect the way several management and contractual issues are addressed [18], as summarized in Table I.

TABLE I  
Traditional vs. Agile Paradigms

Process	Traditional	Agile
Measure of success	Conformance to plan	Response to change, working code
Management culture	Command and control	Leadership / collaborative
Requirements and design	Big and up front	Continuous / emergent / just-in-time
Test and quality assurance	Big, planned / test late	Continuous / concurrent / test early
Planning and scheduling	Detailed, fix scope, estimate time and resource	Two-level plan, fix date, estimate scope

Some of those paradigm changes, however, are undesirable by some companies [18, 7, 11] and may cause much consternation in executive offices [18]. For governance politics or other needs, there are cases in which the customer wants or even needs early planning and contract among other plan-driven process practices. Thus, to take advantage of productivity and dynamics of agile practices without renouncing overall traditional process and management, an integration approach seems to be an interesting choice.

The objective of this research is to analyze how software development estimation is affected by integrating Scrum [27, 26] practices in a traditional plan-driven process structure.

We explore the hypothesis that agile-traditional integration is better for some types of customer-supplier relationships, particularly those in which at least one of them does not have governance process compatible with purely agile software contract and development principles. The contribution of this paper is to provide empirical results from a case study on effort estimation implications in that context.

The research was carried out in a CMMI-ML2 company that was used to execute a RUP-customized CMMI-adherent process in a traditional fashion - using early cost and schedule formal agreement. Since 2008, however, the company started to execute some projects using a hybrid process that integrates Scrum practices through the traditional-predominant development process. Metrics were extracted from several projects in order to evaluate how this Scrum-RUP integration affects software project effort estimation.

Guidelines from [15] were considered to carry out and report this research.

## 2. Related Work

This paper is about software development process or methodology, particularly addressing integration of agile and plan-driven methods in the context of Scrum. In this section we review main agile and plan-driven integration publications present in literature.

The scope of this paper is different from agile estimating research field, because our research deals with a traditional-predominant process with early effort estimation for customer-supplier agreement, which is fundamentally different from the agile philosophy as its proposal implies in a paradigm break of fixed scope and estimated schedule and cost. Instead, all work is intended to fit in a time box, where the time is fixed and scope emerges [18].

In general, there is great lack of empirical evidence regarding software engineering methods [1, 25, 24, 9]. Some studies can be found in integrating agility and plan-driven methods, which are briefly presented next.

Boehm and Turner [7] have proposed a framework for agile and discipline integration based on risk. They discuss two case studies and conclude that risk is the key to successfully

integrate agile and discipline. A qualitative case study in [11] reported success on integrating practices from software product line engineering and agile software development using Evo [10] as agile method. Kroll and MacIsaac [16] present a set of agile and plan-driven practices from OpenUP [22] and RUP [17]. Lastly, studies and discussions on agile methods in the context of software architecting can be found in [4, 20, 23].

Software development effort estimation is a broad topic, branching to several subtopics like regression, function point, neural networks, analogy and so on. We recommend [14] for an overview of recent advances.

There are publications that address effort estimation in the context of agile and iterative-incremental software development processes. A well-known publication in agile estimating and planning is [8]. In [5], a model for effort estimation is proposed for iterative-incremental process, which is derived from COCOMO [6]. A similar counterpart for use case points is [19].

Despite recent work done in agile estimating, we are not aware of any study focused on effort estimation implications of a hybrid agile and plan-driven software development process approach using Scrum.

### 3. Context

The case study was carried out in a medium-sized IT company focused in software project development, outsourcing and IT consulting for large enterprises. It was appraised as a CMMI-ML2 enterprise in 2006.

The company is used to execute a process based on the Rational Unified Process (RUP) [17] and its non-proprietary counterpart Unified Software Development Process (USDP) [13]. Since its effort on getting CMMI-ML2, the company executes its software development projects in a RUP-customized CMMI-adherent process.

The company is highly experienced in use-case requirements management and software development effort estimation using Function Point Analysis [12] as basis for sizing software functionality. Its customers usually demand prescriptive negotiation and planning for early contractual agreement. Many of its customers do not work with close interaction with the supplier throughout the project. Besides, many of their customers are not willing to fix date and let scope emerges as recommended by the agile philosophy. Rather, they prefer plan-driven fixed scope and estimated cost and schedule.

All software projects included in this case study consist of business automation systems, or specific parts of a bigger enterprise one.

For those particular features of the company, its costumers and projects, we decided that this company is well-suited for our empirical studies.

## 4. Research design

This research is of an exploratory character, which helps developing a general vision with certain phenomena in perspective, with the goal of identifying relevant variables that should be considered in the research. A quantitative approach was chosen because of the nature of the variables in study as we describe later.

The research fits in the method of case study research [28] as it consists of an in-depth investigation of the phenomena observed in one particular company.

Case study has the following limitations: the research consists of only one case and the results cannot be generalized to other contexts. This is a single company multi-project case study. It is not trivial to find a company that meets all necessary features we need like reliable process maturity, will to adopt the hybrid Scrum-RUP process framework practices we propose, permission to access process measurement data and so on. Despite the single case limitation, we consider this study is a good basis for reflection. Also, the reader may consider context information and be able to identify experiences that may be transferred to another software development organization.

### 4.1. Sample

Our sample consists of seven projects executed according to the proposed Scrum-RUP integration process. Those seven projects were executed from July/2008 to December/2009, i.e. 18 months. We consider this a reliable sample because possible strong discrepancies from adoption adaptation could be mitigated after about two or three projects, as reported by the company's practitioners, according to their own experience.

### 4.2. Control group

As control group, we use same data from seven other projects executed without Scrum integration, but only using the company's ordinary RUP-customized, CMMI-adherent process. Those control group projects were executed from January/2008 to Jun/2009, i.e. 18 months. Notice that there is an overlap period of one year in control group projects and sample projects. Also, notice that the overall time of the projects was two years long.

## 5. Data collection and analysis

The company keeps a set of development metrics, most of them gathered by software system automation. Also, there is a quality department responsible for managing and monitoring metrics for process and quality, especially since early years before the company's CMMI-ML2 appraisal. By the automation software and CMMI-ML2-appraised Measurement and Analysis (MA) process area they implement, we consider data collection completeness and accuracy satisfactory.

Data collection was made by retrieving convenient data from the measuring software system with proper query tools. Data was then tabulated into a Microsoft Excel spreadsheet and the same tool was used to perform statistical operations and extraction of proper summaries.

As mentioned in section 3, the company uses function point as measure for their software projects functionality. The effort of a particular requirement is calculated considering its functional size, technology, risk and historical productivity knowledge. The project functionality effort estimation is the sum of all its requirements effort estimations.

Below we formally describe measures and metrics adopted by the company so the reader can be able to understand how they calculate their metrics, as well as the discussions in this research.

### 5.1. Definition of measures

#### Technology (tech)

Definition	tech(req) :: {"Java EE", ".NET", "C++"}
Description	technology desired by a requirement.
Example	tech("req001") = "Java EE"

#### Current historical productivity (chp)

Definition	chp(tech) :: man-hour / fp
Description	current known average man-hours necessary to develop one function point of a requirement of a given technology. This measure is an estimate based on historical knowledge.
Example	chp("Java EE") = 12.5 mh / fp

#### Function point (fp)

Definition	fp(req) :: function point
Description	function point count of a requirement by using the IFPUG Method.
Example	fp("req001") = 6 fp

#### Requirement risk (risk)

Definition	risk(req) :: {"low", "medium", "high"}
Description	risk of a requirement, evaluated by the system analyst or software architect.
Example	riskrate("req001") = "medium"

#### Risk factor (rf)

Definition	rf(risk) :: numeric
Description	a multiplier for applying in effort estimation, depending on the risk rate of a requirement.

Example	rf("low") = 1 rf("medium") = 1.05 rf("high") = 1.1
---------	--

#### Requirement effort realized (rer)

Definition	rer(req) :: man-hour
Description	the real effort realized to develop a given requirement.
Example	rer("req001") = 83 mh

### 5.2. Definition of metrics

#### Requirement effort estimation (ree)

Definition	ree(req) :: man-hour
Description	effort estimation of a given requirement.
Calculation	ree(req) = chp(tech(req)) * fp(req) * rf(risk(req))
Example	ree("req001") = chp(tech("req001")) * fp("req001") * rf(risk("req001")) = chp("Java EE") * 6 * rf("medium") = 12.5 * 6 * 1.05 = 78.75 mh

#### Project effort estimation (pee)

Definition	pee(project) :: man-hour
Description	effort estimation of a project (project = list of requirements)
Calculation	1) pee([ ]) = 0 mh 2) pee([head:tail]) = ree(head) + pee(tail)
Example	-

#### Project effort realized (per)

Definition	per(project) :: man-hour
Description	the real effort realized to develop a project (project = list of requirements)
Calculation	1) per([ ]) = 0 mh 2) per([head:tail]) = rer(head) + per(tail)
Example	-

#### Requirement relative effort deviation (rred)

Definition	rred(req) :: numeric
Description	percentage that tells how the real effort spent on developing a requirement corresponds to planned effort. Positive value means delay (effort underestimation) and negative value means anticipation (effort overestimation).
Calculation	rred(req) = (rer(req) - ree(req)) / ree(req)
Example	rred("req001") = (83 - 78.75) / 78.75

	= 0.054 (5.4% delay)
--	----------------------

avg	166	1.28			2098	2111	0.07
-----	-----	------	--	--	------	------	------

\* weighted average

### Project relative effort deviation (pred)

Definition	pred(project) :: numeric
Description	Percentage that tells how the real effort spent on developing a project corresponds to planned effort. Positive value means project delay (effort underestimation) and negative value means project anticipation (effort overestimation).
Calculation	$\text{pred}(\text{project}) = (\text{per}(\text{project}) - \text{pee}(\text{project})) / \text{pee}(\text{project})$
Example	-

## 6. Findings and discussion

We collected raw data from each functional requirement for each project. For space constraints and for readability we are not going to present data in requirement degree of detail, though requirement detail might be mentioned during discussion if necessary.

Table II shows summary information of projects from the control group of the case study, i.e. those projects executed without Scrum. Last line is average.

Some information in Table II is noteworthy. First, both .NET projects registered the major project delays (underestimations). It may be caused by team low seniority in this particular technology at the moment the projects were executed. Second, the minimum -11.6% *pred* in C4 project may be caused by its risky nature (1.36 average risk), which increases time estimated to develop the product. So if the risks have not sanctioned development time, overestimation is naturally expected. Third, data suggests that this company is experienced in effort estimation and often does good estimates.

TABLE II  
Control group projects summary

Project	Functionality size (fp)	Weighted average risk	Technology	chp (mh / fp)	Project effort estimation (mh)	Project effort realized (mh)	Project relative effort deviation (%)
C1	169	1.21	Java EE	12.5	2144	2220	3.5
C2	100	1	Java EE	12.5	1250	1180	-5.6
C3	298	1	.NET	11	3278	3511	7.1
C4	169	1.36	Java EE	12.5	2171	1919	-11.6
C5	155	1	.NET	11	1705	1880	10.3
C6	122	2	C++	16	1882	1876	-0.3
C7	152	1.42	63%JavaEE 37%C++	13.8 *	2253	2188	-2.9

Table III shows summary information of sample projects that are object of this case study, i.e. those projects executed with Scrum practices integrated in plan-driven-predominant process. Last line, again, is average.

TABLE III  
Scrum-RUP hybrid projects summary

Project	Functionality size (fp)	Weighted average risk	Technology	chp (mh / fp)	Project effort estimation (mh)	Project effort realized (mh)	Project relative effort deviation (%)
S1	80	1	Java EE	12.5	1000	858	-14.2
S2	123	1.04	Java EE	12.5	1543	1372	-11.1
S3	128	1	.NET	11	1408	1329	-5.6
S4	159	1.16	Java EE	11.5	1843	1880	2.1
S5	234	1.2	Java EE	10	2378	2138	-10
S6	249	1.17	65%JavaEE 35%C++	11.4 *	2885	2666	-7.6
S7	168	1	Java EE	10	1680	1558	-7.3
avg	163	1.08			1820	1686	-7.67

\* weighted average

**S1, S2, S3:** The projects S1, S2 and S3 have their effort estimated with plan-driven chp (current historical productivity), i.e. the same parameters from the non-integrated-Scrum process. The project relative effort deviations, in those cases, were -14.2, -11.1 and -5.6 respectively. Those initial results have encouraged managers to reduce chp for next projects (notice that the less chp, the more productivity).

**S4:** For project S4, chp was reduced by 8% (chp("Java EE") = 11.5, instead of 12.5) in order to get an 8% man-hour lesser estimation to develop the project. This project had a delay of 2.1%, which is not considered a noteworthy underestimation.

**S5, S6, S7:** Then in projects S5, S6 and S7, the chp was reduced by 20% compared to original chp (chp("Java EE") = 10 instead of 12.5 and chp("C++") = 14 instead of 17), and the project relative effort deviations still remained less than -7 on all three cases (-10, -7.6 and -7.3 respectively).

Figure 3 show a graphical view of relative effort deviation of control group projects. Metrics suggest that effort estimation accuracy is stabilized in an ordinary statistical deviation (some underestimations and some overestimations in similar proportions).

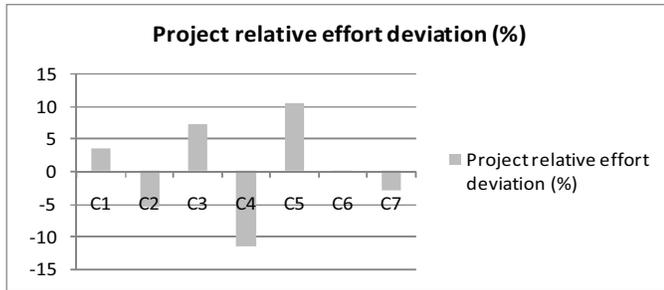


Figure 3. relative effort deviation of control group projects

Contrasting that, the same graphic from Scrum-RUP projects (Figure 4) shows a recurring negative relative effort deviation, which means that projects are being overestimated, even after managers reduced chp.

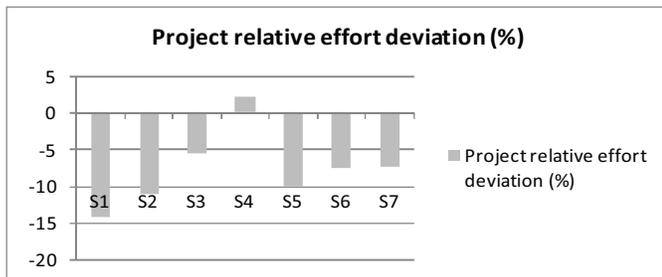


Figure 4. relative effort deviation of Scrum-RUP projects

Notice that, for instance, project relative effort deviation (pred) of project S5 was 10% as one can see in Table III and Figure 4. However, 10% is the deviation relative to the tighter 20%-reduced effort estimation. So an interesting question arises: what is the overall pred of project S5 if compared to the original chp (12.5 mh/fp) before integrating Scrum? To calculate the overall pred, one must consider the chp reduction (-20%) and its proportional effort deviation. Let us continue with the S5 project as example to show calculations:

- Original Java EE chp = 12.5 mh/fp
- Reduced Java EE chp = 10 mh/fp (reduced to 80%)
- S5 project relative effort deviation = -10% (reduced to 90%)

Thus, the original effort was reduced to 80% by the first estimation, and then reduced again to 90% when realized. The resulting reduction can be then calculated:

$$S5 \text{ project overall realized effort} = 80\% \times 90\% = 72\%$$

In other words, the overall project relative effort deviation of project S5, if considering original 12.5 chp, was 28%. Figure 5 summarize overall realized effort for all projects.

It is important to notice that all 7 projects have been overestimated, considering the original chp measure. Besides, the arithmetic average gain of this Scrum-RUP integration was 16.89%.

## 7. Conclusions

This case study research analyzed 14 software development projects - 7 as sample and 7 as control group - executed during two years in a CMMI-ML2 company, in order to examine effort estimation implications by integrating Scrum in a plan-driven-predominant development process.

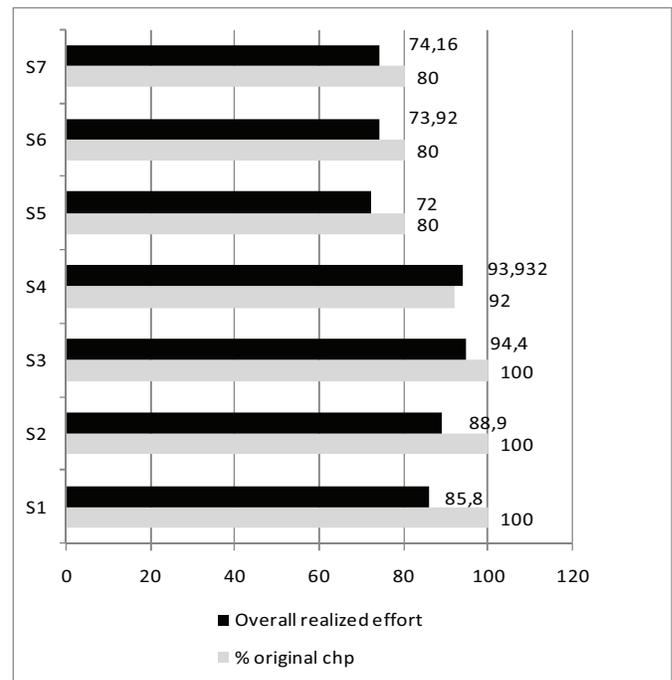


Figure 5. Overall realized effort of all Scrum-RUP projects

The empirical studies presented in this paper suggest that by integrating Scrum practices in a plan-driven predominant process approach, changes must be carried out in productivity parameters for software development effort estimation. Particularly, results demonstrated a recurring overestimation of projects in which Scrum was integrated (figure 5). The maximum pred of a single project was 28%, and the arithmetical average of all projects was 16.89%

Several factors influence in effort estimation, including functional size, risk, technology and historical knowledge and team productivity. In this research, we have explored the process factor. The evidence provided by this case study is a modest contribution if compared with the work that still has to be carried out to scientifically validate methodological claims, as mentioned in section 2. Still, we consider that this case study provided interesting results that point to a conclusion: that 15-20% effort estimation reduction is realistic when

integrating Scrum in a traditional plan-driven software development process.

The evidence provided by this case study may be useful for companies because it suggests that they can offer more competitive prices to their customers or increase profit by finishing projects earlier.

## 8. Future work

As future work, we are currently formalizing a framework for Scrum-RUP integration, and also examining comprehensive development metrics from the same company in order to present evidence of such an integration regarding productivity and quality. Besides, agile and plan-driven integration literature is still very sparse (see section 2), so much research addressing agile and plan-driven integration still need to be done in order to validate existing claims and explore others. Hybrid methodological approaches to provide agility benefits in customer-supplier prescriptive contract environments, outsourcing, offshoring, global software development and software product line engineering are interesting and little explored research areas.

## References

- [1] Abrahamsson, P., Warsta, J., Siponen, M. T., and Ronkainen, J. 2003. New directions on agile methods: a comparative analysis. In *Proceedings of the 25th international Conference on Software Engineering*.
- [2] Ambler, S. W. Dr. Dobb's Journal Agile Adoption Survey 2008. Available at (accessed on March 2, 2010): <http://www.ambysoft.com/surveys/agileFebruary2008.html>
- [3] Ambler, S. W. The Agile System Development Cycle. Available at (accessed on March 2, 2010): <http://www.ambysoft.com/essays/agileLifecycle.html>.
- [4] Babar, M. A. and Abrahamsson, P. 2008. Architecture-Centric Methods and Agile Approaches. In *Proceedings of the 9<sup>th</sup> International Conference on Agile Processes and Extreme Programming in Software Engineering*.
- [5] Benediktsson, O., Dalcher, D., Reed, K., and Woodman, M. 2003. COCOMO-Based Effort Estimation for Iterative and Incremental Software Development. *Software Quality Control* 11, 4 (Nov. 2003), 265-281.
- [6] Boehm, B. W., Clark, Horowitz, Brown, Reifer, Chulani, Madachy, R., and Steece, B. 2000. *Software Cost Estimation with Cocomo II with Cdrom*. 1st. Prentice Hall PTR.
- [7] Boehm, B. and Turner, R. 2003. *Balancing Agility and Discipline: A Guide for the Perplexed*. Addison Wesley.
- [8] Cohn, M. 2005. *Agile estimating and planning*. Prentice Hall.
- [9] Dybå, T. and Dingsøy, T. 2008. Empirical studies of agile software development: A systematic review. *Inf. Softw. Technol.* 50, 9-10 (Aug. 2008), 833-859.
- [10] Johansen, T. 2005. Using evolutionary project management (Evo) to create faster, more userfriendly and more productive software. Experience report from FIRM AS. In *Proceedings of the International Conference on Product Focused Software Process Improvement*. (Oulu, Finland). Springer Verlag.
- [11] Hanssen, G. K. and Fígri, T. E. 2008. Process fusion: An industrial case study on agile software product line engineering. *Journal of Systems and Software*. 81, 6 (Jun. 2008), 843-854.
- [12] ISO/IEC 20926:2009 - Software and systems engineering - Software measurement - IFPUG functional size measurement method 2009. Available at (accessed on March 02, 2010): [www.iso.org/iso/catalogue\\_detail.htm?csnumber=51717](http://www.iso.org/iso/catalogue_detail.htm?csnumber=51717)
- [13] Jacobson, I., Booch, G., and Rumbaugh, J. 1999. *Unified Software Development Process*, Addison-Wesley.
- [14] Jorgensen, M. and Shepperd, M. 2007. A Systematic Review of Software Development Cost Estimation Studies. *IEEE Trans. Softw. Eng.* 33, 1 (Jan. 2007), 33-53.
- [15] Kitchenham, B. A., Pfleeger, S. L., Pickard, L. M., Jones, P. W., Hoaglin, D. C., Emam, K. E., and Rosenberg, J. 2002. Preliminary guidelines for empirical research in software engineering. *IEEE Transactions on Software Engineering*, 28, 8 (Aug. 2002), 721-734.
- [16] Kroll, P. and MacIsaac, B. 2006. *Agility and Discipline Made Easy – Practices from OpenUP and RUP*. Addison-Wesley.
- [17] Kruchten, P. 2003. *Rational Unified Process: An Introduction*, 3rd ed. Addison-Wesley.
- [18] Leffingwell, D. 2006. *Scaling Software Agility*. Addison-Wesley.
- [19] Mohagheghi, P., Anda, B., and Conradi, R. 2005. Effort estimation of use cases for incremental large-scale software development. In *Proceedings of the 27th international Conference on Software Engineering*.
- [20] Nord, R. L. and Tomayko, J. E. 2006. Software Architecture-Centric Methods and Agile Development. *IEEE Software*. 23, 2 (Mar. 2006), 47-53.
- [21] North American and European Enterprise Software and Services Survey, Business Technographics Ed., 2005.
- [22] OpenUP – Available at: <http://epf.eclipse.org/wikis/openup/> (accessed on March, 10, 2010)
- [23] Parsons, R. 2008. Architecture and Agile Methodologies - How to Get Along. In *Proceedings of the IEEE/IFIP Conference on Software Architecture – WICSA'2008*.
- [24] Ramsin, R. and Paige, R. F. 2008. Process-centered review of object oriented software development methodologies. *ACM Comput. Surv.* 40, 1 (Feb. 2008), 1-89.
- [25] Rombach, D. and Seelisch, F. 2007. Formalisms in Software Engineering: Myths Versus Empirical Facts. In *Second IFIP TC 2 Central and East European Conference on Software Engineering Techniques*.
- [26] Schwaber, K. 1995. Scrum development process. In *Proceedings of the 10th Annual ACM Conference on Object Oriented Programming Systems, Languages, and Applications*.
- [27] Schwaber, K. and Beedle, M. 2001. *Agile Software Development with Scrum*. Prentice-Hall.
- [28] Yin, R. K. 2002. *Case Study Research: Design and Methods*. 3rd ed. Sage Publications.

# A Case Study of Software Process Improvement Implementation

Daniela C. C. Peixoto<sup>1</sup> Vitor A. Bastista<sup>2</sup> Rodolfo F. Resende<sup>1</sup> Clarindo Isaías P. S. Pádua<sup>2</sup>

<sup>1</sup> Department of Computer Science – Federal University of Minas Gerais, Brazil

<sup>2</sup> Synergia, Department of Computer Science, Federal University of Minas Gerais, Brazil

Av. Antônio Carlos, 6627 – Pampulha – Belo Horizonte – MG – Brasil

{cascini, vitor, rodolfo, clarindo}@dcc.ufmg.br

**Abstract**—Managing a successful Software Process Improvement (SPI) is a challenging issue that many software companies face today. Many companies have invested huge amount of money in improving their software processes. This can be confirmed through several papers that present the results of SPI programs. However, as pointed out by the literature, many of these programs have encountered difficulties to achieve the desired benefits. This is usually not caused by incorrect new processes, but due to inadequate SPI implementation and, particularly inadequate SPI adoption. This paper evaluates an SPI program, discussing its implementation problems with emphasis on adoption issues. The analysis was carried out as a case study in a software development organization. Our findings suggest that other than finance, technology and other issues, several implementation aspects, in particular effective adoption strategy, are needed to achieve a successful SPI program. Our main contribution is to give evidences that an SPI implementation process can be assessed and improved using objective measurements and available methods and practices. In particular, we measure the adoption of an SPI practice, analyzing the evolution of the improved practices rework during the development of two software projects.

**Keywords**- *Software Process Improvement - SPI; SPI Implementation; SPI Adoption; Rework Evolution*

## I. INTRODUCTION

Many organizations experience a successful start on their Software Process Improvement (SPI) initiative and after some steps in their improvement process they realize that the overall engagement to change weakens significantly after the initial excitement [1]. Previous studies show critical factors that can be sources for this problem e.g. resistance to change, previous negative experience, lack of the evidence of benefits, imposition, resource constraints and commercial pressures to meet customer demands [2], [3].

Many advances have been made in the development of the SPI standards and models. However, we cannot observe equal advances in the SPI implementation or adoption processes. This has resulted in limited success of many SPI programs [4]. Herbsleb and Goldenson [5] showed that 67% of SPI managers prefer guidance on how to implement SPI activities, rather than additional SPI (*what*) activities to implement.

Despite the importance of SPI implementation, little empirical research has been carried out analyzing in detail

the implementation steps of SPI new processes, specifically in evaluating the SPI adoption.

The purpose of this article is to describe, via a case study, the context of the SPI implementation, with emphasis on adoption issues. We show that the challenges of a successful SPI program are not limited to the identification and creation of new processes. In addition, we verified that there are many barriers to SPI which can be well characterized after the assessment of the SPI implementation. We also present an objective measurement to evaluate the SPI adoption. This measurement can provide for organizations a better understanding of where they are, when considering the SPI implementation process. Considering the successful results of this initiative, we believe that it should be easily adopted by many software development organizations.

The focus of this paper is based on the following research question:

*How SPI implementation can be assessed and can incorporate empirical measurements?*

To answer this question we conducted our case study in a software development organization, evaluating an SPI initiative that started in 2007. During two years, we observed that to be successful, an SPI implementation needs to recognize human, social and organizational aspects with the same or higher consideration given to finance, technology and other issues.

The remainder of this paper is structured as follows. Section 2 describes the background of this work. Section 3 describes the case study and some findings. Section 4 presents the final remarks and future works.

## II. BACKGROUND

### A. SPI Adoption and Implementation Process

A recurrent problem reported in the literature for SPI is a lack of an effective strategy to successfully implement SPI programs. We observed that much attention has been paid to what activities to implement instead of how to implement these activities. In this context, Niazi et al. [4], [6] developed a framework that resulted from SPI literature research and empirical studies. In the design of the framework, Critical Success Factors (CSF) and Critical Barriers (CB) for SPI were identified and extended. In addition, interviews were conducted with Australian practitioners in order to select and analyze the factors that

play positive or negative roles in SPI organization. The framework is composed of three components:

- **Maturity stage dimension:** The IMM (Implementation Maturity Model) is adapted from CMMI [7] and presents four maturity levels. Level 1, “Initial”, indicates that the organization is not aware of SPI implementation process. Level 2, defined as “Aware”, indicates that management and practitioners are becoming aware of SPI implementation programs and their benefits. Level 3, “Defined”, is the level where processes were documented, standardized and integrated into the organizational process. Level 4, “Optimizing”, is the level where the organization establishes structures for continuous improvement.
- **CSFs and CBs dimension:** Instead of the Process Areas (PA) as adopted by CMMI, IMM adopted CSFs and CBs which are organized into three categories “awareness”, “organizational” and “support”. Each category provides a list of related CSF and CB. The categories are divided among the levels of the IMM.
- **Assessment dimension:** This dimension provides a list of practices that allows assessing how well the factor has been implemented in practice.

Niazi and others [6] conducted three case studies in order to test and evaluate the framework. The results of their experimental evaluation have shown that the framework has potential to assist practitioners in the design of effective SPI implementation initiatives. However, their work focuses only on the process dimension of SPI implementation. We believe that for a complete evaluation of a successful SPI implementation it is also important to recognize human, social and organizational dimensions. This means evaluating the adoption of the improvement by the practitioners: developers and managers.

It is not easy to analyze if the improved processes have been fully adopted. It is almost impossible to measure if processes have really changed the hearts and the minds of the practitioners [8]. Process adoption is generally characterized as two orthogonal dimensions: infusion and diffusion [8]. Infusion describes how deeply the new process has been adopted by the target population. Diffusion describes how broadly the target population has adopted the new process [9].

Adoption of a change typically follows a number of stages. In our work, in order to evaluate the human dimension of the SPI implementation process, we applied the seven-phase adoption curve discussed by Conner and Patterson [10]. The seven-phase consists of: contact, awareness, understanding, trial use, adoption, institutionalization and internalization.

### B. Organizational Context

Here we describe a work that was performed as a collaborative study with Synergia. Synergia is a laboratory for software and systems engineering, hosted in the

Computer Science Department at Federal University of Minas Gerais, Brazil. Synergia is internally organized as a commercial software development organization, but it also retains important academic characteristics [11]. Synergia maintains 85 people in its staff composed by undergraduate and graduate students and non-student graduated professionals, most with a Computer Science background.

Synergia uses a tailored version of the Praxis model-driven software development process [12] in its software and systems engineering projects, called Praxis-Synergia. Although the Praxis process has been designed and applied primarily for education and training in Software Engineering, it provides tailoring guidelines, which must be interpreted according to the current situation of the development organization. The Praxis material is available in a book and kept updated in the author’s Web site<sup>1</sup>.

One important characteristic of Praxis, which is maintained in the tailored version, is that it models the development of each product use case as a sequence of development states or control marks. The name of each state evokes how far the use case has advanced towards complete implementation and acceptance. The states are: Identified, Detailed, Analyzed, Designed, Specified, Realized, Implemented, Verified, Validated and Complete. Pádua [11] discusses these states and how each state is associated with one or more quality control procedures such as inspection, test and management review.

## III. STUDY DESIGN AND EXECUTION

We performed a case study in conjunction with a SPI initiative conducted at Synergia. This case study analyses the rework effort and the number of defects detected in a specific test execution task in different projects at Synergia. The aim is to provide more evidences about the results achieved with the process improvement implementation assessment and the empirical SPI adoption measurement.

This is an embedded case study [13] in one company with two units of analysis: two major projects. For every use case of each project we collected the following information: the percentage of rework related to the test execution ( $\% \text{ of rework} = \text{rework} / (\text{work} + \text{rework})$ ) and the number of defects.

### A. Process Improvement

In this section we summarize the SPI initiative that was conducted at Synergia. A detailed description of this program can be found in Peixoto et al. [14].

During 2007, Synergia managed its improvement effort by analyzing problems in a specific project. The work was organized as a Defect Causal Analysis (DCA) process [15]. The DCA process selects recurring problems with the aim of identifying their principal causes and propose some solutions to avoid (or reduce) new occurrences. At that

---

<sup>1</sup> [www.dcc.ufmg.br/~wilson/praxis/](http://www.dcc.ufmg.br/~wilson/praxis/) (in Portuguese)

time, the largest existent project (~ 5,000 Function Points), referred here as Project TBI (To-Be-Improved), was chosen to assure that the recommendations would be executed.

The problem discussed ahead was detected while performing the quality procedures of the Implemented state. Before delivering the executable code to the Test team, the developers themselves were required to execute a list of test procedures (a subset of the complete manual test specification), referred here as verification practice. The purpose of this verification practice was to identify errors made by the developers themselves (for example, a nonoperational save command) with the aim of treating them before the execution of the complete manual test specification by the Test team and avoiding rework.

Although in Project TBI all developers were required to execute the list of test procedures, they did not execute it correctly or they did not execute it at all. In a broader evaluation, the non-execution of these test procedures could be the result of time pressure, project cost overrun, and lack of training or experience of the team. But at Project TBI, the developer had enough time to do their task and they knew or were trained in what they had to verify. Trying to find one possible cause, the developers in a DCA meeting were asked about the reasons. It was not possible to identify one specific source of the problem. However, observations during the meetings drew attention to a specific team behavior: they simply neglected this task. But, why this happened? Even if the project manager had required the developers to execute this verification practice they did not execute it! One rationale behind this problem could be resistance to change, or in other words, the resistance to include this practice in the developers list of 'daily' practices. As Humphrey observed [16] this is not a trivial problem "particularly because even intelligent people often will not do things that common logic, experience, and even hard evidence suggests that they should". Trying to understand this resistance to change behavior, we found some studies [17], [18] that show how cultural aspects can affect software process improvement programs.

Considering this hypothesis, one adopted solution was to force the developers to execute the test procedures using a specific tool called IBM Rational Manual Tester, referred here as RMT, which records each step executed and the test result (whether it succeeded or failed). In this way, the developers would provide to the Test team a proof, metaphorically like signing a document, that they had really executed the test procedures.

After five months of implementation, the benefit of this action amounted to 217% of the investment in DCA meetings. At the end of Project TBI, in 2008, there was an average reduction of 5.66 hours per use case of rework to fix and check for defects detected by the Test team, resulting in total savings of 796 hours. The number of use cases without any defects increased from 21% to 53%, with this specific verification practice conducted by the Test

team. The average number of defects detected by the Test team reduced from 5.1 to 1.7 defects per use case and the number of critical defects was reduced by 70%.

The other project, referred here as Project IMPROVED, started in 2009 and ended in January, 2010. It had approximately 1,500 Function Points and it presented the same characteristics of project TBI. Almost all developers that worked in Project IMPROVED came from Project TBI.

Comparing the rework originated from the verification practice in the two projects we observed that:

- With 95% of confidence, despite all the rework and number of defect reductions, we cannot say that there is a difference between the number of defects per use case before and after the process improvement in Project TBI (Fig. 1).
- With 95% of confidence, we can say that there is a difference between the number of defects per use case comparing Project TBI and Project IMPROVED. (Fig. 2)

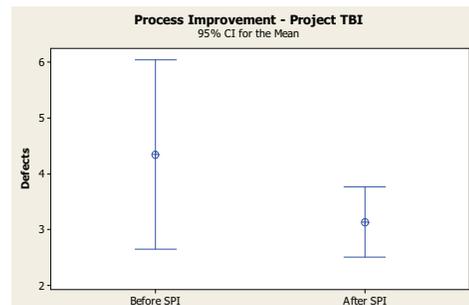


Figure 1. Interval plot from Project TBI.

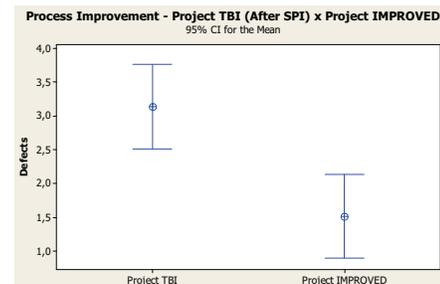


Figure 2. Interval plot from Project TBI and Project IMPROVED.

In the next sections, we evaluate the SPI implementation process trying to understand the modifications in the number of defects and rework effort in both projects.

### B. SPI Assessment

We assessed the SPI initiative in two phases. First we evaluated the SPI implementation process using the IMM model [4] [6]. In this case, we evaluated only Project TBI, which was the first project to adopt the improved practice (the verification practice). Second, we evaluated the improvement adoption by the Development team in both projects. Then we documented the lessons learned in both phases.

1) *SPI Implementation Process Assessment*: In spite of Project TBI improvements, some use cases still presented unreasonably high number of defects, though much less critical ones. We conducted a semi-structured interview with each developer that had defects reported for their use case, trying to find the reasons. We noticed that:

- Some developers generated a log without carrying out all the test procedures. RMT has a command that allows users to select all the test procedures and mark them as successful. The developers said that some test procedures were so extensive and detailed that the purpose to be a simple verification was missed.
- Some developers reported that some test procedures specified by the Test team were incorrect. So when the Test team executed them, they reported improper errors.
- Some developers reported that the requirements changed and the test procedures were not properly updated.

To highlight the improvement process problems, the Software Engineering Process Group (SEPG) conducted a *post-mortem* process assessment of the SPI implementation process. Specifically, they evaluated this verification practice and its implementation problems using the IMM framework. The process maturity assessment method involved assessing the SPI implementation process in the organization and using the results to determine a process maturity level.

The SEPG filled the assessment form together with some of the practitioners that participated of the DCA. Table I summarizes the results of the assessment. The complete IMM practice list can be found in Niazi et al. [4].

TABLE I. SUMMARY OF RESULTS.

Maturity Level	Weak implementation factors
Level 2 – Aware	Awareness of SPI, Staff involvement and Senior management commitment
Level 3 – Defined	Creating process action teams, Experienced Staff, Staff time and resources, Time Pressure and Organizational Politics
Level 4 - Optimizing	Reviews

Our suspicions that the organization was operating at low maturity levels were confirmed. Considering the results, the organization stands at Level 1, “Initial”, of IMM because three factors of Level 2 were not fully implemented (score less than 7). The Critical Success Factors have an average score of 6. Some considerations of the results are:

- Level 2: our attention was drawn to the first problem detected during the developers’ interviews which is reflected in the factor “awareness of SPI”. We noticed that the benefits of the SPI program were promoted only before the SPI implementation. In addition, the awareness was not stimulated after the implementation in Project TBI. Related to the staff involvement and senior management commitment, the low score was obtained mainly because some managers were not involved in the SPI process.

- Level 3: the main concern in this level is to have documented, standardized and integrated SPI implementation processes into the organizational process. As expected in a Level 1 organization, most of the initiatives were *ad-hoc* and non-standardized.
- Level 4: as the process is not documented and integrated, no revisions were planned to be conducted during the SPI initiative. In this way, it was not possible to establish continuous improvement.

We were surprised that the organization successfully used good practices, as formal methodologies (for example, testing an improvement firstly in a pilot project and providing training) in spite of the fact that it did not really have a strong base of fundamental practices. This may suggest that the notion of what is an advanced practice is wrong or that the organization can successfully use such practices without having the other practices implemented beforehand. In addition, we were aware that without quantitative information it would not be possible to promptly identify SPI implementation problems and provide the corrective solutions, as it happens to be very difficult to truly change the behavior of engineers and managers.

Some important lessons learned during this assessment were:

- The IMM provides a good methodology to evaluate the weak and strong implementation factors. Also it provides a way for an organization to improve its practices of SPI implementation process.
- It is crucial the commitment of the people involved in SPI. The SPI literature [19] recognizes that without commitment from all organizational levels the improvement goals will be difficult to achieve. So, people involved with the SPI initiative must perceive the benefits deriving from its deployment, and not only its costs. Also it is important to overcome the resistance of the team developers to adopt the new practices. This involves supporting them to give up established ways of working.
- It is important to take measurements throughout the whole process, including adoption measurement in order to determine a successful adoption of the new practices.

2) *Adoption measurement*: With the aim of measuring how far the target population has come along the stages of the adoption curve we used an adoption measurement of the SPI implementation. Instead of calculating the number of people who are at a certain stage of the adoption curve and give weights to each stage, as proposed by Heijstek and van Vliet [9], we evaluated the evolution of the rework of the execution of the verification practice by the Test team. Since we did not have data about the team knowledge in previous projects, we carried out this *post-mortem* analysis. Based on our historical data and using expert judgment, we determined a percentage of accepted rework for each stage, considering this specific verification (Table II). In this way,

we believe the institutionalization happens when the use cases do not present (significant) rework. A clear understanding of the nature of rework could lead to stronger support for the evaluation of the implementation process.

The reasons for our measurement are:

- The test procedures verified by the developers are detailed and objective. So, if developers carried out this verification, no defects should be detected by the Test team, when carrying out the same test procedures.
- The tool that supports the verification generates a log of the test procedures execution. Before conducting the verification, the tester should verify the tool log. If there are problems, the developer would be notified to solve them before delivering the code.

TABLE II. MEASURES FOR EACH STAGE OF THE ADOPTION CURVE.

Stage	Measurement
Contact	No changes are observed in the rework. Our historical data shows that we have usually 45% of rework during the test procedures verification.
Awareness	The rework reduced but not expressively (~5%).
Understanding	The rework reduced, but not expressively (6-10%).
Trial Use	Because more developers adopted the tool, the rework reduced in 11-20%.
Adoption	Because all developers adopted the tool, the rework reduced in 21-25%.
Institutionalization	Reduced rework, between 0- 20%.
Internalization	-

The evolution of the rework percentage computed during the verification practice carried out by the Test team is represented in Fig. 3. The medium value of the rework for project TBI was calculated for each month starting in 2007 and ending in the middle of 2008, and, for project IMPROVED, starting in 2009 and ending in January 2010. The area labeled “A” shows data before the improvement. This period presented the highest rework percentage. The area labeled “C” shows the lowest rework percentage and corresponds to Project IMPROVED. Fig. 4 presents similar data related to areas “B” and “C” of Fig. 3.

We observed that the percentage of rework presented a value equal or less than 20%, only in areas “C” and “D” of Fig. 4. Besides the usage of RMT tool in Project TBI by all the developers, we cannot observe an expressive reduction of rework. This is due to the SPI implementation problems, specifically SPI adoption problems. As discussed in the previous section, after analyzing the data and after the interviews we observed that two important factors were: the awareness of the SPI (process factor) and also the resistance of the team to change their behavior (human factor).

The human factor plays an important role during the SPI implementation program. We observed that SPI implementation had several flaws, but the resistance to change affected directly the rework percentages.

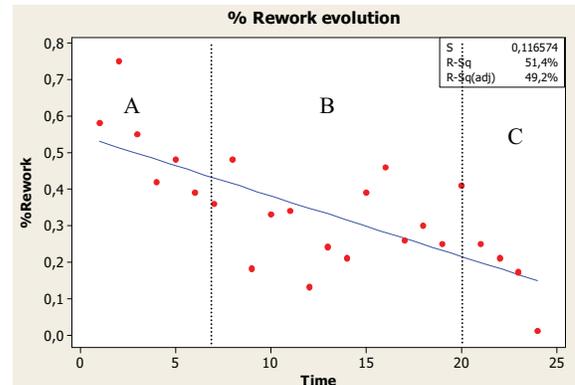


Figure 3. Monthly rework percentage. A) Project TBI - before the improvement implementation. B) Project TBI- after the improvement implementation. C) Project IMPROVED.

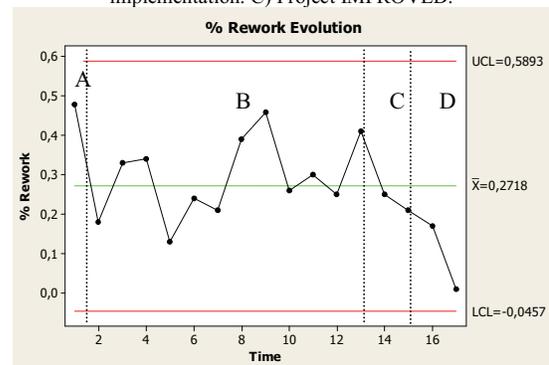


Figure 4. Monthly evolution of the rework percentage. A) Data of the first month after the improvement – Project TBI. B) Data from Project TBI. C) and D) Data from Project IMPROVED.

Some important lessons learned during the adoption measurement are:

- Determining an effective way to evaluate the adoption of a certain technology or methodology is not an easy issue. We need to consider factors that are difficult to measure, as the level of knowledge of each team member in that subject. We observed that in both projects the main source of rework problems were the human behavior.
- It is possible to develop alternative ways of evaluating the adoption, which can show an interesting evolution pattern during some period of time.
- The adoption of a simple change in the process can take a long time to be really effective, as we presented in the previous discussion. Only in Project IMPROVED, after an improvement in the way that the test procedures were specified, was that the practitioners achieved the “Institutionalization” stage.

### C. Limitations

There are some limitations in our study. First, our data come exclusively from Synergia. In addition, this study needs to be replicate in other organizations to provide a clear picture of the general application of the results.

However, the SPI implementation and adoption problems seem to be similar to that of comparable companies.

Second, we showed that it is possible to develop alternative ways of evaluating the adoption. However, this can have external influence of other factors, for example, team turnover, requirements and processes change. We tried to reduce this threat to validity triangulating the data with other sources such as interviews and management evaluations.

Finally, the team might not express their real opinions during the interviews and meetings. This threat was limited by participants being guaranteed anonymity and being shown the benefits that one single improvement can bring to the organization.

#### IV. CONCLUSION

Our work revealed some fundamental difficulties of assessing process maturity and adoption of software process improvement practices.

The IMM model allows the organization to identify factors that have positive and negative impact on implementing SPI, and it also serves as a guide to improve the whole SPI implementation process. Besides the process assessment, another important dimension that needs to be evaluated during an SPI implementation program is the human dimension. In this paper, we applied an adoption measurement approach to evaluate how deeply the new process has been adopted by the target population. We observed that with this measurement it is possible to evaluate when the process improvement really achieve the adequate values expected by the organization.

The key lesson we learned is that while it was reasonably easy to identify areas where improvement is needed, it is much more difficult to understand the main cause of the implementation problems, and how to identify and propose reasonable solutions. Other important observation is that since change is not easily accepted by everyone in a team, the focus in this first phase should be to get people excited about the benefits of the change instead of managing resistance.

In conclusion, we should define assessment methodology with a broader vision of the problems that could affect the SPI implementation. Experiences with other organizations could help us in identifying alternative ways to evaluate the adoption of an SPI implementation process. As a future work, we will evaluate other process improvements implemented in the organization with the aim to compare implementation process problems and the adoption measurement alternatives.

#### ACKNOWLEDGMENT

We would like to thank Capes Foundation grant n. BEX 1893/09-2 and Fundação de Amparo a Pesquisa de Minas Gerais (FAPEMIG) grant CEX - APQ-01201-09.

#### REFERENCES

- [1] A. Börjesson, and L. Mathiassen. "Making SPI Happen: The IDEAL Distribution of Effort". In Proceedings of the 36th Annual Hawaii International Conference on System Sciences, IEEE Computer Society Press, 2003, pp. 328b.
- [2] N. Baddoo, and T. Hall. "De-motivators for software process improvement: an analysis of practitioners' views". *Journal of Systems and Software*, 2003, 66 (1), pp. 23–33.
- [3] M. H. N. Md. Nasir, R. Ahmad, and N. H. Hassan. "Resistance factors in the implementation of software process improvement project". *Journal of Computer Science*, 2008, 4(3), pp. 211-219.
- [4] M. Niazi, D. Wilson, and D. Zowghi. "A maturity model for the implementation of software process improvement: an empirical study". *Journal of Systems and Software*, 2005, 74(2), pp.155-172.
- [5] J. D. Herbsleb, and D. R. Goldenson. "A Systematic Survey of CMM Experience and Results". In Proceedings of the 18th International Conference on Software Engineering, 1996, pp. 323-330.
- [6] M. Niazi, D. Wilson, and D. Zowghi. "A framework for assisting the design of effective software process improvement implementation strategies". *Journal of Systems and Software*, 2005, 78(2), pp.204-222
- [7] SEI. "CMMI for Development". CMMI-DEV. Version 1.2. SEI CMU/SEI-2006-TR-008. 2006
- [8] S. Garcia. "Are You Prepared for CMMI?" *The Journal of Defense Software Engineering*. SEI-2002.
- [9] A. Heijstek, and H. van Vliet. "Measuring the Adoption of Software Processes". In Proceedings of the 3rd Software Measurement European Forum, 2006, pp. 1-14.
- [10] D. R. Conner, and R. W. Patterson. "Building Commitment to Organizational Change". *Training and Development Journal*, 1982, 36(4) pp. 18-26.
- [11] C. Pádua, B. Pimentel, W. Pádua, and F. Machado. "Transitioning model-driven development from academia to real life". In Proceedings of Educators' Symposium of the ACM / IEEE 9th International Conference on Model Driven Engineering Languages and Systems, 2006, pp. 61-77.
- [12] W. Pádua. "A Software Process for Time-constrained Course Projects". In Proceedings of the 28th International Conference on Software Engineering, 2006, ICSE'06, pp. 707-710.
- [13] R. K. Yin. "Applications of Case Study Research". Second Edition. Volume 34. 2003. Sage Publications
- [14] D. C. C Peixoto, V. A. Batista, R. F. Resende, and C. Pádua. "How to Welcome Software Process Improvement and Avoid Resistance to Change". To appear in International Conference on Software Process, 2010, accepted for publication.
- [15] D. N. Card. "Learning from Our Mistakes with Defect Causal Analysis". In: *IEEE Software*, 1998, 15(1), pp. 56–63
- [16] W. S. Humphrey. "Why don't they practice what we preach?" *Annals of Software Engineering*, 1998, 6(1/4), pp. 201–222
- [17] K.V. Siakas. "What has Culture to do with SPI?" In Proceedings of the 28th Euromicro Conference, 2002, pp. 376-381
- [18] B. Wong, and S. Hasa. "Cultural Influences and Differences in Software Process Improvement Programs". In Proceedings of the 6th International Workshop on Software Quality, 2008, pp. 3-10
- [19] P. Abrahamsson. "Commitment Development in Software Process Improvement: Critical Misconceptions". In Proceedings of the 23rd International Conference on Software Engineering, ICSE'01, 2001, pp. 71–80.

# Software Process Reuse by Pattern Weaving

Ya-sha Wang, Xiao-yang He, Jin-gang Guo, Jia-rui Jiang

Institute of Software, EECS, Peking University

Key laboratory of High Confidence Software Technologies (Peking University), Ministry of Education

Beijing 100871, China

wangys@sei.pku.edu.cn

**Abstract**—Developing software processes is a knowledge intensive, time-consuming and error prone work. Reusing is a promising way to increase the quality and productivity of process models. A pattern based software process reuse method is proposed in this paper. In this method, the process patterns are considered as reusable assets, each of which encapsulates some process knowledge in the form of a common solution to a given problem or concern in some particular context. When building a project-specific process model, some patterns will be chosen according to the project’s context and problems, and then the chosen patterns will be applied to an organizational standard software process model (OSSP). Since the solutions of some patterns cross-cut into the OSSP, a weaving approach is also given.

**Keywords**—software process; process pattern; process reuse; cross-cutting concern

## I. INTRODUCTION

Developing an appropriate process model for a specific project is a knowledge intensive, time-consuming and error prone work. Research data shows that effort spent to design processes for a KPA at around 800 to 1000 or more person-hours [5].

In order to increase the quality and productivity of software process models, people try to utilize the method of reuse. The basic idea of process reuse here is to build different projects’ process models with some reusable assets. Delighted by Osterweil’s famous conclusion “software processes are software too” [3], some researchers tried to apply the idea of CBSD (Component Based Software Development) on software processes. These works led to the emergence of the concept process component. A process component is a modular fragment of process model and is intended to be reused in the development of multi process models [4,6].

Although CBSD seems successful in software, but the variety of software components, process components, are not suitable to handle cross-cutting concerns which are ubiquitous in software processes. Hence, a pattern based software process reuse method (PBSPR) is proposed in this paper. Here one process pattern represent a general solution to some recurring problems or concerns in different projects’ scenarios [1, 2]. Unlike the process components the process patterns are not modular fragments of process models, they are a set of constraints of activities, roles and artifacts.

In PBSPR, a meta-model of process pattern is given for pattern modeling, and a pattern reuse approach is also proposed. By this pattern reuse approach, the reusable process patterns can be reused by means of weaving all their constraints into an organizational standard process model to generate project specific process models. Here weaving means to add or change activities, roles, artifacts and their relations through different sub-processes in a existing process model to make sure that all constrains of a pattern is satisfied by the resulting process model. The patterns to be reused are chosen according to the projects’ characteristics, and the pattern choosing criteria are omitted here because of limit of length of this paper.

In the following of this paper, we will discuss the cross-cutting concerns of software process modeling and improvement, and explain the reason why a process pattern is more reusable than common process components in section 2, and then introduce the meta-model of our process pattern in section 3, and introduce our pattern reuse approach in section 4. In section 5 we give a case study of our method. In section 6, related works are discussed. At last, in section 7, we conclude this paper and give out the future works.

## II. BACKGROUND

In a process reuse method, the first thing to ask is often what is the reusable asset or in what way the reusable assets are organized. Delighted by CBSD, some researchers take process components as reusable assets. But, process components are not suitable to handle cross-cutting concerns which are ubiquitous in software processes. Consider the development of a software process has the following steps: First, a common framework is provided by the organizational process standard. Second, the characteristics of the given project are analyzed, and as the result some concerns that should be solved in the developing process model are identified. Third, common solutions of these concerns are found from literatures or worked out according to process engineers’ personal knowledge. At last, these common solutions are applied to above common framework to build a project specific process model. Because software development is a knowledge-intensive work and need close cooperation between different stakeholders, there exist a lot of cross-cutting concerns in the development of software process. Here, by cross-cutting concern we mean that when apply the solution of this concern to an existing process model, more than one module (i.e., a sub-process, or a single

activity) should be changed. Reducing-the-risk-of-intellectual-property-problem-caused-by-misusing-open-source-codes (shorted as Handling-IP-Problems) is an example of the cross-cutting concern in software process development. According to one common solution of this concern we summarized from three Chinese companies, at least two sub-processes of the existing process model should be affected: In project management sub-process, an extra role of IP lawyer and an extra artifact of potential-IP-problem-list should be added, and the activities of risk analysis and project planning should be changed to cope with the IP problems; in another sub-process of development, an extra tool of code scanner which can help people find out the usage of open-source codes should be added, and the activity of code review should be changed to address the IP issue. Since the solution of the concern Handling-IP-Problem affects two sub-processes, we name it a cross-cutting concern delighted by AOP (Aspect Oriented Programming). The cross-cutting concerns are very hard to handle if we organize a process model as a composition of a couple of modular process components. In order to apply a solution to a cross-cutting concern, several components should be changed or replaced. The ubiquitous cross-cutting concerns in software process will make the structure of the composition too complex to be understood and managed. Furthermore, the solution for a cross-cutting concern itself is a logical entity and usually contains valuable and reusable knowledge. But these logical entities can't be modeled as components.

Based on the analysis of the problems of process components, we decide to use process patterns as reusable assets. Unlike the process components, process patterns are not modular fragments of process models, they are a set of constrains of activities, roles and artifacts. The basic underlying idea of our process pattern is based on a simple observation: a common solution of a process development concern, such as a best practice or process guidance, take different forms when applied on different project-specific process models, but if we absorb these process models very carefully, we can find that some basic features that represent the substitutive characteristics of the common solution are recurrent. We model these recurrent features as a set of constrains. For example, figure 1 shows three feasibility analysis process fragments cutting from three different project specific process models. From the figure, we can tell that these process fragments are quite different. But if we study them very carefully, we can find that some constrains are satisfied in all of them. These constrains are: 1) the activity Identify alternatives is always in the beginning of the performing sequence while Choose an alternative is always at the end; 2) In the middle of the sequence although the order of activities are different one process model from another, but there always exist four activities including Determine operational feasibility, Determine economic feasibility, Determine technical feasibility, and Identify risks. These recurrent constrains can be regarded as common knowledge about how to perform a feasibility analysis, and

are modeled as a logical entity named as pattern by our method.

According to the above analysis, patterns are in a higher level of abstraction compared with process components. A Pattern is modeled as a common solution of a given process concern or problem, instead of the solution for one specific project. Because a pattern is regarded as a set of constraints, and each constraint can be defined as a demanding relationship between activities across different sub-processes, it is very suitable to support cross-cutting concerns. As to the reuse approach, instead of assembling several modular building blocks into a bigger one, our approach weaving process patterns into existing process models. A pattern can take different forms after weaving into different models, because it is adapted to the project specific characteristics of different models. By separate project specific characteristics from the common knowledge of a solution, process patterns can be more reusable than fragments of any existing process models.

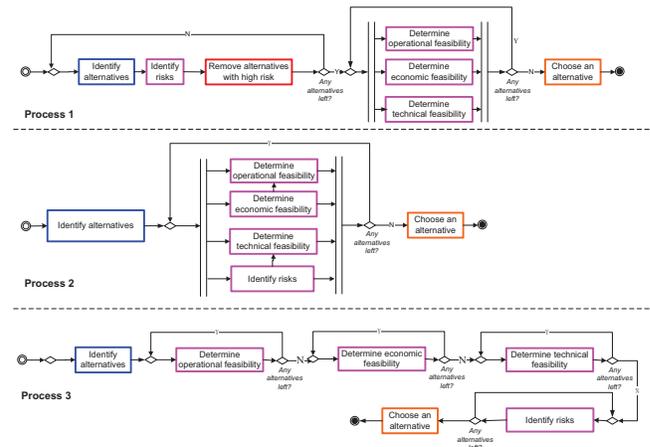


Figure 1. Some constrains are kept unchanged in three different process models

### III. PROCESS PATTERN META-MODEL

In our meta-model for process pattern (figure 2), a pattern consists of an ExecutableNode set to represent the essential activities and their temporal/logical relationships.

Activity is the work definition performed by certain Role which uses and produces some WorkProduct. A StructuredActivity groups its subordinate nodes by certain relationship, and each relationship can be regarded as a kind of constraint which should be satisfied by the resulting process model after weaving the pattern into it. The subordinate nodes must belong to only one StructuredActivity, although they may be nested. Classified by the collaborative ways of subordinate nodes, there are five kinds of StructuredActivity:

*Sequence:* Subordinate nodes are executed sequentially.

*Parallel:* Subordinate nodes are executed concurrently.

*Choice*: Exclusive choice among several alternatives.

*Cycle*: Subordinate nodes are executed sequentially and repeatedly until the stopping condition is satisfied.

*And*: Subordinate nodes should all exist and they should have the same predecessor and successor. The And relation defines a loose temporal relationship among the activities. After weaving the pattern into some existing process model, the And relations will be replaced by other determinate relations (i.e. Sequence, Parallel, or Cycle).

The ExecutableNode set is presented by a tree structure named Process Pattern Structure Tree (PPST). Each leaf node in PPST is an Activity. Every sub-tree with more than one node is a StructureActivity defining how its subordinate nodes are coordinated. The root node of the sub-tree is decided by the StructuredActivity type (Sequence, Parallel, Choice, Cycle or And which is represented with the symbol of  $\ominus$ ,  $\textcircled{/}$ ,  $\otimes$ ,  $\odot$  or  $\wedge$  accordingly). Every Activity belongs to one and only one SubstructureActivity.

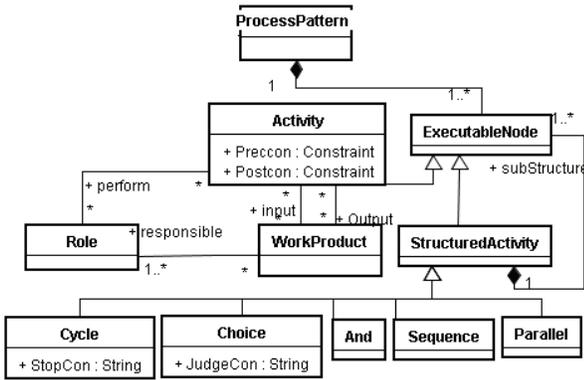


Figure 2. Meta-model for Process Pattern

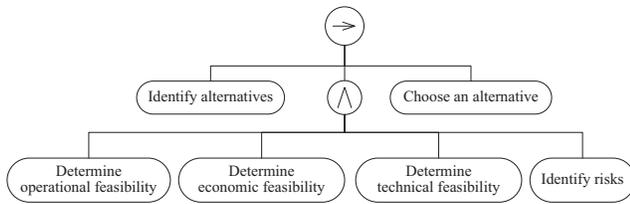


Figure 3. PPST of Pattern "Feasibility Analysis"

Figure 3 is an example of the PPST of Pattern "Feasibility Analysis" which is depicted in figure 1.

#### IV. PROCESS PATTERN REUSE APPROACH

In order to reuse a process pattern into different process models, we provide an approach to weave its constraints into existing process model automatically. Here, the existing process model is called source process while the process model output by our weaving method is called target process.

In our approach, three important principles should be followed:

*Principle 1*: All constraints in the pattern should be satisfied in target process.

*Principle 2*: No other constraints could be added into the target process.

*Principle 3*: Under the premise of meeting principle 1 and 2, the original constraints of the source process should be preserved.

#### A. Overview of the pattern weaving approach

*Phase 1*: Parse the source process to a Process Structure Tree (PST).

Here our approach is based on process parsing method [8, 9] to decompose a process model into a set of SESE (Single Entry Single Exit) fragments and then organize these SESE fragments into a Process Structure Tree (PST) which is similar to the PPST described in section 2.

*Phase 2*: Locate the activities that both exist in the pattern and the source process

A pattern may contain a set of activities, in this phase, our method search the source process and locates all those activities that both exist in the pattern and the source process, and mark up those activities that only exist in the pattern.

*Phase 3*: Change the source process to satisfy all constraints contained in the pattern.

Based on the result of phase 2, our method adds activities that only exist in the pattern into the source process; detects conflicts between constraints in the pattern and the source process; and changes the relations between activities in the source process to solve the conflicts. (The detail of change mode will be described in section 3.2.) When any change takes place in the source process, the 3 principles are carefully considered.

*Phase 4*: Simplify the PST.

In this phase our approach reformates the PST to eliminate redundant nodes and edges brought in phase 3. For example, if the node "parallel" has only one child node A, then the node "parallel" is removed from the PST and its father node is linked to A directly.

*Phase 5*: Restore the PST to the target process.

At last, our approach transfer the after weaving PST back to a workflow graph, and then output the target process.

#### V. CASE STUDY

Pattern "Feasibility Analysis" has already been described in Section 3. Now it is applied to two cases. In Case 1(Figure 4(b)), the alternatives are identified firstly, and then economic feasibility, technical feasibility and operational feasibility are determined parallel to choose an alternative. Case 2 (Figure 4(d)) is a little different from Case 1. After identifying alternatives, risks of each alternative are evaluated and those with high risks are discarded. If there is still any alternative left, make a choice after assessing the

economic and technical feasibility. Compared with the pattern, Case 1 does not have the activity of identifying risks while Case 2 lacks the activity of identifying operational feasibility. The resulting processes of Case 1 and 2 are shown in Figure 4(c) and Figure 4(e) respectively, and they demonstrate totally different shapes. The activity “Identify risks” is parallel to “Determine economic feasibility” in Case 1 but before “Determine economic feasibility” in Case 2. Moreover, Case 2 keeps the activity “Delete alternatives with risks” and its logic relationships which are not occurred in the pattern.

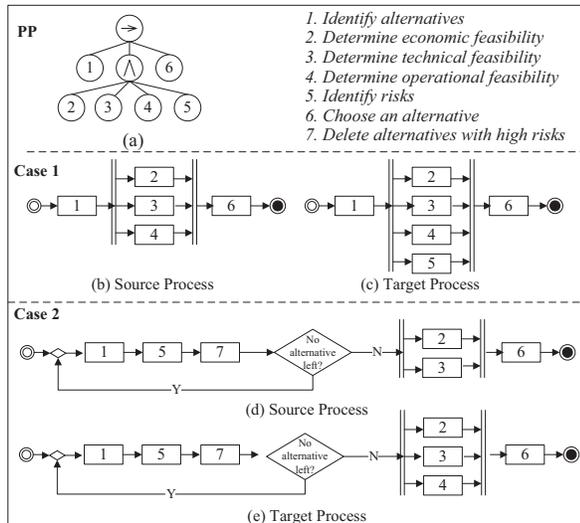


Figure 4. Example of “Feasibility Analysis”

From the example, it is can be seen that the pattern described by our approach is capable of providing high reusability and feasibility. Since the pattern only defines the essential activities and relationships, the applied process can keep itself activities and relationships as long as it does not violate pattern’s constraints.

## VI. RELATED WORK

Recently, the pattern approach has been adopted by process communities to capture and reuse process knowledge for recurrent development problems. Although process pattern approach is very promising, this concept still remains difficult to be exploited in practice due to the lack of formalization and supporting methodology [7, 10].

Some research regards process pattern as a black box which may composited together to construct a process, such as [7, 11]. In this way, their process patterns are similar to process components. Tran, H.N etc. [12] broaden the use of process patterns and begin to concern with the pattern's effect on process structure, but their approach can only be done manually. Förster [13] introduces a visual pattern specification language based on UML Activity Diagrams that enables the expression of quality constraints as patterns. He could represent the variabilities in the pattern, but he still does not solve the problem of merging pattern with existing process either. The problem of parsing workflow graph is

deeply discussed in [8, 9]. Parsing workflow graph helps speed up process analysis and comparison. A parsing algorithm is proposed in [9] to ensure obtaining a unique and modular parsing, which lays the foundation of our work.

## VII. CONCLUSIONS AND FUTURE WORK

Pattern based process reuse is a promising method. In our method, a process pattern can be considered as a set of constraints, and in this way the reusability of pattern is better than modular process components. And, our method utilizes weaving instead of assembling to reuse patterns into different process models, and in this way to support ubiquitous cross cutting concerns of software process. We are now developing the supporting tool for pattern representation and automatic weaving.

## ACKNOWLEDGEMENT

This work is supported by the High-Tech Research and Development Program of China under Grant No. 2009AA010307, the Science Fund for Creative Research Groups of China under Grant No. 60821003, the National Natural Science Foundation of China under Grant No. 60803011, 60803010.

## REFERENCE

- [1] S.W. Ambler, *Process Patterns: Building Large-Scale Systems Using Object Technology*. 1998: Cambridge University Press.
- [2] J.O. Coplien, *A Generative Development - Process Pattern Language*, in *The Patterns Handbook: Techniques, Strategies, and Applications*. 1996, Cambridge University Press. p. 243-300.
- [3] L. Osterweil, *Software Processes Are Software Too*. in *Proceedings of the 9th international conference on Software Engineering*, 1987: p. 2-13.
- [4] K. A. Gary, T. E. Lindquist, *Cooperating process components*, *Proceedings - IEEE Computer Society's International Computer Software and Applications Conference*, 1999: p. 218-223.
- [5] Hollenbach, C., W. Frakes. *Software process reuse in an industrial setting*. in *Software Reuse. Proceedings Fourth International Conference*. 1996. 22-30.
- [6] D. Rombach, *Integrated software process and product lines*, 2006. LNCS 3840:p. 83-90.
- [7] M. Hagen, V. Gruhn. *Towards flexible software processes by using process patterns*. in *Software Engineering and Applications*. 2004.p. 436-441
- [8] J. Vanhatalo, H. Völzer, F. Leymann. *Faster and more focused control-flow analysis for business process models though SESE decomposition*. in *ICSOC 2007*. 2007.p. 43-55
- [9] J. Vanhatalo, H. Völze, J. Koehler. *The Refined Process Structure Tree*. in *BPM 2008*. 2008: Springer-Verlag Berlin Heidelberg
- [10] H.N. Tran., B. Coulette, B.T. Dong. *A UML-Based Process Meta-model Integrating a Rigorous Process Patterns Definition*. in *PROFES 2006*. 2006.429-434
- [11] H. Störrle, *Describing Process Patterns with UML Software Process Technology*, 2001. LNCS 2077:173-181.
- [12] H.N. Tran, B. Coulette, B.T. Dong. *Modeling Process Patterns and Their Application*. in *International Conference on Software Engineering Advances*. 2007
- [13] A. Förster, G. Engels, T. Schattkowsky, et al. *A Pattern-driven Development Process for Quality Standard-conforming Business Process Models*. in *VL/HCC 2006*. 2006.135 – 142

# A Cross-Layer Design for Adaptive Multimodal Interfaces in Pervasive Computing

Jun Kong      Weiyi Zhang      Juan Li      Arjun G. Roy  
North Dakota State University  
{jun.kong, weiyi.zhang, j.li, arjun.roy}@ndsu.edu

## ABSTRACT

*Multimodal interfaces have attracted more and more attentions. Most researches focus on each communication mode independently and then fuse the information at the application level. Recently, several frameworks and models have been proposed to support the design and development of multimodal interfaces. However, it is still a challenging issue of supporting adaptations in the multimodal interfaces. Existing approaches are using rule-based specifications to define the adaptation of input/output modalities. Distinct from previous work, this paper presents a novel approach, which quantifies the usability of each modality and then formalizes the adaptation issue as searching for a set of input/output modalities that produce the highest usability. Furthermore, our approach supports a cross-layer design, which considers the adaptation from the perspectives of the interaction context, available system resources and QoS requirements. In other words, our design crosses application, system and network layers. An optimal solution and a heuristic algorithm are developed to automatically select an appropriate set of modality combinations under a certain situation. The numerical evaluation shows good performance of both solutions.*

## 1. Introduction

Multimodal interfaces process two or more combined user input modes (such as speech, pen, gaze, or manual gestures) in a coordinated manner with multimedia system outputs [13]. One main challenge in multimodal interaction design is to automatically select an optimal set of modality combinations that users will find easy and intuitive to produce and that the system will be able to interpret [4]. In a pervasive environment, modality combinations should further adapt to different interaction contexts, such as a different physical environment, due to the user mobility. Few studies have been conducted on adapting a multimodal interface to different interaction contexts. The frameworks of FAME [6] and MOSTe [16] are valuable for developing adaptive multimodal applications. However, those approaches use rule-based specifications to define adaptation, which has the problems of completeness and coherence [16]. This paper proposes a novel approach, which considers adaptation from three layers: the interaction context in the application layer, the resources

allocation in the system layer and the QoS provisioning in the network layer.

In human computer interaction, computers and humans establish various communication channels, over which messages are exchanged with associated effects [11]. However, user perceived effects may be reduced by constraints on interaction platforms, physical environments and the static and dynamic features of a user. For example, visual effect may be limited by a small screen on a mobile device; a noisy environment can greatly reduce the usage of auditory effects; and blind users are absent of all visual stimulus processing. Furthermore, user's dynamic features, such as walking and driving, can also influence interaction effects. For example, a fast movement (i.e. running) can affect a user reading information and selecting a menu in an interaction. Behavioral rules [6] have been commonly used to specify the adaptation, triggered by a change in the interaction context. Distinct from existing work, our approach quantifies the usability of each modality. Based on the quantification, the issue of modality adaptation is formalized as an optimization issue with the objective of maximizing the usability under an interaction context with certain constraints satisfied.

In addition to dynamic interaction contexts, handheld devices in a pervasive environment in general are constrained with limited computing capability, which raises an interesting topic of applying multimodal HCI to mobile devices that have limited input/output resources [7]. On a mobile device, different modalities compete for limited system resources, such as battery and CPU. Therefore, we must assure that the resources requested by selected modalities cannot exceed the total available resources.

Different modalities may have different QoS requirements. For example, an auditory message requires higher bandwidth for a short delay than a textual message. So, it requires a QoS-assured connection for one selected modality. QoS-assured routing is a fundamental issue in wireless and wired networks, and has been extensively investigated. This paper considers delay as the QoS parameter and uses the shortest path algorithm to search for a satisfied routing path. In the case of multiple QoS constraints, a randomized algorithm [9] can be applied to solve the multi-constraint QoS path problem.

In summary, this paper proposes a cross-layer design, which specifies adaptation from the application layer (i.e. interaction context of user, platform and environment), the system layer (i.e. resources) and the network layer (i.e. QoS

requirements). By quantifying each modality with a usability value, we formalize the adaptation issue as searching for input/output modalities with the objective of achieving *the highest usability* under constraints (1) do not exceed the maximum available system resources and (2) QoS requirements are assured for selected modalities. This paper first proposes an optimal solution based on integer linear programming. Due to the high complexity of the optimal solution in a large problem, this paper also proposes an efficient heuristic algorithm to solve the adaptation issue based on the classic 0-1 knapsack problem.

## 2. A Cross-Layer Design for the Development of Multimodal Interfaces

Figure 1 shows a cross-layer design for adaptive multimodal interfaces. Under a certain interaction context, a mapping from a modality space to a usability space defines an *interaction context profile*, which indicates the usability of a modality under a given interaction context; a mapping from a modality space to QoS preferences defines a *QoS profile*, which indicates the QoS requirements of a modality; and a mapping from a modality space to system requirements defines a *resource profile*, which indicates the resource requirements of executing a modality. Based on the user profile, the QoS profile and the resource profile, an optimal solution and a heuristic algorithm are designed to determine input/output modalities, which achieve the highest usability under a given interaction context with satisfied system requirements and a QoS-assured connection.

### 2.1. Interaction Context

Interaction context describes the state under which a person uses a device [1]. Getting inspired by previous researches [15, 17, 18], we consider three main entities in the interaction context: *User*, *Device*, and *Environment*.

**User** plays a central role in the human computer interaction. The delivery and rendering of information to a user must fit user's personalized features, such as the user's preferences, motion, physiological and mental states, mood, current activity and etc.

The **environment** in which the user interacts with the device is another important entity in the human computer interaction. It continuously affects the interaction of a communication mode. For example, a high noise level could significantly reduce an auditory effect.

The **device** entity determines the input and output capacities, i.e. a *modality space* that defines all available communication modes supported by the device. Furthermore, the characteristics of a device may affect the usability of a communication mode.

### 2.2. Modality Space

A device in a pervasive environment determines available input and output modalities, which construct a modality space. Under a given interaction context, a subset of appropriate modalities should be selected from the modality space. Since

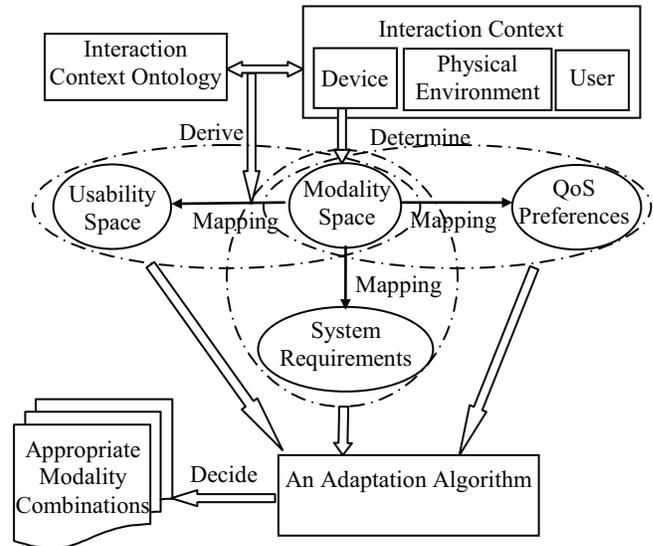


Figure 1. Design of Adaptive multimodal interfaces

user mobility may change the infrastructure of a pervasive environment, a modality space needs to be updated accordingly.

The modalities in a modality space have two fundamental relations, i.e. redundancy and complementarity [12]:

- **Redundancy:** If two communication modes deliver the same amount of information, they have the redundancy relationship. For example, during computer-mediated, distance-learning lectures, 100% of the presenter's handwriting was accompanied by semantically redundant speech [2, 8].

- **Complementary:** The presentation and interpretation of information depend on two or more modalities. For example, speech and pen inputs consistently contribute different and complementary semantic information [12].

In a multimodal interface, the modalities with a redundancy relation can be selected independently. However, modalities with a complementarity relation must be selected at the same time, and they are considered as a unified composite modality in the process of modality adaptation. The execution of a modality needs to consume system resources. The consumption of system resources implicitly reflects the competition among modalities. Therefore, the modality adaptation must consider the availability of system resources.

### 2.3. Usability space

Some researchers have investigated the suitability of a modality under different interaction contexts [10, 11]. Based on previous work, the usability of a modality can be quantified as a value between 0 and 1. Such a usability value represents a user's personal satisfaction/preference on a modality. More specifically, a modality with value 1 indicates that a user can easily interact with a system through this modality. On the other hand, a modality with value 0 means that this modality is not usable for a user. For example, the output modality of photograph has a "0" usability for a blind user. The usability value of a modality is dynamic and may change according to

an interaction context. For example, when a user is moving from his/her office to a vehicle and starts driving the vehicle, the usability value associated with a visual display will be significantly reduced. Under a certain interaction context, a mapping from a modality space to a usability space defines an interaction context profile, which represents the effect of a modality.

## 2.4. QoS Preferences

Different types of information representation may have different QoS requirements. For example, in order to support a smooth playback, speech-based information requires a higher bandwidth to have a short delay on a network than text-based information. To the best of our knowledge, most existing approaches do not consider QoS requirements. This paper takes *delay* as the primary QoS constraint, but our work can be easily extended to multiple QoS constraints.

## 2.5. System Requirements

The selection of a modality triggers the execution of a corresponding software application, which consumes system resources. For example, a speech-based interaction requires relevant voice recognition/synthesis software. Though the computing capability of a mobile device becomes more and more powerful, it is still constrained with physical limitations. Therefore, the modality adaption needs to consider the system requirements of each modality and assures that the requested resources do not exceed the maximum available resources. This paper considers three different types of system resources: CPU, memory and bandwidth while our work can easily extend to other resources.

## 3. An Optimal Solution

This section gives an optimal solution to the modality adaptation problem based on integer linear programming (ILP). The following notations have been used in the problem formulation:

- *MI*: The set of input modalities.
- *MO*: The set of output modalities.
- *MDS*: The set of available modalities, i.e. the *modality space*, and  $MI \cup MO = MDS$ .
- *BAND*: The total available system bandwidth of a pervasive computing system.
- *CPU*: The total available CPU capacity of the system.
- *MEM*: The total available memory of the system.
- *i*: Representing an input modality in the set of *MI*.
- *o*: Representing an output modality in the set of *MO*.
- *m*: Representing a modality in the set of *MDS*.
- *Usability(m)*: The usability value for the modality *m*.
- *Band(m)*: The bandwidth required by the modality *m*.
- *Cpu(m)*: The CPU required by the modality *m*.
- *Mem(m)*: The memory required by modality *m*.

- *Delay(m)*: *delay* requirement of the modality *m*.
- *Mod(m)*: Indicating if a modality *m* is selected. The value 1 means that the modality is chosen. Otherwise the value is 0.
- *delay(i, j)*: The delay of link (*i, j*) of the network.
- *Cap(i, j)*: The capacity of link (*i, j*) of the network.
- *Adj(v)*: The adjacent nodes of node *v* in the network.

As defined in formula (1), the objective of modality adaptation is to *find a set of modalities, which achieve the maximum usability*. Furthermore, the selected modalities should include at least one input modality and one output modality, as defined in formulas (2) and (3). Formulas (4) to (6) define the constraints of system resources: requested resources of selected modalities should not exceed corresponding system resource capacities.

$$(1) \text{Maximize} \left\{ \sum_{i \in MI} \text{Mod}(i) * \text{Usability}(i) + \sum_{o \in MO} \text{Mod}(o) * \text{Usability}(o) \right\}$$

$$(2) \sum_{i \in MI} \text{Mod}(i) \geq 1$$

$$(3) \sum_{o \in MO} \text{Mod}(o) \geq 1$$

$$(4) \left( \sum_{i \in MI} (\text{Mod}(i) * \text{Band}(i)) + \sum_{o \in MO} (\text{Mod}(o) * \text{Band}(o)) \right) \leq \text{BAND}$$

$$(5) \left( \sum_{i \in MI} (\text{Mod}(i) * \text{CPU}(i)) + \sum_{o \in MO} (\text{Mod}(o) * \text{Cpu}(o)) \right) \leq \text{CPU}$$

$$(6) \left( \sum_{i \in MI} (\text{Mod}(i) * \text{Mem}(i)) + \sum_{o \in MO} (\text{Mod}(o) * \text{Mem}(o)) \right) \leq \text{MEM}$$

The corresponding software component of a modality has its delay constraint. In the following formulations from (7) to (11),  $s_m$  indicates the information receiver, in general the mobile user; and  $t_m$  represents the service provider. Formulas (7), (8), and (9) are used to find a path for a modality in the network. More specifically,  $f_{(i,j)}^m$  indicates whether link (*i, j*) is used for routing modality *m*. The value 1 means that link (*i, j*) is on a path for modality *m*. Otherwise the value is 0. Formula (10) assures that the found path for modality *m* must satisfy the delay requirements of *m*. Formula (11) defines that each link in the network could be used for the transmissions of multiple applications, but the total traffic on the link cannot exceed its capacity.

$$(7) \sum_{i \in \text{Adj}(s_m)} f_{(s_m, i)}^m - \sum_{i \in \text{Adj}(s_m)} f_{(i, s_m)}^m = \text{Mod}(m), \forall m \in \text{MDS}$$

$$(8) \sum_{i \in \text{Adj}(t_m)} f_{(i, t_m)}^m - \sum_{i \in \text{Adj}(t_m)} f_{(t_m, i)}^m = \text{Mod}(m), \forall m \in \text{MDS}$$

$$(9) \sum_{i \in \text{Adj}(j)} f_{(j, i)}^m = \sum_{i \in \text{Adj}(j)} f_{(i, j)}^m, \forall m \in \text{MDS}, \forall i \in V \setminus \{s, t\}$$

$$(10) \sum_{(i,j) \in E} (\text{delay}((i, j)) * f_{(i,j)}^m) \leq \text{Mod}(m) * \text{Delay}(m), \forall m \in \text{MDS}$$

$$(11) \sum_{m \in \text{MDS}} (\text{Band}(m) * f_{(i,j)}^m) \leq \text{Cap}(i, j), \forall (i, j) \in E$$

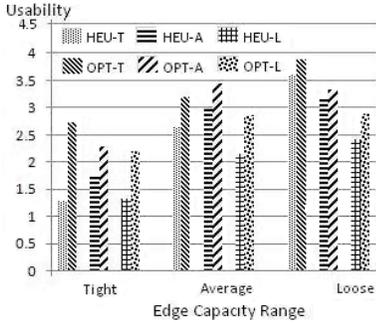


Figure 2. Usability on a random network with 2 input modalities and 8 output modalities

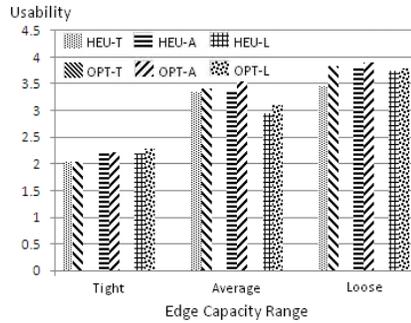


Figure 3. Usability on a random network with 4 input modalities and 6 output modalities

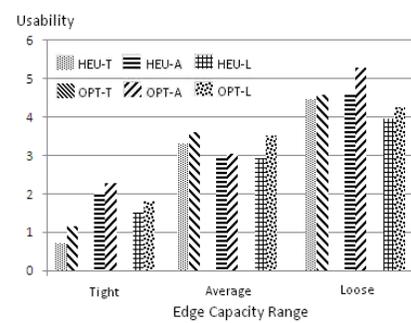


Figure 4. Usability on a random network with 5 input modalities and 5 output modalities

The above linear programming formulas provide an optimal solution. When the network and the modality size are relatively small, it can return an optimal solution. However, when the problem input is large, solving this integer linear program could be time consuming. Next section introduces an efficient heuristic algorithm for the problem with a larger size.

#### 4. A Heuristic Algorithm

Briefly speaking, an efficient heuristic approach is developed to solve the modality adaptation in three steps:

- Search for a set of input and output modalities, which can achieve the maximum usability and do not exceed the system resource capacities. This step is essentially a 0-1 knapsack problem. After this step, all modalities are classified as selected and unselected.
- Then, we calculate the shortest path one by one for each modality  $m$  selected in the first step. If a path with satisfied delay constraint for modality  $m$  exists,  $m$  is finally selected. Otherwise,  $m$  is removed from the selected set.
- Starting from the modality with the highest usability in the *unselected* set, if a path with a satisfied delay constraint for modality  $m$  exists,  $m$  is finally selected. This process continues until every modality in the unselected set is checked or no available resource left.

##### 4.1. Numerical Evaluation

We implemented our heuristic algorithm (denoted by **HEU** in the figures), and compared it with the optimal solution (denoted by **OPT** in the figures) given by the integer linear programming (ILP) formulation in Section 3. As in [20], we used both well-known Internet topologies and randomly generated topology to study the suitability and computational time complexity of the algorithms. All tests were performed on a 2.4GHz Linux PC with 2G bytes of memory. Due to limited space, we only present the evaluation results on a randomly generated topology while the well-known topologies have the similar trend as the random ones.

For each input or output modality, its usability was uniformly generated in a given range  $[0,1]$ . And its bandwidth, CPU and memory resource requirements were uniformly

generated in a given range  $[1, R]$ . We consider different sets of modality resource requirements, **Tight** bound ( $R = 30$ ), **Average** bound ( $R = 50$ ), and **Loose** bound ( $R = 70$ ).

In our simulation, different modality sets were generated. The size of the input/output modality sets are given as: *different* (2 input modality/8 output modality), *equal* (5 input modality/5 output modality) and *close* (4 input modality/6 output modality).

First we compare our solutions on a random generated network (50 nodes, 300 edges). Meanwhile, on each link in the network, as in [9, 20], the delay of each link was also randomly generated in a given range (we used the range  $[1,10]$ ). For each test case, we considered three scenarios according to the range of the capacity of the links: tight case (given range  $(0,10]$ ), average case (given range  $(0,20]$ ) and loose case (range  $(0,100]$ ). In Figure 2, we show the usability performance on the random network with *different* modalities (2 input modalities and 8 output modalities). As expected, **OPT** always performed better than **HEU** did regardless of modality resource requirements (**T**, **A** and **L** represents *Tight* bound, *Average* bound, and *Loose* bound, respectively). Meanwhile, we can see that **HEU** delivered very good performance because it provided near-optimal solutions in most cases. Similar observations can be made for the *close* set of modalities in Figure 3, and for the *equal* set of modalities in Figure 4.

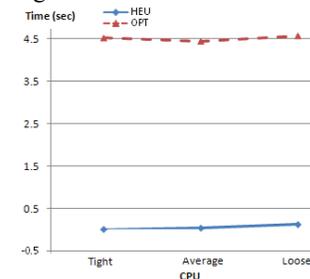


Figure 5. Running time on a random network with 2 input modalities and 8 output modalities

Though **OPT** has better usability performance than **HEU**, it takes much longer time to get a solution for each testing case. As shown in Figure 5, it took at least 10 times longer time for **OPT** to find a solution than **HEU** did. For example, in Figure 5, when modality resource has a *loose* bound, **HEU** spent 0.13 seconds to find a solution with usability 3.6. **OPT** took 4.5 seconds to

find an optimal solution with usability 3.9.

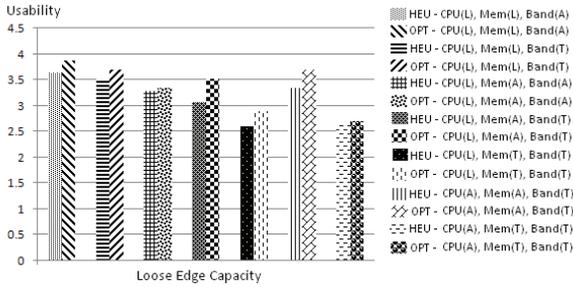


Figure 6. Usability on a random network with different system resources

In all above experiments, CPU, memory and bandwidth requirements were using the same bound. In the following, we check the performances for our solutions in general scenarios, in which some resource requirements are more tight or loose than others. For example, CPU is in the *Tight* bound while memory could be in the *Loose* bound. In Figure 6, we tested all the possible cases on a random network topology. We have the similar results regardless the network edge capacity range. Therefore, we only show the *Loose* edge capacity scenario on the random network. We observed that HEU performed well by delivering near-optimal solutions.

Based on our comprehensive numerical evaluation, OPT and HEU both delivered good performances in terms of usability and running time.

## 5. A Case Study

In this case study, we consider a pervasive application, which supports a salesman to access information of products anytime and anywhere. This application includes a mobile device, e.g. a blackberry smart phone, which a salesman can take with at various customer locations. This mobile device determines the modality space which includes four input modalities and five output modalities. The input modalities are listed as: textual input is supported through a small keypad or a Bluetooth-enabled portable keyboard; speech-based input is enabled through a microphone; users can use a pen/stylus to perform pen-based interaction; and a Bluetooth mouse is supported to make a selection. The output modalities include speech, audio files, video files, graphical representations and textual information. The application of a modality needs the support of some special software and consumes system resources. For example, the speech-based input needs the voice recognition software of *IBM ViaVoice for Embedded devices*, which requires 700KB RAM. The system requirement of each modality defines the resource profile. Furthermore, some modality has its QoS requirements. For example, if an audio file is transferred from a server located at the headquarter to the salesman's mobile device, the delay should not exceed certain seconds in order to provide a smooth playback. Accordingly, those QoS requirements for each modality represent the QoS profile.

In addition to QoS and resource profiles, the interaction context, which is classified into three categories (i.e. user, environment and device), also affects the selection of

modalities. With the above mobile application, we can list five common scenarios: (1) when a user is driving a vehicle, his/her hands are busy; (2) a user may need to access information during a meeting, which indicates a noisy environment; (3) a user accesses sensitive information in some hostile environment; (4) a user retrieves information in an airport, in which there is only limited wireless connection; (5) user's device has a small screen. From those common scenarios, we can summarize a list of features that affect the usability of modalities. For example, the user's dynamic state (such as driving, sitting or walking) can affect the usability of vision-based modalities; the noise level in a physical environment can affect the audio-based modalities; and the screen size of a mobile device can affect the usage of large pictures. An interaction context profile is defined by identifying those interaction-context features and the effects of those features on the usability of each modality. Based on the above profiles, an adaptation engine can automatically choose the most appropriate modalities under a specific scenario.

## 6. Related Work

By providing a natural communication between the user and the computer, multimodal human-computer interaction has been growing fast. A majority of current approaches target to address issues in specific modalities [6]. For example, Jaims and Sebe gave an extensive survey for multimodal human computer interaction from the perspective of computer vision [7]. Instead, this paper focuses on how to automatically select appropriate modalities based on human emotion and other features.

Several studies have been conducted to develop context-aware mobile applications [18, 19], which can adapt to dynamically changing environmental and physiological states. The above approaches focus on context recognition, instead of modality adaptation.

An important requirement for context-aware applications is the ability to adapt at run time. Several researchers [5, 14] have worked on dynamically allocating system resources according to available resources and user preferences. Those approaches focus on reconfiguration of system software, instead of modalities.

Lemmela *et. al.* [10] proposed a 5-step iterative process, which includes observing and analyzing interaction contexts, to design multimodal user interfaces. Based on the characteristics of different modalities, the designers can determine suitable modalities for different situations. Bouchet *et. al.* [3] proposed a component-based approach for rapid development of multimodal interfaces. This approach includes two types of software components, i.e. elementary components for pure modalities and composite components for combining modalities. Those frameworks and models are valuable to design and develop multimodal interfaces, but they do not support automatic adaptation.

Rousseau *et. al.* [16] developed a Multimodal Output Specification Tool, called *MOSTe*, which specifies multimodal output in terms of interaction components, interaction context and information units. A behavioral model based on election rules is used to define the adaptation upon different situations.

Duarte and Carrico [6] proposed a conceptual framework, i.e. *FAME*, for the development of an adaptive multimodal system. The *FAME* architecture uses different models to specify the features of a multimodal application from the perspectives of user, platform and environment. An innovative behavioral matrix is introduced to represent adaptation rules. Though rule-based specifications propose a simple reasoning with a low learning cost, they have the problems of the completeness and the coherence of the rules base [16]. Distinct from the above approaches, this paper formalizes the adaptation as searching for an optimal set of modalities with the highest usability under a specific scenario. Furthermore, our work considers adaptation not only from the perspective of interaction context (i.e. user, platform and environment), but also from the aspects of system resources constraints and underlying QoS-assured routing.

## 7. Conclusion and Future Work

It stills leaves as an open problem of addressing the modality adaptation in the process of multimodal interface design. Recently, several frameworks and models have been proposed to support adaptation through rule-based specifications. Rule-based adaptation may not be complete to different scenarios and different rules could conflict with each other. This paper presents a novel approach based on integer linear programming. Different from previous work, this paper formalizes the adaptation issue as searching for a set of input/output modalities that produce the highest usability with satisfied resource and QoS requirements. Given a specific interaction scenario, both an optimal solution and a heuristic algorithm are developed to automatically select an appropriate set of modalities combinations. The numerical evaluation shows good performance of the proposed solutions. In the future, we will consider combining modality adaptation with content adaptation.

## 8. Acknowledgement

This work is supported in part by ND EPSCoR State Seed Grants #FAR0015845 and #FAR0015846, and HP Labs Innovation Research Program (2009-1047-1-A).

## 9. Reference

- [1] G. D. Abowd, A. K. Dey, R. Orr, J. Brotherton, J., Context-awareness in wearable and ubiquitous computing, *Virtual Reality*,3,(3), pp. 200 – 211, 1998
- [2] R. J. Anderson, C. Hoyer, S. A. Wolfman, and R. Anderson, "A Study of Digital Ink in Lecture Presentation", *Proc. CHI'04*, pp. 567-574, 2004.
- [3] J. Bouchet, L. Nigay, and T. Ganille, "Icare Software Components for Rapidly Developing Multimodal Interfaces", *Proc. ICMI'04*, pp. 251-258, 2004.
- [4] M. L. Bourguet, "Automatic Generation of Multimodal Interaction Models from Behavioral Data", *Proc. 2006 Workshop on Computer Assisted Recording, Pre-Processing, and Analysis of User Interaction Data*, 2006.
- [5] L. Capra, W. Emmerich and C. Mascolo, "CARISMA: Context-Aware Reflective mIddleware Systems for Mobile Applications", *IEEE Transactions on Software Engineering*, Vol. 29(10), pp.929-945, 2003.
- [6] C. Duarte and L. Carrico, "A Conceptual Framework for Developing Adaptive Multimodal Applications", *Proc. IUI'06*, pp.132-139, 2006.
- [7] A. Jaimes and N. Sebe, "Multimodal Human Computer Interaction: A Survey", *Proc. IEEE International Workshop on Human Computer Interaction*, 2005.
- [8] E. Kaiser, P. Barthelmess, C. Erdmann, and P. Cohen, "Multimodal Redundancy Across Handwriting and Speech During Computer Mediated Human-Human Interactions", *Proc. CHI'07*, pp.1009-1018, 2007.
- [9] T. Korkmaz and M. Krunz, "A Randomized Algorithm for Finding a Path Subject to Multiple QoS Requirements", *Computer Networks*, Vol. 36, pp. 251-268, 2001.
- [10] S. Lemmela, A. Vetek, K. Makela, D. Trendafilov, "Designing and Evaluating Multimodal Interaction for Mobile Contexts", *Proc. ICMI'08*, pp. 265-272, 2008.
- [11] Z. Obrenovic, J. Abascal, and D. Starcevic, "Universal Accessibility as A multimodal Design Issue", *Communication of the ACM*, Vol.50(5), pp.83-88, 2007.
- [12] S. Oviatt, "Ten Myths of Multimodal Interaction", *Communications of the ACM*, Vol. 42(11), pp. 74-81, 1999.
- [13] S. Oviatt, "Breaking the Robustness Barrier: Recent Progress on the Design of Robust Multimodal Systems", *Advances in Computers*, Vol.56, pp.305-341, 2002.
- [14] V. Poladian, J. P. Sousa, D. Garlan, and M. Shaw, "Dynamic Configuration of Resource-Aware Services", *Proc. the 26<sup>th</sup> International Conference on Software Engineering*, 2004.
- [15] D. Preuveneers, J. Van den Bergh, D. Wagelaar, A. Georges, P. Rigole, T. Clerckx, Y. Berbers, K. Coninx, V. Jonckers and K. De Bosschere, "Towards an extensible context ontology for ambient intelligence", *Ambient Intelligence*, pp.148-159, 2004.
- [16] C. Rousseau, Y. Bellik, and F. Vernier, "Multimodal Output Specification / Simulation Platform", *Proc. ICMI'05*, pp.84-91, 2005.
- [17] B. N. Schilit, N. I. Adams, R. Want, "Context-Aware Computing Applications", *Proc. Of the Workshop on Mobile Computing Systems and applications*, 1994.
- [18] A. Schmidt, K. A. Aidoo, A. Takaluoma, U. Tuomela, K. V. Laerhoven, W. V. de Velde, "Advanced Interaction in Context", *Proc. Handheld and Ubiquitous Computing*, pp.89-101, 1999.
- [19] D. Siewiorek, A. Smailagic, J. Furukawa, A. Krause, N. Moraveji, K. Reiger, and J. Shaffer, "SenSay: A Context-Aware Mobile Phone", *Proc. 7<sup>th</sup> IEEE International Symposium on Wearable Computers*, 2003.
- [20] G. L. Xue, A. Sen, W. Y. Zhang, J. Tang, and K. Thulasiraman, "Finding a Path Subject to Many Additive QoS Constraints", *IEEE/ACM Transactions on Networking*, Vol. 15, pp.201-211, 2007.

# Characteristics of Ubiquitous Software Projects: Pertinence, Relevance, and Use

Rodrigo Oliveira Spínola and Guilherme Horta Travassos  
PESC-COPPE/UFRJ  
68511, Rio de Janeiro, Brazil, +55 (21) 2562 8712  
{ros,ght}@cos.ufrj.br

## Abstract

*Ubiquitous software projects differ from traditional ones. The challenges introduced by the ubiquity characteristics have driven us to think on how our understanding about ubiquity software projects could be improved to support the planning, requirement analysis, designing, and coding of this new software category. Therefore, we decided to follow an evidence based research strategy to identify, organize and evaluate knowledge regarding this type of project. By combining evidences from the technical literature and experts from the field, some ubiquitous software characteristics and factors were identified. In complement, this paper also presents the ubiquity characteristics pertinence and relevance levels, and a brief description of a software technology to support requirements definition in ubicomp domain that has been developed based on these ubiquity characteristics. We believe these results can be useful to motivate new researches regarding software engineering applied to ubiquitous software projects.*

## 1. Introduction

In 1991, Mark Weiser published a seminal paper outlining his vision of the next computing generation. He referred to this as Ubiquitous Computing, or UbiComp, representing a new paradigm in which information processing is thoroughly integrated into the everyday objects and activities [5]. This way, computing facilities should spread across the user's environment becoming common place and easily used.

Currently, ubiquitous software represents a new software application category. Its development involves additional characteristics, such as context sensitivity, user experience capture, omnipresence, alternative user interfaces and so on [4], which are usually not addressed in traditional software projects.

From the software engineering's point of view, this development scenario can bring new challenges in tailoring or building software processes, impacting current methods, techniques, architectural styles, requirements gathering and verification [1, 2, 3].

Thus, we believe that it could be interesting to address some investigation aiming at identifying what the ubiquitous software characteristics are, and to organize a body of knowledge to support the development of this software category. This is an important step towards understanding how the ubiquity domain can impact the software development life cycle.

Therefore, we have decided to follow an evidence based research strategy to support our ubiquitous applications development research. The ubicomp body of knowledge organization followed the scientific approach represented in Figure 1 [10], available in the eSEE knowledge repository [16]. It makes use of systematic reviews [9] (secondary studies) and surveys (primary studies) to acquire knowledge from the field.

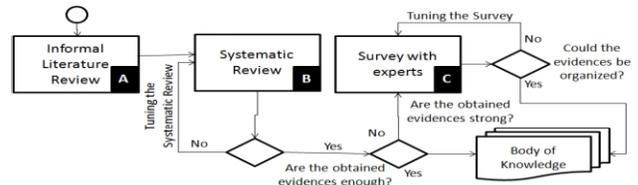


Figure 1. Research Steps.

This research strategy follows Mary Shaw's suggestions [7] regarding what makes good research in software engineering: (1) Defining a research question (the ubicomp field characterization); (2) Identifying the correct research results (the set of ubicomp characteristics and their factors); (3) Validating the obtained results (surveying specialists).

Based on this context, this paper presents the organization and evaluation of ubicomp characteristics and their factors by using secondary and primary studies. Some results obtained from using this research strategy for the characterization of ubiquitous software projects are presented on Section 2. Sections 3 and 4 present the evaluation of the organized body of knowledge through survey. This evaluation allowed us to identify the pertinence and relevance levels of ubiquity characteristics. In Section 5, an example of software technology supporting the requirements specification of ubiquitous software projects based on this body of knowledge is described. Finally, Section 6 presents the final considerations and the next steps of this research.

## 2. Organizing Ubicomp Body of Knowledge

As presented in Figure 1, the first step was an informal literature review (step A). A search on ACM and IEEE digital libraries was performed. The results obtained at that moment were the theoretical foundation about ubicomp allowing the first systematic review planning [4].

Next (step B), the first systematic review goal was to characterize the ubicomp field. For this, it was defined the following research questions: **(i)** What is ubiquitous computing?; **(ii)** How ubiquitous computing is actually being presented?; **(iii)** What characteristics do define applications for ubiquitous computing? The study objective was to make a characterization of the ubicomp field. There is no comparison between intervention and alternatives [17] nor will meta-analysis be applied. Therefore, this secondary study, although systematic, can be considered a *quasi*-systematic review [16].

A research protocol was prepared to support this *quasi*-systematic review execution on the IEEE Explorer, ACM Digital Library, INSPEC, and EI COMPENDEX digital libraries. It resulted, based on pre-defined inclusion and exclusion criteria, in the selection of 41 papers. These papers allowed us to update the definition regarding ubiquitous computing and the identification of a set of characteristics that should be present in ubiquitous software projects.

Hence, from this 1st *quasi*-systematic review, we could observe that ubiquitous computing is present when computational services or facilities become available to the people in such a way that the computer is no longer visible nor needed to be used as an essential tool to their access. The services or facilities can materialize themselves at any time or place, transparently, through the use of common daily devices. To make it happen, it is necessary that systems of this application category take the following 10 characteristics into consideration [4]: **Service Omnipresence:** it allows users to move around with the sensation of carrying computing services with them; **Invisibility:** ability of being daily present using objects, weakening, from user's point of view, the sensation of explicit use of a computer and enhancing the perception that objects or devices can provide services or some kind of "intelligence"; **Context Sensitivity:** ubiquitous systems should have mechanisms to collect information from the environment where it is being used; **Adaptable Behavior:** ability of dynamically adapting itself according to the offered environment services, respecting its limitations; **Experience Capture:** ubiquitous systems should have mechanisms to support capturing and registering experiences for later use; **Service Discovery:** this characteristic states that ubiquitous systems should have mechanisms to support pro-active discovery of services, which should be according to the environment where it is being used, in

order to find new services or information to achieve some desired target; **Function Composition:** to be able to create a service required by the user based on available basic services; **Spontaneous Interoperability:** ability to change partners during its operation and according to its movement; **Heterogeneity of Devices:** provides application mobility among heterogeneous devices. Thus, the application could migrate among devices and adjust itself to each device; **Fault Tolerance:** ability to self adapt when facing environment's faults.

However, one can observe that the 10 identified ubicomp characteristics were described in high abstraction level, making their use hard when dealing with concrete software projects. Therefore, it was necessary to go further looking for more concrete factors associated with each of them. Therefore, a refinement in the first review protocol was needed considering the following research question: what are the functional and restrictive factors that characterize each ubiquitous characteristic?

To undertake the second *quasi*-systematic review, the IEEE Explorer and ACM Digital Library were used as paper sources. The description of the extended research protocol can be found in [4]. At the end, 59 scientific papers were selected. They allowed the identification of 168 ubiquity factors that were organized into factor groups according to their definition. These factors represent functionalities usually found on ubiquitous software projects. For instance, for the context sensitivity characteristic the following factors were identified: (i) To contextualize the obtained information; (ii) To store the captured information.

At that moment, it was important to analyze the captured concepts to decide if a new refinement on the review protocol could be necessary. For this, the set of identified characteristics and their factors was used to create a checklist to characterize 12 ubiquitous software projects described in the technical literature. The intention was to observe if the concepts were captured in an appropriate detail level and how the ubiquity characteristics have been captured in real ubiquitous software projects, helping in understanding how they could influence the software project and obtaining insights about what ubiquity characteristics have been usually explored in practice [6]. Here, it is important to mention that, based on the checklist use, the set of characteristics and factors seems to make sense, and; currently, some few characteristics are more explored on ubiquitous software projects while the use of other ones appear as isolated initiatives, even considering the analyzed projects represent examples along the years 2000 to 2007, where some technological evolution took place.

In order to have an initial evaluation from the point of view of other researchers about the organized body of knowledge, we considered to survey software engineering ubicomp domain experts. The survey planning and execution are described in the next section.

### 3. Initial Survey

The goal of this study was to **analyze** ubiquitous computing characteristics and their factors extracted from the technical literature, **with the purpose of** understanding, **with respect to** their applicability and scope, **from the point of view of** software engineering researchers working with research and development of ubiquitous software **into the context of** ubiquitous software projects.

During its planning, three pilot studies were executed with the purpose of evaluating the questionnaire. After that, a new version of the questionnaire was elaborated and distributed among the subjects. Brazilian researchers' population was considered. The choice of the subjects was based on searching the Brazilian National Council for Science and Technology Development Research Groups Search Directory considering research groups which carry on ubicomp research. About 60 ubicomp researchers have been identified and were invited and surveyed by e-mail. At the end, 10 subjects (17% of the invited researchers) answered the questionnaire.

The results allowed us to make some improvements on the initial ubiquitous characteristics set and their corresponding factors. Basically, the changes were:

(1) Inclusion of three additional characteristics: (i) **Privacy and Trust**: indicates the system's ability to withhold the operations performed by the user and ensure that operations are not circumvented; (ii) **Scalability**: is the property of a system which indicates its ability to either handle growing amounts of work in a graceful manner or to be readily enlarged; (iii) **Quality of Service**: is the ability to provide different priority to different applications, users, or data flows, or to guarantee a certain level of performance to functionality during its execution.

(2) Characteristics were reorganized considering two perspectives: functional and restrictive; and

(3) Exclusion of three factors.

Table 1 presents the ubicomp knowledge organized so far. It is important to note that the population size was considered small and not representative considering a global scenario in ubicomp. Thus, the result of the survey could not be used as an evaluation of the body of knowledge, although the results are important to its evolution. Therefore, a second survey was accomplished.

**Table 1. Initial set of Ubicomp Characteristics**

Characteristic	# Factors	Functional/ Restrictive
Service Omnipresence	10	F
Invisibility	10	F
Context Sensitivity	30	F
Adaptable Behavior	32	F
Experience Captures	7	F
Service Discovery	24	F
Function Composition	23	F
Spontaneous Interoperability	12	F
Heterogeneity of Devices	11	F

Fault Tolerance	6	R
Scalability	*	R
Quality of Service	*	R
Privacy and Trust	*	R

\* Not yet identified.

### 4. Evaluating the Ubicomp Body of Knowledge

This study's goal was to **analyze** ubiquity characteristics extracted from the technical literature **with the purpose of** characterization, **with respect to** their adequacy and relevance when characterizing ubiquitous software projects, **from the point of view of** researchers on the Ubicomp field, **in the context of** ubiquitous software projects.

Therefore, the following research questions were considered: (i) Are the characteristics extracted from the technical literature adequate (or not) to characterize ubiquitous software projects? (ii) Is there any additional characteristic to characterize a ubiquitous software project that could be considered? (iii) What is the importance (relevance level) of each characteristic when characterizing ubiquitous software projects?

In this study, adequacy indicates if each characteristic is or is not useful to describe or to define a body of knowledge regarding Ubicomp Characteristics. Relevance indicates how useful is it when characterizing a ubiquitous software project, that is, the weight of this characteristic on the ubiquitous software projects characterization.

#### 4.1. Instrumentation and Population Planning

As instrumentation, an online questionnaire has been developed and was published on Internet. It is filled in three steps: (i) Subject characterization (for instance: personal data, academic degree, and experience level on ubiquitous software projects); (ii) Identification of those characteristics adequate or not to characterize ubiquitous software projects; (iii) Definition of each characteristic's relevance level to support the characterization of ubiquitous software projects considering six relevance levels (Likert Scale): No Relevance, Very Low, Low, Medium, High, and Very High Relevance.

The population of this survey was represented by authors that published papers: (i) identified by two *quasi*-systematic reviews presented on section 2, or; (ii) in the proceedings of UBICOMP – one of the most important conferences in the area. These authors have been contacted by e-mail and they were able to access a website with the questionnaire.

We are assuming that this population can be representative in the context of Ubicomp researchers, and the subjects answered the questionnaire using their background and experience in this field.

Table 2. Pertinence and Relevance Level of Characteristics to Ubiquity Software Projects

Characteristic	Pertinence Level	Rank	Relevance Level	Rank	Functional/Restrictive
Context sensitivity	93,47%	1	84,81%	1	F
Adaptable behavior	89,71%	2	71,79%	2	F
Privacy and trust	76,21%	3	59,32%	3	R
Heterogeneity of devices	62,34%	4	44,56%	7	F
Experience capture	62,16%	5	46,69%	5	F
Scalability	60,91%	6	46,44%	6	R
Service omnipresence	59,03%	7	48,93%	4	F
Fault tolerance	56,26%	8	41,34%	8	R
Quality of service	52,42%	9	35,06%	9	R
Spontaneous interoperability	49,28%	11	31,70%	11	F
Universal usability	-	-	-	-	R
Service Discovery	41,23%	12	30,43%	13	F
Invisibility	40,61%	13	31,06%	12	F
Function Composition	32,02%	14	23,74%	14	F

## 4.2. Data Analysis Planning

For the data analysis stage, it was necessary to define how the following variables could be calculated: subject's weight, characteristic's pertinence, and characteristic's relevance level [8]. The formula used to obtain the weights for a subject "i", is:

$$Weight(i) = f(i) + p(i) + e(i) + \frac{t(i)}{MedianTP} \text{ , where:}$$

- $f(i)$  is the higher level of academic degree;
- $p(i)$  is the indicator about the number of papers regarding Ubicomp published by the subject;
- $e(i)$  is the subject's experience level regarding the development of ubiquitous software projects;
- $t(i)$  is the total number of ubiquitous software projects that he/she has participated;
- $MedianTP$  is the median of the total number of ubiquitous software projects considering the answers of all subjects.

To calculate the pertinence level of a ubiquity characteristic to characterize ubiquitous software projects, it is necessary to sum the answer of each subject multiplied by its respective weight:

$$Pertinence(j) = \sum_{i=1}^M (Answer(i,j) * Weight(i)) \text{ , where:}$$

- $Pertinence(j)$  is the total value of the answers of all subjects (multiplied by their weights) about the adequacy of the characteristic j to characterize ubiquitous software projects;
- $Answer(i,j)$  is the indicator of being adequate (1) or not (0), defined by the subject i for the characteristic j;
- $Weight(i)$  is the weight attributed for the subject i;
- $M$  is the amount of survey's subjects.

The definition if a characteristic is adequate or not to characterize ubiquitous software projects must be based on a cut-off point, that is, a threshold indicating if the characteristic shall be included (value greater than the threshold) in the final set or not (value lower than the threshold). The threshold is 50% of the maximum value that could be obtained for a characteristic j in the variable  $Pertinence(j)$  if all subjects answer YES regarding its adequacy to characterize ubiquitous software projects.

$$Threshold = 0,5 * \sum_{i=1}^M Weight(i)$$

Finally, to define the relevance level of each characteristic classified previously as adequate, it is

necessary to sum the answers of each subject multiplied by its respective weight.

$$RLevel(j) = \sum_{i=1}^N (Scale(i,j) * Weight(i)) \text{ , where:}$$

- $RLevel(j)$  is the total value of the answers of all subjects (multiplied by their weights) for the characteristic j;
- $Scale(i,j)$  is the scale of relevance level (0-5) defined by the subject i for the characteristic j;

## 4.3. Results

This survey was executed considering a population of 280 subjects. Of this total, 31 researchers from different regions (North America, Asia and Europe) answered the questionnaire (about 11%). 22 of them have Ph.D., 7 Masters, and 2 undergrads. On average, subjects had already participated in 7 ubiquitous software projects.

As a result, it was possible to evaluate the organized body of knowledge by identifying the pertinence level and the relevance level of each ubiquity characteristic.

In addition, three researchers reported the need of including a new characteristic: universal usability, which is associated to the fact that the project usability is adhering to good usability standards, while considering different target user groups.

Table 2 summarizes the reached results:

- The rows highlighted in gray indicate the characteristics to be considered in the final set of ubiquity characteristics. This selection was performed according to the inclusion criteria defined in the survey plan;
- Context sensitivity and adaptable behavior are the most pertinent and relevant characteristics;
- We could observe that there is a balance between functional (6/11) and restrictive (5/11) characteristics. This can indicate that non-functional characteristics are critical for this software category.

Based on those findings, we could update our interpretation about the ubicomp definition presented on section 2 for: *Ubiquitous computing is present when computational services or facilities can materialize themselves at any time or place, transparently, through the use of common daily devices. To make it happen it is desirable that systems of this application category take the following characteristics into consideration:*

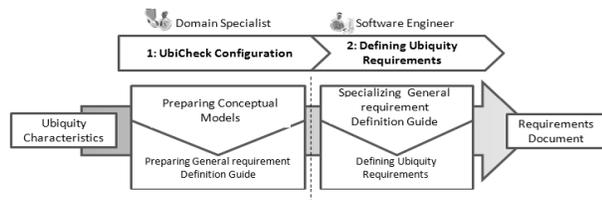


Figure 2. UbiCheck overview.

- **Functional:** context sensitivity, adaptable behavior, service omnipresence, heterogeneity of devices, experience capture, spontaneous interoperability;
- **Restrictive:** scalability, privacy and trust, fault tolerance, quality of service, and universal usability.

In the next section an application example of this body of knowledge to support building new software technology regarding the requirements engineering area will be briefly described.

## 5. Supporting Requirements Definition in Ubiquitous Software Projects

Requirements engineering represents a crucial phase for software development [12] [13]. Several techniques and methods have been proposed to deal with it [15]. However, most of them support the development of conventional projects which can reduce its efficacy and efficiency when working with specific application domains such as ubiquitous software projects [14].

Thus, to assure the quality of ubiquitous software and reduce costs associated with rework, the set of ubiquity characteristics and their respective factors (presented on sections 2, 3, and 4) can be used to support software engineers during requirements definition concerned with the ubicomp domain. For instance, UbiCheck represents a checklist-based approach to support requirements definition in the ubicomp domain [11]. Its goal is to help software engineers during the requirements definition increasing the efficiency and effectiveness by reducing the number of omission defects and execution time.

Figure 2 represents the steps and activities regarding UbiCheck application. The approach is composed by 4 sequential activities grouped in 2 main steps:

- Configuration: Its main goal is to create a set of guidelines based on the ubicomp domain to guide the identification of ubiquity requirements in software projects. These guidelines are composed of (see Figure 3): (A) Instructions about the guidelines use; (B) Ubiquity characteristics that could be present on the current software project; (C) Relevant information about ubicomp; (D) Questions about the relevant ubiquity information to be identified by the software engineer; (E) Link to a glossary; (F) Additional information about what is expected to be answered for the question, and; (G) Suggestions where that concept could be described in the requirements specification.

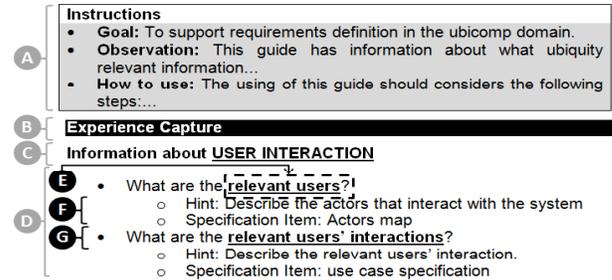


Figure 3. Fragment of a requirements definition guide.

- Defining Ubiquity Requirements: the goal of this step is to tailor the general set of guidelines for a specific project, and to use the specialized guidelines in a specific software project to support the requirements definition activities.

This approach has already been evaluated by an empirical study. The goal of the study was to analyze UbiCheck, with the purpose of characterization, with respect to its applicability, into the context of ubiquitous software projects from the point of view of software engineering students. The population of this study was represented by software engineering Master and PhD students. A set of 8 subjects from an Object Oriented Software Engineering course took part. They were grouped in 3 teams (A-3, B-3, and C-2 subjects). Each team received two ubiquity scenarios to evolve the requirements specification of an Inventory Management System.

The study allowed us to identify some initial behaviors (other results including improvements and restrictions of ubicheck and study limitations can be found in [11]):

- The subjects reported that UbiCheck helped in the ubiquity requirements definition activity. They said the questions in UbiCheck have led them to think about the important issues regarding ubicomp domain that they usually do not think nor capture when working with traditional software requirements documents.
- UbiCheck was able to reduce on average 23.3% the effort in time to define ubiquity software requirements when compared with the *ad-hoc* approach.

By using UbiCheck, the organized body of knowledge allowed us: (i) to characterize software projects regarding ubiquity; (ii) to prepare conceptual models to support the approach specialization according to the software project characterization; (iii) to define checklists to guide the software engineer on the definition of requirements for ubiquitous software projects. More details about the technology and its evaluation can be found in [11].

## 6. Conclusions and Current Work

In this paper, the organization and evaluation of ubicomp characteristics and their factors through the use of a research strategy based on primary and secondary

studies was described. From this point of view, this research strategy allowed us to reach some results:

- (1) 1<sup>st</sup> and 2<sup>nd</sup> *quasi*-Systematic Reviews: a more recent definition for ubicomp and its characteristics, and identification of functional and restrictive factors for each ubicomp characteristic;
- (2) Initial Survey: improvement of the body of knowledge considering functional and restrictive perspectives and 3 new characteristics;
- (3) Body of Knowledge Evaluation: evaluation and improvement of the body of knowledge through the definition of pertinence and relevance of ubicomp characteristics.

On February 2010, motivated by the fact that the initial literature review just considered papers published until 2005 and its results were evaluated by a survey on 2008/2009, we have updated the body of knowledge considering the scientific papers published from 2006 until 2010. For this, a literature review protocol was planned and executed considering the Scopus digital library. After its execution, 34 additional papers were selected for data analyses, resulting: **(i)** it was not possible to identify any new ubicomp characteristic; **(ii)** 7 new functional factors were identified and included in the body of knowledge.

These results allow us to say that the body of knowledge presented on this paper remains valid, stable and updated considering the technical literature regarding current ubicomp research.

We consider the organization of this body of knowledge an important step towards the definition of new software engineering technologies to support the development of ubiquitous software projects. As an initial example we have described UbiCheck. However, it is important to note that its use is not limited for this type of software technology. Other usage opportunities are also possible, such as:

- to prioritize the development of new software technologies according to the pertinence and relevance levels of each characteristic;
- to identify project risks associated to each ubicomp characteristic;
- to provide guidance on architectural design based on the characteristics definitions and their respective factors;
- to select efficient testing strategies for ubiquitous software projects (different ubicomp characteristics could bring different restrictions about testing).

Thus, we hope these results can provide some hints and directions to new research trends regarding software engineering applied to ubiquitous software projects. Currently, we are working on the development of an approach to support the verification of ubiquitous software requirements based on this body of knowledge.

## 7. Acknowledgements

Our thanks to CNPq (Grant 75459/2007-5), CAPES, and FAPERJ for the financial support.

## 8. References

- [1] Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J., and Burgelman, J.-C. (2003). "Ambient Intelligence: From Vision to Reality". IST Advisory Group Draft Rep.
- [2] Niemela, E., Latvakoski, J., 2004. Survey of requirements and solutions for ubiquitous software, Proc. of the 3rd Int. Conf. on Mobile and ubiquitous multimedia, October 27-29 Maryland.
- [3] Sakamura, K. (2006) "Challenges in the Age of Ubiquitous Computing: A Case Study of T-Engine, An Open Development Platform for Embedded Systems". Proc. 28th ICSE, Shanghai, China. Pages: 713 - 720.
- [4] Spínola, R.O., Silva, J.L.M., Travassos, G.H. (2007) "Checklist to Characterize Ubiquitous Software Projects". Proc. XXI Brazilian Symposium on Software Engineering.
- [5] Weiser M. (1991) "The Computer for the 21st Century". Scientific American, pp. 94-104.
- [6] Spínola, R.O., Pinto, F.C.R., Travassos, G.H. (2008) "Supporting Requirements Definition and Quality Assurance in Ubiquitous Software Project". In: 3rd ISOLA, Greece.
- [7] Shaw, M., 2002. What Makes Good Research in Software Engineering?. Presented at ETAPS 2002, Grenoble, France; International Journal of Software Tools for Technology Transfer, 2002, vol. 4, no. 1, pp. 1-7.
- [8] Dias-Neto A.C., Travassos, G.H., (2008); "Surveying on Model Based Testing Approaches Characterization Attributes", In: 2nd ESEM, Germany, 2008.
- [9] Biolchini, J., Mian, P.G., Natali, A.C.C., Travassos, G.H. (2005). "Systematic Review in Software Engineering". Technical Report ES 679/05. COPPE/UFRJ.
- [10] Dias-Neto, A. C.; Spínola, R.O.; Travassos, G. H., (2010). Developing Software Technologies through Experimentation: Experiences from the Battlefield. To appear in: XIII Ibero-American Conference on Software Engineering, Ecuador.
- [11] Spínola, R.O.; Pinto, F.C.R.L Travassos, G. H.. (2010). UbiCheck: An Approach to Support Requirements Definition in the Ubicomp Domain. In the Proc. of the 25th Symposium On Applied Computing – SAC 2010, Sierre, Switzerland.
- [12] Boehm, B. W., Basili, V.R., 2001, "Software Defect Reduction Top 10 List.", IEEE Computer 34: 135-137.
- [13] Wheeler, D.A., Brykezynski, B., Meeson, R.N., 1996, Software Inspections: An Industry Best Practice, IEEE.
- [14] Oliveira, K. M., Zlot, F., Rocha, A.R. C., Travassos, G. H., Silva, C. G. M., Menezes, C. S., (2004). Domain Oriented Software Development Environment. Journal Of Systems And Software, v. 72, n. 2, p. 145-161.
- [15] Nuseibeh, B., Easterbrook, S., (2000). "Requirements engineering: a roadmap," in ICSE '00: Proceedings of the Conference on The Future of Software Engineering. New York, NY, USA: ACM Press, pp. 35-46.
- [16] Travassos, G. H.; Santos, P. S. M; Mian, P.; Dias Neto, A.C.; Biolchini, J., (2008). An Environment to Support Large Scale Experimentation in Software Engineering. In: IEEE Int. Conf. on Engineering of Complex Computer Systems, Belfast.
- [17] Pai, M., McCulloch, M., Gorman, J.D., *et al.*, 2004. Systematic Reviews and meta-analyses: An illustrated, step-by-step guide. The National Medical Journal of India, 17, 2.

# Software Engineering in the Embedded Software and Mobile Robot Software Development: A Systematic Mapping

Daniel Feitosa, Katia R. Felizardo, Lucas Bueno R. de Oliveira, Denis Wolf, Elisa Y. Nakagawa  
Dept. of Computer Systems, University of São Paulo - USP  
PO Box 668, 13560-970, São Carlos, SP, Brazil  
fdaniel@grad.icmc.usp.br, {katiarf, buenolro, denis, elisa}@icmc.usp.br

## Abstract

*Currently, embedded software have been required more and more by a diversity of new products. As a consequence, an increase in the software complexity can be observed, requiring more attention to the software quality. Initiatives of exploring software engineering knowledge to develop this type of software can be identified, resulting in the Embedded Software Engineering (ESE) research area. However, there is a lack of a complete panorama about researches conducted in the context of ESE. This paper intends to present a view about how software engineering has been currently used in the embedded software development. For this, we have used systematic mapping, a technique based in the Evidence-Based Software Engineering (EBSE). Achieved results point out that in spite of the increase in the interest of applying software engineering to develop embedded software, there are still important lines of research that must receive attention.*

## 1. Introduction

Recently, a large amount of products containing embedded software, for instance, mobile telephones, cars and aircrafts, has been developed and used, bringing an effective impact to the society [6]. Furthermore, a specific type of product, the robots, has been also developed, covering a wide range of important applications. According to Graaf et. al [4], in the next years, the market for these products is forecasted to grow exponentially. However, the complexity and diversity of these products and, as a consequence, the embedded software, are increasing, creating a considerable challenge for embedded software development. In this perspective, both academic and industrial research have therefore been concerned and focused on their development, aiming at achieving sufficient product quality and timely delivery [4].

It is worth highlighting that software engineering technologies (that include software engineering activities, processes, methods, techniques, approaches, and tools) have provided considerable contribution to software development, through application of a systematic, disciplined, and quantifiable approach to the development, operation, and maintenance of software. Thus, in order to optimize the timeliness, productivity, and quality of embedded software, consolidated knowledge and software engineering technologies could support the development of such software. In this scenario, interesting and important initiatives have been proposed [5, 6]. As a consequence of this need and initiatives, a new research area — the Embedded Software Engineering (ESE) — has been recently established. It aggregates new needs in software engineering, since embedded software development is fundamentally different from that of nonembedded software [4]. In spite of ESE achievement and related work, there is a lack of recent work that summarize and present a complete overview about how software engineering technologies has been explored to develop embedded software.

In another perspective, Evidence-Based Software Engineering (EBSE) has recently arisen, and has given relevant contribution to the Software Engineering area [3]. Considering the need of reviews to identify publications related to a specific subject when studying a new knowledge area, EBSE proposed systematic mapping [7]. It refers to a technique that provides a systematic way in order to comprehend a topic of research [7]. In this context, an individual evidence (for instance, a case study or an experimental study divulged in a publication/paper) which contributes to a systematic mapping is called *primary study*, while the result of a systematic mapping is a *secondary study*. In short, systematic mapping is conducted by planning, conducting of search, and screening of primary studies using inclusion and exclusion criteria [1]. This technique also conducts data extraction and analysis through the identification of categories and classification of the primary studies in these categories. As a result, this technique provides maps

(for instance, in table or graphical format) containing condensed information about a research area. Considering its relevance, systematic mapping for different domains can be found, including software engineering domain, for instance [2]. Thus, the conduction of systematic mapping in the ESE domain seems also to be interesting.

The main objective of this paper is to present a panorama containing possibly all research work that have investigated and applied software engineering to the development of embedded software, including robotic software. For this, we have adopted and conducted a systematic mapping. Results achieved point out that more efforts must be applied to effectively explore advantages and knowledge of software engineering in the development of such software. Thus, this paper is organized as follows. In Section 2 we present the conducted systematic mapping. In Section 3 we discuss about achieved results and in Section 4 we summarize our contributions and discuss perspectives for further work.

## 2. Conducted Systematic Mapping

Our systematic mapping aims at identifying primary studies that apply software engineering for development of embedded software and mobile robot software. This systematic mapping was conducted from August/2009 to November/2009 and involved five people (a researcher in software engineering, a researcher in embedded systems, a specialist in systematic mapping and two graduate students). In order to conduct our systematic mapping, we have used the process presented in [7]. In short, it is composed by four steps: (i) systematic mapping planning; (ii) conduct of the search; (iii) selection of the primary studies; and (iv) analysis, classification, and map building. These steps are explained in more details during presentation of our systematic mapping<sup>1</sup>. Following, we detail each step:

### 2.1 Step 1: Systematic Mapping Planning

In this step, it is defined the plan that will be used as basis to conduct the mapping. This plan consists of the formulation of the research questions, selection of the sources of primary studies, and establishment of selection criteria.

**Research Questions (RQ):** These questions are structured corresponding to the objective that is intended with the systematic mapping that, in our case, it is the identification of a panorama involving software engineering and embedded software or mobile robot software. Our research questions are presented in Table 1.

**Selection of Sources:** We selected larger and complete databases as sources of primary studies: IEEE Xplore,

<sup>1</sup>For the sake of space, this paper includes only the main information related to our systematic mapping. More details can be found in <http://www.grad.icmc.usp.br/~fdaniel/seke10>

**Table 1. Research Questions**

RQ	Description
1	Which software engineering technologies have been investigated for the development of embedded software and mobile robot software?
2	Which software engineering technologies have been more widely researched for the development of embedded software and mobile robot software?
3	Which software engineering technologies have been more recently researched for the development of embedded software and mobile robot software?

ACM Digital Library, Springer Link and Scirus. These databases are efficient to conduct systematic mapping in the context of software engineering [1, 3]. Furthermore, we decided that only papers written in English will be considered in our mapping, since English is more widely adopted to write scientific papers.

**Establishment of Selection Criteria:** Another important element of the systematic mapping planning is to define the Inclusion Criteria (IC) and Exclusion Criteria (EC). These criteria make possible to include primary studies that are relevant to answer the research questions and exclude studies that do not answer them. Thus, the inclusion criteria of our systematic mapping are presented in Table 2 and the exclusion criteria are presented in Table 3.

**Table 2. Inclusion Criteria**

IC	Description
1	The study involves software engineering technologies to test embedded software or mobile robot software.
2	The study involves software engineering technologies to maintain embedded software or mobile robot software.
3	The study involves software engineering technologies to manage requirements of embedded software or mobile robot software.
4	The study involves software engineering technologies to design embedded software or mobile robot software.
5	The study involves software engineering technologies to implement embedded software or mobile robot software.
6	The study involves software engineering technologies to improve quality of embedded software or mobile robot software.

**Table 3. Exclusion Criteria**

EC	Description
1	The study describes an experience related to teaching in ESE.
2	The study has been already analyzed (i.e., it is repeated).
3	The study scope is out of our interest.

### 2.2 Step 2: Conduct of the Search

In this step, the search by primary studies is conducted according to previously established plan. This identification is done by looking for all primary studies that match with the search string in the search sources. This can be automatically conducted if these sources provide an efficient search engine. For this, we established the keywords and the search strings.

**Keywords:** To keep the scope of the systematic mapping, the keywords chosen must be simple and well chosen. Thus, they must be simple enough to bring many results and also rigorous enough to cover only the desired area. The keywords chosen to our mapping are: *Software Engineering*, *Embedded Software Engineering*, *Mobile Robot* and *Mobile Robotic*.

**Search Strings:** Based on the keywords, the search string is built (through connections AND/OR) to incorporate the semantic expected to the search. The search string used in our mapping was: “*Embedded Software Engineering*” OR (“*Software Engineering*” AND (“*Mobile Robot*” OR “*Mobile Robotic*”)). We have also built the string and/or settings for each search engine, since each engine accepts a specific formatted. As result of this step, we obtained a total of 414 primary studies: 238 from ACM digital library, 98 from IEEE Xplore, 40 from Scirus and 38 from Springer Link.

### 2.3 Step 3: Selection of the Primary Studies

In this step, the selection criteria (i.e., inclusion and exclusion criteria) are applied to select the relevant primary studies. Table 4 summarizes the number of primary studies obtained after applying the inclusion/exclusion criteria. From a total of 414 primary studies previously identified, 99 studies (i.e., 23.91 %) were selected. It is observed that IEEE was the most effective source, because 42.86% of the primary studies were used, i.e., from total of 98 studies, 42 studies were relevant for our mapping. Regarding ACM, in spite of identifying more studies if compared with other sources, only 18.49% of the primary studies were selected. Regarding Scirus and Springer Link, besides few primary studies identified, only 17.5% and 15.79% of primary studies were respectively selected as relevant.

**Table 4. Amount of primary studies included and excluded**

Source	Included	Excluded	Total
ACM Digital Library	44	194	238
IEEE Xplore	42	56	98
Scirus	7	33	40
Springer Link	6	32	38
<b>Total</b>	<b>99</b>	<b>315</b>	<b>414</b>

### 2.4 Step 4: Analysis, Classification, and Map Building

In this section we present the categorization of the primary studies. For this, two tasks were conducted: (i) search by keywords; and (ii) grouping of the primary studies into categories. Firstly, primary studies were carefully read (i.e., title, keywords and abstract). As a result, considering terms that seem to be more relevant or their number of occurrence, these terms were selected as keywords of our

mapping: requirements, analysis, design, test, maintenance, tool, method, framework, study, implementation, architecture, UML, case study, reengineering, reuse, and simulation. Based on these keywords, categories were created. For this, research questions were considered. In the case of our mapping, we were interested in identifying relationship among software engineering and the development of embedded systems and mobile robot software. Thus, two categories were created: “Fundamental Activities of Software Engineering”; and “Main Topics of Research”.

In the first category, the keywords used were: Requirement Management, Analysis and Design, Coding, Testing and Maintenance. This can show which main software engineering activities have been investigated and/or used in the development of embedded software and mobile robot software. In the second category, the keywords used were: Tool, Method, Framework, Design, Coding, and Architecture. This can show which topics of the software engineering domain have been more applied in the development of embedded software and mobile robot software.

Following, we classified the primary studies into topics in the categories, noting that a study may fit into more than one topic per category. As result, we build two maps, one for each category. Figure 1 shows the map to the category “Fundamental Activities of Software Engineering”. It is observed that from first work in 1992, there is an increasing interest in inserting software engineering activities in embedded or robotic projects. In particular, issues related to analysis and design of embedded software and mobile robot software have been more intensively treated. However, important activities, such as requirement management, software testing and maintenance, must receive more attention.

Figure 2 shows the map to the category “Main Topics of Research”. The topics are: *Tool* (coding, analysis or presentation of a tool); *Method* (analysis or presentation of a method to develop software); *Framework* (coding, analysis or presentation of a framework); *Study* (presentation of analysis, for instance, comparison or review, involving previous work); *Coding* (coding or analysis of an implementation); and *Architecture* (analysis or presentation of an architecture). From this map, we can observe that in general there is an increasing interest in all these topics. However, studies that involve analysis of previous work are more numerous. Otherwise, more tools that support development of software for ESE domain must be proposed.

## 3. Discussion

Regarding research questions established for our mapping, it is observed that all of them were successfully answered. Thus, in general, knowledge about embedded software and mobile robot software development has been mapped. We believe that results presented in this work are representative of the whole embedded software domain,

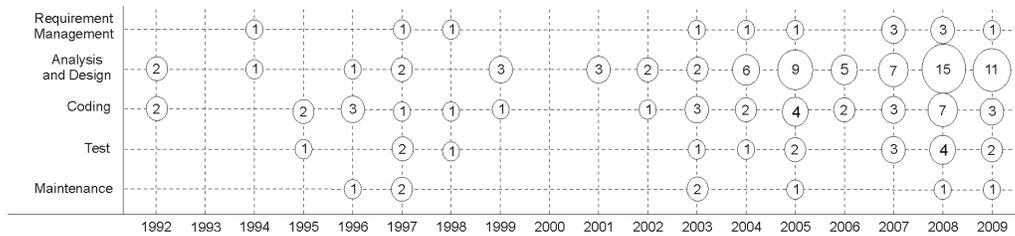


Figure 1. Map related to category “Fundamental Activities of Software Engineering”

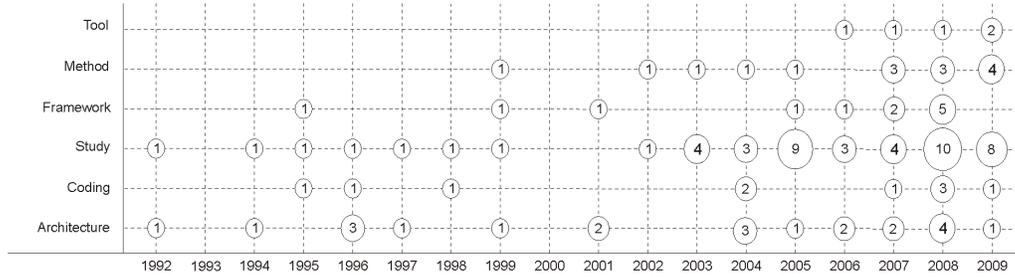


Figure 2. Map related to category “Main Topics of Research”

since systematic mapping provides mechanism to achieve it.

Considering knowledge arisen from this work, it is possible to identify interesting and important research lines that can be investigated in future work. For instance, there is a lack of experimental results, i.e., it is necessary that researchers bring the concern to analyze experimentally methods, techniques, architectures, and other software engineering technologies in the development of embedded software and mobile robot software. There are also other important research lines in software engineering that appear to be unexplored. For instance, software testing, reference architectures, and aspect-oriented development.

Regarding limitation of this work, other categories could be established. Other research questions could be also established, considering specific topics of research. Moreover, it is worth highlighting that conduct of a systematic mapping is not a trivial task, mainly because the number of papers that need to be manipulated. Besides that, both paper reading and category analysis are manually conducted. Furthermore, conduct of a systematic mapping involving a recent research area — for example, the ESE, considered in this work — is not easy. There is not consensus in terminology and concepts used by different researchers. Sometimes it is necessary to infer a conclusion to make some decisions.

#### 4. Conclusions

The main contribution of this work is to present an overview of software engineering application to the embedded software and mobile robot software development. We

believe that this view can contribute to the ESE area, getting new lines of research. Motivated by the promising results, we intend to conduct other systematic mapping involving more specific topics. Thus, we aim at achieving a better understanding about the intersection between embedded software development and software engineering.

**Acknowledgments:** This work is supported by Brazilian funding agencies: FAPESP, CAPES, and CNPq.

#### References

- [1] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil. Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software*, 80(4):571–583, Apr. 2007.
- [2] N. Condori-Fernandez, M. Daneva, K. Sikkil, R. Wieringa, O. Dieste, and O. Pastor. A systematic mapping study on empirical evaluation of software requirements specifications techniques. In *ESEM'09*, pages 502–505, Washington, USA, 2009.
- [3] T. Dybå, T. Dingsoyr, and G. K. Hanssen. Applying systematic reviews to diverse study types: An experience report. In *ESEM'07*, pages 225–234, Los Alamitos, USA, 2007.
- [4] B. Graaf, M. Lormans, and H. Toetenel. Embedded software engineering: the state of the practice. *IEEE Software*, 20(6):61–69, 2003.
- [5] M. Kim, S. Kim, S. Park, M.-T. Choi, M. Kim, and H. Goma. UML-based service robot software development: a case study. In *ICSE'06*, pages 534–543, New York, USA, 2006.
- [6] P. Liggesmeyer and M. Trapp. Trends in embedded software engineering. *IEEE Software*, 26(3):19–25, 2009.
- [7] R. S. M. Petersen, K. Feldt and M. Mattsson. Systematic mapping studies in software engineering. In *EASE'08*, pages 1–10, Bari, Italy, June 2008.

# A Visual Bug Report Analysis and Search Tool

Carlos Eduardo Albuquerque da Cunha<sup>1,2</sup>, Yguaratã Cerqueira Cavalcanti<sup>1,2</sup>,  
Paulo Anselmo M. Silveira Neto<sup>1,2</sup>, Eduardo Santana de Almeida<sup>2,3</sup>, Silvio Romero L. Meira<sup>1,2</sup>

<sup>1</sup>Center for Informatics – Federal University of Pernambuco (UFPE)

<sup>2</sup>Reuse in Software Engineering (RiSE)

<sup>3</sup>Computer Science Department – Federal University of Bahia (UFBA)

{ceac, ycc, pamsn, srlm}@cin.ufpe.br, esa@rise.com.br

## Abstract

*According to recent work, duplicate bug reports in bug trackers impact negatively on software maintenance and evolution productivity due to, among other factors, the increased time spent on report analysis and validation. In order to solve this problem, this work presents and evaluates a tool based on Information Visualization techniques, aiming to assist developers during analysis and identification of duplicate bug reports. The tool development was supported by a survey conducted to identify the best opportunities for visualization techniques, according to the cognitive process for understanding bug reports information.*

## 1 Introduction

Aiming to perform faster and more productive manners to manage incoming change requests, companies are relying on different sources for technical assistance, among them: *bug trackers*, such as Bugzilla<sup>1</sup>. These tools are responsible for receiving and managing new change requests and track all evolutionary information about them until their purposes are satisfied.

Unfortunately, the growing usage of such systems presented side effects; one of the most important being the **Duplicate Bug Report Problem** [1, 2, 3, 4]. This problem is characterized by the submission of two or more change requests that describe a single issue to the bug tracker responsible for a product, and its main consequence is the overhead of work to perform the search and analysis of such similar requests.

Motivated by this problem, this work presents an approach to the Duplicate Bug Report Problem, based on Information Visualization [5] techniques, that is able to assist users on handling important data while maximizing their experience and perception of the vital information to their tasks of analyzing and identifying duplicate change requests, thus decreasing this problem effect on software development teams productivity.

The remainder of this work is organized as follow: Section 2 presents a survey with developers to characterize the cognitive process for understanding bug reports information; Section 3 presents the tool developed to address the problem of duplication; Section 4 describes an experiment conducted to evaluate the tool; Section 5 presents a set of related work; and, finally, Section 6 presents the conclusion and future work.

## 2 Understanding Bug Report

Automatically retrieving information and evaluation of bug reports can assist developers, who, at the highest level, are the ones to assure duplication cases. Due to this scenario, a survey was conducted aiming to identify commonalities in bug report structures and understand how developers extract and interpret important information. The survey was applied to developers, who are regular users of bug report tracking systems, and it is described next. The complete questionnaire can be found in [6].

*Q1. How long have you work in software development?* In general, the group of respondents can be considered to have high experience in software development. Most developers, 60% of those who answered the questionnaire, have 5–10 years of experience and other 12% with more than 10 years of experience. Only 28% of developers indicated that have less than 5 years of development experience.

*Q2. How many companies have you worked for?* This is particularly interesting to discover the number of different development processes which developers were inserted. Each process has its own defined set of activities and restriction about maintenance, which includes rules for creating and handling bug reports. Only 16% of all developers worked for just one company, with 48% of developers working for 2–3 different companies and 36% of participants working for at least 4 companies.

*Q3. How many software development projects have you contributed?* A higher number of projects can help developers face different expectation from stakeholders. A contribution to a software project can reflect participating in

<sup>1</sup><http://www.bugzilla.org>

both development phase and maintenance phase, the later being one of the focused of this survey. As for results, 12% of developers indicated that had contributed to more than 20 projects, while 36%, 32% and 20% of developers contributed to 10–20, 5–10 and less than 5 projects respectively.

*Q4. How many of these projects used bug trackers for managing change request?* According to received answers, 20.83% of the developers indicated that more than 75% of the projects he/she participated used bug report tracking systems, followed by 37.5% indicated that 50–75% of the projects used that type of tool. An additional 29.17% of developers indicated that 25–50% of the projects used bug trackers, and, finally, 12.5% indicated that less than 25% of projects were assisted by bug trackers.

*Q5. How many different bug trackers have you used? Which one(s)?* Developers who used more of such tools have better conditions to give comparative opinions. Analyzing the results, we identified that the majority of the developers have good comparative bases because 68% of developers used at least 3 different bug trackers, while 24% used 2 different tools and only 8% of the developers who answered the questionnaire used one bug tracker.

Another aspect is the dominance of certain bug trackers. Bugzilla and Mantis have been used by 71% of the developers, while the third most used, Trac, reached only 25%. Others bug trackers did not reach 10% of developers.

*Q6. What are the most common fields within the bug trackers have you used?* As a result, a common bug report structure would consist of the given fields: (100%) Description; (86%) Severity, Assigned to, and Priority; (82%) Title; (68%) Summary; (64%) Attachments; (55%) Current Status, Comments and Bug Report Identification Number; and (50%) Change History. The percentage in parentheses represents the occurrence of that field in the different bug trackers used by the developers. Responses from developers who used only one system were discarded.

*Q7. When analyzing bug reports, is there a most important/essential field? Which one?* As a result, 88% of developers affirmed that bug reports have a most in influential field. Then, developers were asked which field is the most important and, as a result, the description field was selected by 56% of developers, followed by summary and title fields selected 12% each. Other fields summed up 20%.

*Q8. What kind of information do you expect from this most important field?* Most developers are concerned about how exactly the problem happened and the steps to reproduce it. They expect a short and clear explanation about the flaw and what behavior was expected by the system.

*Q9. Which other fields are used to fully understand the reported issue?* Other auxiliary fields may be important to fully understand the issue. An auxiliary field can provide contextual information, specific data or more detailed information that could not be provided in the most important

field. Summary field was selected by 40% of developers, followed by attachments (32%), comments (28%), title and severity(24%), and priority (20%).

The last set of results analyzed are concerned with bug trackers functionalities for searching, analyzing and identifying bug similarities and duplicate cases. Developers were questioned about whether they perform searches to locate similar and duplicate report to better understand the problem being analyzed. Most developers, 64%, affirmed that they perform searches to locate similar reports, and 76% never used a bug tracker that provided an automatic feature for such means.

This results is important because allows us to identify a point for optimization in the bug report analysis process. To ratify this conviction, developers were asked about their wiliness for a tool to automatic analyze and present similar and duplicate reports. As a result, 88% of developers think a tool with that capability is useful and 88% believe the analysis process can be optimized by its usage.

As a final question, developers answered whether automatic analysis and identification of similar and duplicate reports would be essential to the analysis process; opinions were divided with 48% agreeing with the assumption and the other 52% believing otherwise.

### 3 A Visual Bug Report Analysis/Search Tool

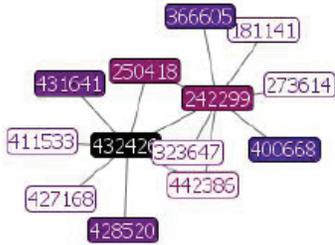
The tool is composed buy two main components: (i) *Visual Component* and (ii) *Search Component*. Each component was divided into modules to facilitate implementation and functionality modularization.

The **Visual Component** is responsible for the implementation and deploy of Information Visualization techniques for data display and developers interaction. This component is divided in four main modules: (i) *Search Panel*, (ii) *Data Display*, (iii) *Relationship Display*, and (iv) *Context Display*. For a full sized view of the tool, please refer to <http://tinyurl.com/seke-screenshot>.

*Search Panel*. This panel is located at the top. It was composed to provide a simple functionality and quick understand of it: begin analysis for the intended report. Once the search button is clicked, the analysis functionality of the tool is trigged.

*Relationship Display*. In order to present relationships among reports and group of reports, a graph visualization was implemented. The Graph Visualization enables a quick perception of relationships among reports by connecting the returned similar ones (Figure 1).

*Data Display*. This module displays all important data store within a bug report (Figure 2). This component is particularly important to assist developer in deeper analysis and comparison of contents to ratify ongoing duplicate cases. The developer must click on a report identification number on either graph (*Relationship Display*) or chart (*Context Display*) visualization to visualize the data.



**Figure 1. Reports nodes relationships**

Each report is displayed in the graph as a node with its identification number. The analyzed report node is the most dark one. Other reports nodes are filled in with white color and their border and identification number printed according to a red-blue color scale, where reports with higher similarity rates are painted in predominant red color variation, and reports with lower similarity rates are painted in predominant blue color variation.



**Figure 2. Bug report data display**

Once the tool search functionality is activated, the analyzed report node is painted followed by its similar reports nodes; then, every time a report node is clicked new nodes are added to the graph, representing the collection of similar reports to the clicked one. The newly added nodes are painted using the same red-blue color scale; however, the similarity rate used as base for indicating the color is calculated according to the original analyzed report.

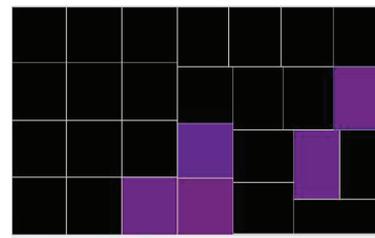


**Figure 3. Chart view**

The graph visualization is connected with other visual-

izations in the tool interface, in the sense that, once a node is selected, its counterpart in the chart view (Figure 3) is highlighted, its associated component is highlighted in the TreeMap view (Figure 4) and the report data, which it represents, is displayed in the Data Display visualization.

*Context Display.* This module provides two visualization techniques: Chart View (Figure 3) and TreeMap (Figure 4). The Chart Visualization was selected because we intended to provide a comparative base for similarity rates of the returned similar bug reports alongside the distribution of those bug reports throughout system components. As for the TreeMap Visualization, the objective was to provide an overview of the impact of the returned bug reports in the existing system components.



**Figure 4. Component view**

According to the survey presented, automatic generated similarity rates provide a starting point for developer analysis and comparisons of reports to identify duplicate occurrences. The presentation and distribution of reports throughout system components was a design decision to enhance developers awareness of contextual information.

The chart view also utilizes red-blue color scale to print report representations and connective actions – highlighting counterpart representation in graph view (*Relationship Display*), highlighting associated component in TreeMap view and display report data in *Data Display* module.

The TreeMap View was designed to present an overview of the effect of retrieved similar reports on system components. Each rectangle printed represents a system component and its level on components tree.

If a component is not associated with any report returned for analysis, that component is filled up with black color, while, a component associated with a group of reports is filled according a red-blue color scale, where the average of all similarity rates is used to determine the color. Additionally, the TreeMap view implements connective actions with other visualizations, where once a component is selected, reports representation in both graph *Relationship Display* and chart views are highlighted for quick identification.

All visualizations were carefully implemented to perform global interactions and speed up developers cognition, that means, an action performed in a given visualization also effects others, thus, developers can perceive and understand bug reports influence on similar and software components.

**Search component** is responsible for every data validation, processing, search, classification, rank and retrieval which happens throughout developers interaction with the tool. This component is divided into three major modules: (i) *Validation Module*, (ii) *Classification and Rank Module*, and (iii) *Data Repository*.

*Validation Module.* This module is responsible for validating income data parameters for all functionalities and search for valid bug report entries. This module provides security and data integrity for developers throughout analysis, implementing major business rules for data validation procedures.

*Classification and Rank Module.* This module is responsible for creating, retrieving and comparing a bug report representation against the whole bug report population to return most significant similar reports. The most important functionalities for the analysis process are implemented and deployed within this module. The list of functionalities are described as following.

*Parsing* – It is only used when a bug report, or set of reports, are uploaded to the tool to be stored in the data repository and available for analysis. *Text Processing* – This functionality is responsible to process text fields from bug reports (description, summary, and so forth) in order to create report’s representation. The functionality include: text *tokenization*, stop-words removal, and stemming using Porter algorithm [7] for stemming. *Classification and Ranking* – To classify and rank results for similar report search, it was used the Vector Space Model (VSM) [8]. Each term is associated with a *Term Frequency-Inverse Document Frequency* (TFIDF) [9] metric, which states that the value of the weight of a term is calculated by the frequency in which this term appears in the collection of documents, and the frequency in which it appears in each document.

*Data Repository.* This module is responsible to persist all bug report data and representation throughout analysis.

## 4 Experimental Study

This section presents the experimental study designed to evaluate the tool. The experiment was defined using the Goal Question Metric method [10]. Goal Question Metric is very useful because it guides the experiment by establishing the goal of the study, the questions to be answered, and all metrics used in order to interpret those answers.

The goal of this experiment is to analyze **a tool to improve analysis and identification of duplicate bug reports by developers** for *evaluating it* with respect to its *effectiveness* and *efficiency* on optimizing the analysis of bug reports, by decreasing time for identification and increasing duplicate identification, in the *view point of* researchers, in the *context* of software development projects.

*Quantitative* and *qualitative* questions were defined in order to reach the established goal. The first ones focuses on the data collected during the execution of the experiment,

and the others concerned with developers feedback about the new tool, its adoption and visual and interactive features. The questions are as following.

- 
- |     |   |
|-----|---|
| Q1. | Does the tool decrease the time for analysis of duplicate bug reports?    |
| Q2. | Does the tool increase the identification rates of duplicate bug reports? |
| Q3. | Is the new tool useful?   |
- 

**Planning.** All participants received explanations about the experiment and its goals. Each participant received a previous selected list with bug reports identification numbers to perform analysis. The list is completely the same for every participant, where 18 reports are known duplicate and 13 reports are considered unique. The results will be collected using a *time-sheet*. The null ( $H_0$ ) and alternative ( $H_1$ ) hypotheses were defined as follows:

$$H_0: \mu_{time\ with\ tool} \geq \mu_{time\ with\ baseline} \quad \text{and} \\ \mu_{identification\ with\ tool} \leq \mu_{identification\ with\ baseline}.$$

$$H_1: \mu_{time\ with\ tool} < \mu_{time\ with\ baseline} \quad \text{and} \\ \mu_{identification\ with\ tool} > \mu_{identification\ with\ baseline}.$$

In order to test both hypothesis defined for the experiment, descriptive statistics was used to analyzed and interpret the results. Qualitative analysis was conducted to understand subjective aspects of the new tool, such as visualizations performance and usefulness.

It was used the *two-group posttest-only randomized experiment*, which defined two groups to participate in the experiment. Participants assigned to the first group were required to use the new tool, while participants assigned to the other group were asked to use the baseline tool.

Regarding to the formation of participants groups, the randomization principle was applied on the assignment of participants to a tool [11]. The experiment was considered balanced since each tool received a group with the same number of developers and with similar overall experience. This fact strengthens the statistical analysis of the data [11].

Finally, the following threats to the validity of this experiment were identified:

*Boredom.* Some participants may felt upset or disappointed with the experiment since it took great amount of time and effort from them;

*Participants knowledge on bug reports contends.* The results could be influenced by the lack or excess of knowledge of participants regarding the contends of bug reports data;

*Environment.* Environments have positive and negative effects on participants. Developers assigned to the baseline tool could perform their analysis any place they wish since a link in the Internet was available for them; although it provides more comfort, results could be compromised due to slower internet connections.

**Operation.** The Firefox browser was selected as target to the experiment, because it is a well known product by de-

velopers and can provide a large diversified bug repository. The Bugzilla bug tracker was loaded with bug reports from Firefox. It was chose to be the Baseline tool for the experiment because it is used as the official Firefox bug repository and prior developers knowledge on its operation.

The bug repositories for both tools were composed by the same set of Firefox bug reports. No participant of the experiment had previous knowledge about any of the bug reports in the bug repository or in the list given for analysis. Additionally, participants were not informed how many reports were duplicates.

A total of 20 participants were selected by convenient sampling method [11, 12] and distributed into two groups – 10 participants each. All participants are developers from the local software development industry in Recife, Brazil.

**Analysis of the Results.** *Descriptive statistics.* An overview of statistics extracted from the experiment results is shown in Table 1. Analyzing the time aspect for identification of duplicate bug reports, the new tool proved to be more effective than Bugzilla; generally, the average time necessary for analysis decreased 67%, from 5.06 minutes presented by participants assigned to Bugzilla to 1.96 minutes by participants assigned to the new tool.

As for correctly identifying duplicates and unique occurrences, both groups of participants had similar performance. While participants assigned to the new tool almost reached the 75% average mark for correctly classifying reports, participants assigned to Bugzilla nearly got 73%.

	Time spent in analysis		Correct analysis	
	Tool	Bugzilla	Tool	Bugzilla
Mean	1.96	5.06	23.2	22.6
Maximum	4.93	9.62	28	26
Minimum	0.52	3.06	19	18
S.D.	1.32	2.08	2.96	2.29

**Table 1. Experiment results overview**

Although the similar performance of participants from both tools, a deeper analysis of the results is required to strengthens positive aspects of them. Regarding duplicate identification, participants using the new tool reached 77.78% of correct identification while Bugzilla users had 64.44%; many reasons contributed to this increase in identification rate, among them are the visualizations deployed which highlighted important data for comparison.

As for identifying unique reports, Bugzilla users presented better results with 84.61% of correct identification, while participants using the new tool reached 70.77%. However, the difference shown in false-positive rates – unique reports classified as duplicates – can be explained by the lack of knowledge of participants regarding the bug repository. Some participants commented that, when they were not sure about a report’s classification, they would classify it as duplicate.

The descriptive analysis showed that, in general, the new

tool proved to be more effective in analyzing and identifying duplicates than Bugzilla. The time aspect of the analysis and identification tasks presented the greater difference, where the new tool decreased the time necessary for analysis in 68%. As for correctly classifying reports, the tools presented similar overall results for correctly classifying reports; the new tool increased the identification of duplicate rate in 17.15%, however, it also increased the false-positive rate among unique reports from 15.39% – presented by Bugzilla users – to 29.23%.

*T-test.* The results collected after the experiment were used to perform a Student’s t-test [13] with 95% of confidence. Table 2 shows a summary of the t-test results. Analyzing the results for the time variable, the t-test rejected the null hypothesis; the new tool reduces the amount of time necessary to perform analysis and identification of duplicates. As for the identification aspect, the t-test did not reject the null hypothesis; there is no significant gain related to identifying duplicates with the new tool.

	Time spent in analysis	Duplicates identified
T-value	3.77	2.03
Degrees of freedom	18	18
Probability	0.05	0.05
T-distribution	2.10	2.10
Result ( $t \text{ value} > t$ )	$H_0$ : rejected	$H_0$ : not rejected

**Table 2. T-tests with 95% of confidence**

*Usefulness.* All participants rated the new tool as “Very Useful” when asked how they would rate the tool in terms of analysis and identification of duplicate reports. Visualization techniques were also rated as developers stated that Data Details and Chart View were considered the most useful. Data Details was considered indispensable because of its ability to display important data for user comparison in a simple way. Chart View was considered very useful for analysis due its ability to highlight the similarity rate among reports, calculated by the tool. Graph View did no received any significant comment. Finally, no participant mentioned the TreeMap View as a good features, nevertheless, some users stated that they would remove this visualization from future versions of the tool.

*Usability.* The first aspect regards the search functionality, which developers felt that a shortcut for the functionality would make the tool more complete and speed up their tasks. The second aspect was the real-time update to the Graph View, which some participants complained about turning it more difficult to read. As new nodes were added to the graph, graph edges and nodes were drawn one over the other.

## 5 Related Work

In work of Jhon Anvik [1], it was analyzed potential problems raised by bug repositories from open source projects. Cavalcanti at al. [2, 3, 4] goes further on the duplication problem and characterized it in order to understand

the consequences of it to software development and the factor that may intensify the problem.

Hiew's work [14] presented an approach to detect duplicate bug reports using clustering techniques. Runeson's work [15] addressed the bug report duplication problem using Natural Language Processing (NLP). While Wang et al. [16] proposed an approach to detect duplicate bug reports using NLP and execution information.

Jalbert and Weimer [17] build a system to automatically classify duplicate bug reports as they arrive. Bettenburg et al. [18] approached the same problem by providing developers with additional information, such as *tracebacks*. Sandusky et al. [19] proposed a method to identify and visualize bug report networks. With such networks, it was possible to analyze relationships among bug reports.

Cavalcanti [20] proposed a Web-based tool to improve search and analysis of duplicate bug reports. Similar to other works, it was used the Vector Space Model [9] to rank bug reports. Although this work had some aspects of information visualization and usability, it was not the main focus of the proposed technique.

We differ from other work because we performed a deeper analysis and understanding on the process which developers interpret bug reports data. Also, those approaches did not present any visual or interactive models for developers to validate possible duplicate cases, which was done in this work.

## 6 Conclusion and Future Work

This work proposed and evaluated a solution to the Duplicate Bug Report Problem. There are three main contributions made by this work: (i) presented a survey about developers cognitive process for understating bug report data importance and information extraction; (ii) developed a tool for classification and ranking of reports; finally, (iii) conducted an experiment, simulating real software development environment, comparing the proposed tool with a common, well-known bug tracker.

Some limitations and comments about the visualizations were reported by users which can lead to future releases of the tool: (i) improve search and ranking techniques; (ii) experiment replication, more can be done in terms of quantitative aspects to demonstrate significant and conclusive results; (iii) evolve the tool, participants appointed defects and improvements that can be made in terms of interactive features; and, finally, (iv) find appropriate developer, as the analysis phase is completed and a bug report is validated.

## Acknowledgement

This work was partially supported by the National Institute of Science and Technology for Software Engineering<sup>2</sup>, funded by CNPq and FACEPE, grants 573964/2008-4, APQ-1037-1.03/08 and RHAЕ-559839/2009-0.

<sup>2</sup>INES - <http://www.ines.org.br>

## References

- [1] J. Anvik, L. Hiew, and G. C. Murphy. Coping with an open bug repository. In *Proc. of the 2005 OOPSLA workshop on Eclipse technology eXchange*. ACM, 2005.
- [2] Y. C. Cavalcanti. A Bug Report Analysis and Search Tool. Master thesis, Federal University of Pernambuco, 2009.
- [3] Y. C. Cavalcanti. *A Bug Report Analysis and Search Tool: Improving search and analysis of duplicate bug reports*. LAP Lambert Academic Publishing, 2009.
- [4] Y. C. Cavalcanti, E. S. Almeida, C. E. A. Cunha, D. Lucrédio, and S. R. L. Meira. An initial study on the bug report duplication problem. In *Proc. of the 14th European Conf. on Soft. Maint. and Reeng. (CSMR)*. IEEE, 2010.
- [5] S. K. Card, J. D. Mackinlay, and B. Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann, 1999.
- [6] C. E. A. Cunha. A visual bug report analysis and search tool. Master thesis, Federal University of Pernambuco, 2009.
- [7] M. F. Porter. An algorithm for suffix stripping. *Program*, 3(14), 1980.
- [8] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. ACM, 1999.
- [9] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11), 1975.
- [10] V. Basili, R. Selby, and D. Hutchens. Experimentation in software engineering. *IEEE Trans.*, 12(7), 1986.
- [11] C. Wohlin, P. Runeson, M. C. O. Martin Höst, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishers, 2000.
- [12] B. Kitchenham and S. L. Pfleeger. Principles of survey research: part 5: populations and samples. *SIGSOFT Soft. Eng. Notes*, 27(5), 2002.
- [13] D. W. Zimmerman. A note on interpretation of the paired-samples t test. *Journal of Educational and Behavioral Statistics*, 3(22), 1997.
- [14] L. Hiew. Assisted detection of duplicate bug reports. Master's thesis, The University of British Columbia, 2006.
- [15] P. Runeson, M. Alexandersson, and O. Nyholm. Detection of duplicate defect reports using natural language processing. In *Proc. of the 29th Inter. Conf. on Soft. Eng. (ICSE)*. IEEE, 2007.
- [16] X. Wang, L. Zhang, T. Xie, J. Anvik, and J. Sun. An approach to detecting duplicate bug reports using natural language and execution information. In *Proc. of the 13th Inter. Conf. on Soft. Eng. (ICSE)*. ACM, 2008.
- [17] N. Jalbert and W. Weimer. Automated duplicate detection for bug tracking systems. In *The 38th Annual Inter. Conf. on Dependable Systems and Networks (DSN)*. IEEE, 2008.
- [18] N. Bettenburg, R. Premraj, T. Zimmermann, and S. Kim. Duplicate bug reports considered harmful? In *Inter. Conf. on Soft. Maint. (ICSM)*. IEEE, 2008.
- [19] R. J. Sandusky, L. Gasser, and G. Ripoche. Bug report networks: Varieties, strategies, and impacts in a f/oss development community. In *Proc. of the 1st Inter. Workshop on Mining Soft. Repositories (MSR)*, 2004.
- [20] Y. C. Cavalcanti, C. E. A. D. Cunha, E. S. Almeida, and S. R. D. L. Meira. BAST: A Tool for Bug Report Analysis and Search. In *XXIII Brazilian Symp. on Soft. Eng. (SBES)*, Fortaleza, Brazil, 2009.

# CFM: A File Manager with Multiple Categorization Support

Ali Sajedi Badashian<sup>1</sup>, Hamidreza Afzali<sup>2</sup>, Iman Khalkhali<sup>3</sup>, Morteza Ashurzad Delcheg<sup>1</sup>, Mohammad Shoja Shafiei<sup>1</sup>, Mehregan Mahdavi<sup>4</sup>

<sup>1</sup>Software Engineering Department, Islamic Azad University - Lahijan Branch, Iran

<sup>2</sup>School of ICT, Royal Institute of Technology (KTH), Sweden

<sup>3</sup>IT Department, Sharif University of Technology, Int. Campus, Kish Island, Iran

<sup>4</sup>Department of Computer Engineering, University of Guilan, Iran

sajedi@iau-lahijan.ac.ir, hr.afzali@gmail.com, imankhalkhali@yahoo.com, mor.delcheg@gmail.com, mohamadshafiei@gmail.com, mahdavi@guilan.ac.ir

**Abstract**— This paper introduces a new file manager to support multiple categorization. The proposed file manager is designed based on a subtle idea named Conceptual File Management (CFM). According to this approach, files are not contained by folders; nevertheless, each file can be a member of one or more folders (concepts). A prototype file manager is designed and implemented based on the new approach. Filtering by set operations and also manual concept selection improves retrieval of the files. CFM improves file system’s clarity and avoids ambiguity and redundancy. As a result, it reduces the size of file system and enhances file access.

**Keywords**—File Manager; Directory Structure; Information Retrieval; Information Visualization; Classification

## I. INTRODUCTION AND RELATED WORK

The folder paradigm and hierarchical file system implemented in UNIX-based systems in the 1970s is used as the primary mechanism for organizing files and folders [1]. This paradigm supports one-to-many relationships between a folder and the files. However, this is not absolutely the way as the users naturally think about organization and retrieval of their information [2]. Lansdale believed that the goal of organizing files is to facilitate their retrieval [2]. As a result, proper retrieval cannot be achieved in absence of suitable categorization tools that help to place a document in all related categories [1].

Quan, et al. implemented a suitable system to automatically categorize web bookmarks based on tags as categories [1]. However, their system implemented too many categories; hence, it cannot be used for large numbers of categories such as folders.

Barreau and Nardi defined the purpose of file search as *location-based search* and *logical search* [3]. For example, several information retrieval systems such as Haystack [4], Presto [5], WinCuts [6] and WorkspaceMirror [7] aid location-based search and GoogleDesktop [8], Mac OS Spotlight [9] and Phlat [10] provide logical search. For example, Phlat [10] is a lookup and tagging system in which searching is done for personal information including file system volumes, e-mail messages, web histories, etc. Cutrel, et al. merged search and browsing through a variety of contextual cues to form an

intuitive lookup interface. However, the achievement in Phlat was focused on logical search based on multiple cues, not the hierarchical structure of the objects.

Finally, VennFS is a prototype file manager that gives the ability of categorizing documents so that a file may be in several categorizes at a moment [11]. However, it abandoned hierarchical visualization of the file system. On the other hand, although their system works well in many cases, finding an item with growth of the bulk of data would be tedious. Moreover, they can address a file from within at most four folders.

## II. MULTIPLE CATEGORIZATION; CFM APPROACH

Suppose a user wants to have the video file of his/her paper presentation accessible both in “My Papers” and “My videos” folders. This is because of logical dependency between the file and the two folders. However, the first possible solution is to copy the video file to both folders. This leads to data redundancy that is undesirable. The redundancy problem causes ambiguity in having multiple versions of a single file in different folders, especially when the versions are updated frequently.

In order to support multiple categorization in file systems, the user may simply use shortcuts [12]. However, they have no extensive UI support nor make two-way relationships with the target files. For example, by moving or renaming the target file, the shortcuts remain useless. Using more advanced available facilities also cannot solve the problem. For example, although Symbolic Links (Soft Links) and Hard Links [12][13][14] enhanced the shortcuts in MS Windows and Linux systems, they do not solve this problem totally and many issues remain unsolved. For example, Hard Links of a file fail to work when changing the place of the file. They save the attributes of the target file separately, so all of them change due to redundant information in the Hard Links. Symbolic links do not support pursuit addressing. They remain orphan when deleting the target file. Besides, changing the place of the target file makes its symbolic links useless. Using Junction Points [14] in Microsoft Windows is also a means of mounting a file or folder to another partition to be appeared here, but actually being stored somewhere else. It is used due to space limitations in

partitions of disk. On the other hand, all of them add an extra layer to users' understanding of the file system even though all the mentioned aspects can be corrected.

CFM method is based on fundamental changes in two file system definitions; Folder and Containment. In hierarchical structures, a folder may contain some files and each file is contained by only one folder. In CFM, each file can be member of one or more sets which are called Concepts. Therefore, the "Containment" relation between files and folders is replaced by "Membership". Using this manner, the folders' hierarchy and file containment is replaced by concepts' hierarchy as logical categorization by which guide to access a file is provided. As a result, the user browses only the shadows of the files via one or more conceptual paths. A shadow is, in fact, a membership in a concept and contains two-way links to related physical files. Each file is at least member of one concept, hence has at least one shadow. In addition to Membership, more mathematical set operations as "Intersection" and "Union" are defined in CFM. However, the current view of the user over the files and folders is often remained unchanged.

### III. CFM SOFTWARE

CFM Software is developed with Microsoft Visual Studio 2008 based on .NET 2 Framework. It contains six Windows Forms and three Modules. Microsoft Access is chosen as its database for more portability. Although CFM offers multiple categorization with a hierarchical UI, it uses two flat structures in which files and concepts are stored respectively with their index numbers. Another tabular structure stores the memberships of the files in the concepts.

#### A. Software Environment

CFM looks like usual file browsers at first look (see Fig. 1.a and 1.b). There is a tree structure in the left side and a file panel in the right. The fundamental operations of CFM are described in this section.

##### 1) Browsing

The tree view shows the classification of concepts. The proposed file manager uses two concept selection methods and three file browse modes:

###### a) Concept selection methods:

- Select a single concept by clicking on its name (see the upper part of Fig. 1.b)
- Select one or more concepts by placing a check mark on the concept icon (see the lower part of Fig. 1.b)

###### b) File browse modes:

- Uni
- Multi  $\cap$
- Multi  $\cup$

Concept selections in the two selection methods can be made simultaneously (e.g. in the lower part of Fig. 1.b, the concept "Study" is selected. Simultaneously, the two check marks of "Study" and "Work" are checked). The selection(s) of one or more concepts is pursued and the file panel is refreshed based on "File browse mode" (shown on top of the file panel).

Concepts are not shown in file panel because the shown files are maybe the result of set operations (i.e.  $\cap$  or  $\cup$ ).

The first browse mode called "Uni" is the most liked method to common file managers. By clicking and selecting a concept, all shadows that are members of that concept will be displayed in the file panel. Concept selection for "Uni" is only based on selected (highlighted) concept (such as common file managers).

The two other modes are like applying mathematical  $\cap$  and  $\cup$  operations to a number of concepts. If no concept is checked in these two modes, no shadow will be displayed in the right panel. If one concept is checked, the result will be the same as selecting that concept in "Uni" mode. Checking more concepts will list only the files that are members of all chosen concepts / any of chosen concepts respectively for Multi  $\cap$  and  $\cup$  modes. Only the second concept selection method is considered if one of these two browse modes is selected. In other words, only the check marks are considered (in these two browse modes) and the selection of highlighted concept is disregarded.

##### 2) Importing Files and Folders

Considering the "file browse mode", "Import File" and "Import Folder" can be used to import new files and folders to CFM. Shadows are being added to the selected (highlighted) concept if the "Uni" mode is selected, but are added to all marked concepts (by checkmarks) if it is set to either of "Multi" options. A whole folder and its sub-folders can also be imported using "Import Folder" option. All the files are created physically and the appropriate shadows are added.

##### 3) Concept-Tree Pop-up Menu

When right clicking on a concept in the concept tree view (left panel), a pop-up menu including the following items is appeared (see Fig. 1.c):

- **New:** add a new concept in the selected concept.
- **Cut:** stores selected concept (and all its sub-concepts hierarchically) in clipboard with cut flag.
- **Copy:** stores selected concept (and all its sub-concepts hierarchically) in clipboard with copy flag.
- **Paste:** copies/moves the concept in the clipboard and all its sub-concepts and shadows to the selected concept based on the last copy/cut triggered operation. Note that the "File browse mode" doesn't affect the concept operation considered in this section. They only affect the manner of visualizing the shadows.
- **Rename:** used to rename a concept.
- **Delete:** deletes a concept with all its sub-concepts and their shadows.
- **Uncheck All:** removes checkmarks of all checked concepts.

##### 4) Shadow Pop-up Menu

By right-clicking on one or more shadows in the right panel, a drop-down menu will be shown as seen in Fig. 1.d. The items of this menu are as follows:

- **Open:** opens the selected shadow with its appropriate application.
- **Cut:** stores selected shadow in clipboard with cut flag.
- **Copy:** stores selected shadow in clipboard with copy flag.
- **Paste:** copies or moves the shadows currently in the

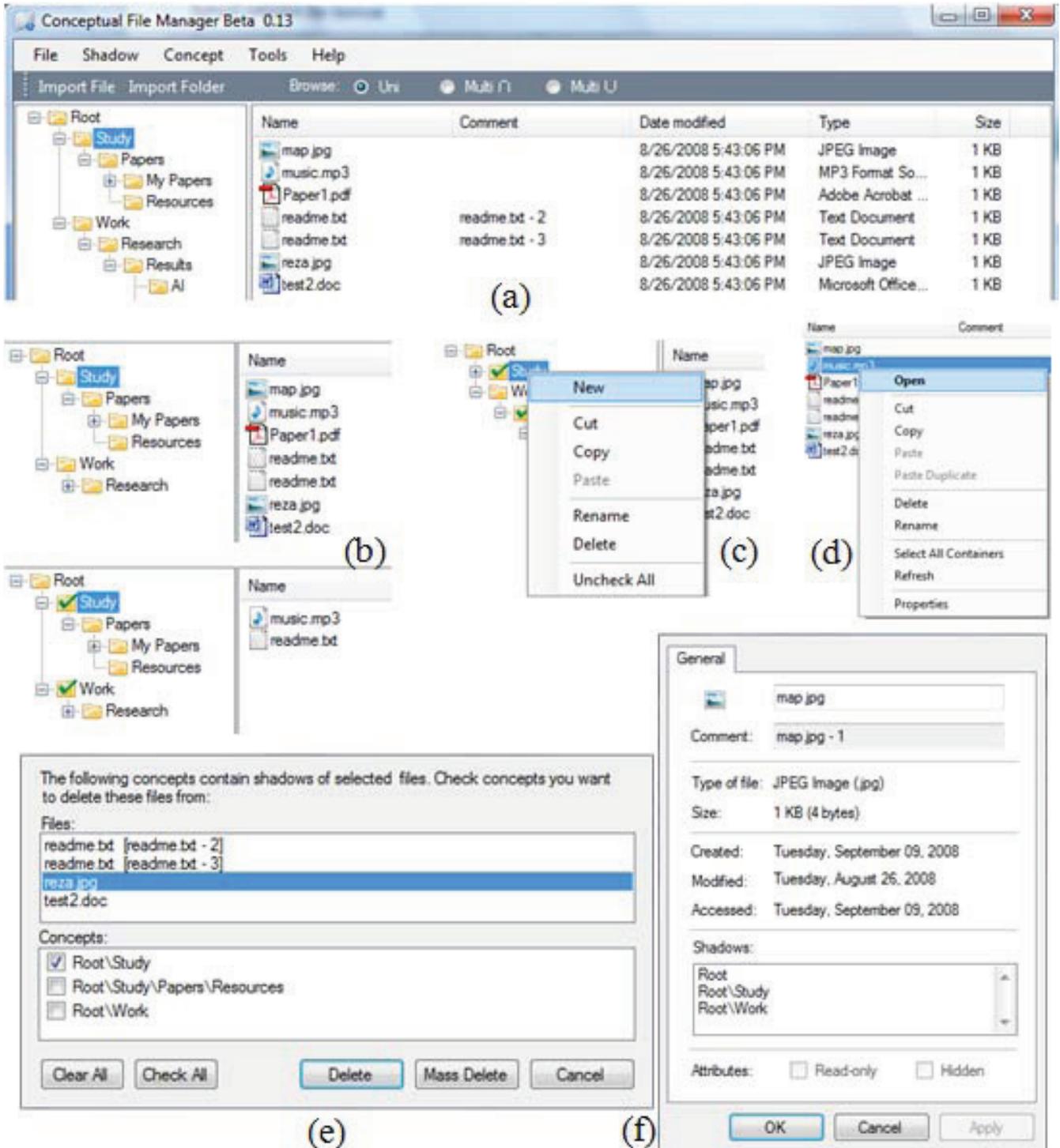


Figure 1. Different parts of the CFM program; (a) CFM environment. (b) Comparing the results in "Uni" and "Multi" file browse modes. (c) Concept-Tree Pop-up Menu. (d) Shadow Pop-up Menu. (e) Shadow Delete dialog. (f) Shadow Properties dialog.

clipboard to selected concepts.

- **Paste Duplicate:** this item will be enabled when a shadow is copied into clipboard (it is disabled when a shadow is cut). It will make a new duplication of copied shadows and the respected file in selected (highlighted) concepts or all the ticked concepts based on “file browse mode”, (“Uni”, “Multi  $\cap$  “ and “Multi  $\cup$  “).
- **Delete:** is used to delete selected shadows. The selected files are shown in the upper list box (see Fig. 1.e). In this form, by selecting a file in the first list box all the related concepts (that the file is a member of) are shown in the second list box. The user is then asked which shadows of each file should be deleted. By selecting the shadows, the user can select which one to be deleted and which one to be kept. There are also options for selecting all instances of a file or deleting all the selected files and their shadows in the system. In the latter case, the physical file will also be deleted.
- **Rename:** renames selected file; it is obvious that all the shadows of the file are renamed.
- **Select All Containers:** the icons of all concepts whose the selected file is a member of will be checked after clearing (i.e. resetting) the check marks of previously checked concepts.
- **Refresh:** refreshes file panel.
- **Properties:** It is similar to properties of files in Microsoft Windows, but there are also some extra information. The name and comment of the shadow are shown at the top of the form. Following that, type, size and dates related to the physical file are seen. There is also a useful list box containing all shadows pointing to

Figure 4- The average access times using CFM and folder paradigms for 19 test cases per user

the selected file. At the end of the form, user can find information about attributes of the physical file such as being read-only or hidden (see Fig. 1.f).

#### IV. CONCLUSION AND FUTURE WORKS

A file manager has been presented to gain a flexible, flat visualization of the file system supporting multiple categorization.

The advantages of CFM are described (but not limited) as:

- Clarity of the system; the user can save a file in the system with as much access methods (paths) as he/she wants. Because in many cases, especially for the files that are dealt with more frequently, there are several supposed paths (usually two to five or even more).
- The system offers less ambiguity in both storage and retrieval.

- There will be no redundant items while the user places a file in several concepts and several ways are devised to access the file (i.e., the several appropriate paths through concepts).
- Quick access to the files (i.e., finding the files in the first attempt).
- Size reduction (due to eliminating redundancy).

Copying a file in several concepts may take some more time, but it benefits retrieval enhancements in both eliminating retrieval time and failure. Using the manner of this paper, the user keeps both benefits of the manual hierarchical organization in nowadays file systems and multiple categorization simultaneously.

The CFM approach is also applicable in other classification based structures such as web mail management, site maps, text categorization and taxonomies that deal with information retrieval.

Finally, An extensive experiment on the mentioned benefits of CFM would be appropriate after adding some built-in capabilities to the most commonly used Windows applications to support CFM (e.g. a “save as” dialog for MS-Word to save a file into several paths).

#### REFERENCES

- [1] [1] Quan, D., Bakshi, K., Huynh, D., and Karger, D. R. (2003) User Interfaces for Supporting Multiple Categorizations. Proceedings of INTERACT 2003, 228-235.
- [2] [2] Lansdale, M., The Psychology of Personal Information Management, *Applied Ergonomics*, 1988, 19(1), 55–66.
- [3] [3] Barreau, D. and Nardi, B., Finding and Reminding: File Organization from the Desktop, *SIGCHI Bulletin*, 1995, 27(3), 39–43.
- [4] [4] Karger, D. R. and Quan, D., (2004), Haystack: a user interface for creating, browsing, and organizing arbitrary semistructured information. In *CHI '04: CHI '04 extended abstracts on Human factors in computing systems*, New York, NY, USA, 2004. ACM Press, 777–778
- [5] [5] Dourish, P., Edwards, W. K., LaMarca, A. and Salisbury, M. Presto: An Experimental Architecture for Fluid Interactive Document Spaces. *ACM Trans. Comput.-Hum. Interact.*, (2):133–161, 1999.
- [6] [6] Tan, D. S., Meyers, B. and Czerwinski, M., (2004), Wincuts: manipulating arbitrary window regions for more effective use of screen space. In *CHI '04: CHI '04 extended abstracts on Human factors in computing systems*, 1525–1528.
- [7] [7] Boardman, R., Spence, R. and Sasse, M. A. Too many hierarchies? The daily struggle for control of the workspace. In Proc. HCI International 2003, 2003.
- [8] [8] Google, Inc. Google Desktop. <http://desktop.google.com/>, 2004.
- [9] [9] Apple, Inc. Mac OS X Spotlight. <http://www.apple.com/macosx/features/spotlight/>, 2004.
- [10] [10] Cutrell, E., Robbins, D. C, Dumais, S. T, Sarin, R., (2006), Fast, Flexible Filtering with *Phlat* — Personal Search and Organization Made Easy. CHI 2006 Proceedings, April 22–27, 2006, Montréal, Québec, Canada.
- [11] [11] D. chiara, R.,Erra ,U.,Scarano ,V, "VENNFS: A Venn-Diagram File Manager," Seventh International Conference on Information Visualization, pp. 120. IEEE Computer Society Washington, DC, 2003.
- [12] [12] Shortcut, <http://kb.iu.edu/data/abhm.html>. 2009
- [13] [13] File managers, [http://en.wikipedia.org/wiki/Comparison\\_of\\_file\\_managers](http://en.wikipedia.org/wiki/Comparison_of_file_managers)
- [14] [14] [http://en.wikipedia.org/wiki/Hard\\_link](http://en.wikipedia.org/wiki/Hard_link)

# TSRR: A Software Resource Repository for Trustworthiness Resource Management and Reuse

Junfeng Zhao, Bing Xie, Yasha Wang

Key Laboratory of High Confidence Software  
Technologies, Ministry of Education  
School of Electronics Engineering and Computer Science,  
Peking University  
Beijing, China

Yongjun XU

Digital China Limited  
Beijing, China

*Abstract—Software reuse is a key technology to improve software quality and software productivity. Software resource repositories, which provide the management mechanism for software resources, are one of the infrastructures of software reuse. The existence of abundant software resources in software resource repositories provides possibility for successful software reuse. In the mean time, assuring the quality of software resources is a critical point to keep the confidence of users to reuse software resources. In this paper we present TSRR, a software resource management system that provide not only effective software resource management but also a software resource search engine on the Internet and trustworthiness management for software resources. The search engine makes TSRR acquire different types of software resources and organize these resources for better retrieval. The trustworthiness management, which includes evidence gathering, evidence trust management and trustworthiness evaluation, provides a mechanism for users to use trustworthy resources. The case study shows that TSRR can effectively help user to select software resources.*

*Keywords—component; formatting; style; styling; insert (key words)*

## I. INTRODUCTION

Software reuse is a key technology to improve software quality and software productivity. Software resource repositories, which provide the management mechanism of software resources, are one of the infrastructures of software reuse. A software resource is, broadly speaking, any cohesive collection of artifacts that solve a specific problem or set of problems encountered in the software development lifecycle. It may be any work-products from the software development lifecycle or software related activities, such as components, patterns, tools, Web Services, frameworks, solutions, documents, test cases or scripts, and so on.

The existence of abundant software resources in software repositories is a key factor for successful software reuse. Currently, user submission is the main way to enrich a software resource repository. However, some users may be reluctant to submit software resources, and therefore user submission may not be a reliable way for a software resource repository to get abundant resources. So a more convenient and automatic way to collect software resources on the Internet is needed. On the other hand, existing software resource repositories lacks mechanisms of guaranteeing the quality of software resources

they provide. This will impact the confidence of users to reuse software resources. Hence, it becomes important for software resource repositories to find a way to assure users that the provided software resources are trustworthy.

To address these two problems and provide developers a more convenient and trustable platform for resource reuse, based on the Trustie (Trust integrated environment) platform, we have developed the Trustie Software Resource Repository (abbreviated as TSRR). TSRR provides a mechanism to describe, collect, evaluate, classify and manage trustworthiness of software resources, to support trustworthy software development. It cannot only support software reuse process but also provide a resource sharing platform among projects. In order to enrich resources in TSRR, we developed a software resource search engine on the Internet. In addition, TSRR provide mechanisms to manage trustworthy software resources and manage their evaluation information. Thus it can make developers and end-users to make full use of software resources to build their own applications with good quality. TSRR is an open system to support the whole process of software development using existing software resources. It provides web service APIs, which can support SOA-based application. The implementation of TSRR is based on JO2nAS[1] and MySQL[2 ], so it makes TSRR an open repository to support software reuse. Moreover TSRR provides an Eclipse plug-in, which can integrate with software development environment to support better software reuse and sharing.

## II. RELATED WORK

Besides our TSRR, there are other software resource repositories like REBOOT [3], Agora [4], CodeBroker [5], OSOR.EU [6], SourceForge.Net [7], Component-Source [8], and Download.com [9]. REBOOT (Reuse Based on Object Oriented Techniques) is a famous repository in 90's. It aims to push the research and development of software reuse. It is comprised by a library to store components and a series of tools to support component publishing, retrieval, classification, selection and evaluation. Agora is a component search engine developed by CMU. It is designed for searching components (like JavaBeans, ActiveX, CORBA etc.) on the Internet. CodeBroker is a repository prototype to realize seamless integration between a repository and a code editing tool, and thus it can provide active inquire services. CodeBroker mainly

stores Java classes to support Java related software development. The OSOR.EU project has implemented a repository and a collaborative development environment. It aims to provide a library where software, documentation and knowledge can be easily accessible according to a specific taxonomy. Component-Source, SourceForge.Net and Download.com are all commercial software resource repositories that provide business component trading through their web sites. Although all of these software resource repositories address the problem of the management of software resource, only TSRR supports the management of trustworthiness for software resources. Moreover it provides a software resource search engine on the Internet, which can make TSRR abound with different types of software resources.

### III. THE FRAMEWORK OF TSRR

TSRR aims to provide a software resource management mechanism and a software resources sharing environment. It provides the following functionalities:

- Software resources acquisition, organization and management on the Internet.
- Mechanisms to describe, collect, evaluate, classify and manage software resources' trustworthiness, to support trust software development.
- A platform and a series of APIs to support software reuse especially support SOA-based software development.

Fig. 1 depicts the framework of TSRR. It has three layers: the storage layer, the function layer and the interface layer.

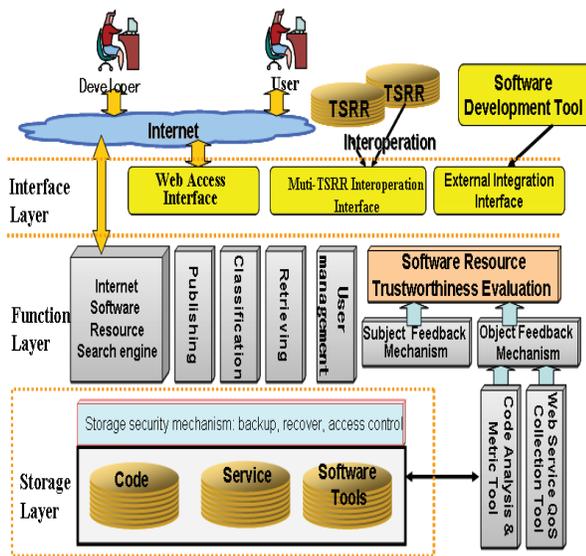


Figure 1. The framework of TSRR

- The bottom layer is the storage layer to store information of software resources including code, services, software tools etc. It provides a series of storage security mechanisms, such as backup, recover, access control etc. The storage layer also stores the information of all kinds of evidence to evaluate the

trustworthiness of software resources. The evidence can be submitted by users or some tools, such as source code analysis tools and Web Service QoS (Quality of Service) acquiring tools.

- The middle layer is the function layer to provide the management for trustworthy software resources. The Internet software resource search engine is for a wide range search of different types of software resources and related information. Part of the results can be published to the TSRR as a software resource, the others that have a close relationship with the resources, such as documents, feedbacks etc., can be saved as evidence for trustworthiness evaluation. The publishing mechanism provides the function for publishing resource descriptions, trust evidence, resource entities etc. In order to improve retrieval accuracy, the classification module is for publishers to describe the resource accurately using keyword classification, facet classification, enumeration classification and other classification methods. The software resource retrieval mechanism provides different ways to find software resources that meet user requirements. For example, it can support users to search resources using different trustworthiness criteria. The user management module is to manage all the information about users including access control information. The trustworthiness evaluation for software resources aims to give an assessment of whether the resource is trustworthy enough for users to use according to the evidence submitted by users or collected at run-time (such as Web Service QoS). The results of trustworthiness evaluation help users to select appropriate software resources.
- The interface layer provides different access interfaces for users to publish, retrieve, classify, evaluate and manage software resources. User cannot only access the repository via the web but also through Web Service APIs. Thus the repository can be integrated with other software development tools or platforms.

### IV. TSRR

TSRR provides a management mechanism to support different software resources, such as source code, web services and documents. Furthermore, it provides a software resource search engine on the Internet and trustworthy software resource management to improve the quantity and the quality of the software resources in TSRR. We present them separately in the following subsections.

#### A. Software resource search engine on the Internet

In order to provide a large number of software resources for developers, we developed a software resource search engine to collect software resources. Currently, there are a great number of software resources available on the Internet. However, these resources are not well organized and managed, which makes developers spend a lot of time to acquire their desired resources. The software resource search engine can harvest, organize software resources on the Internet and make them well

organized for retrieval. The framework of the software resource search engine is depicted in Fig. 2.

In the resource harvesting phase, we proposed a resources harvesting mechanism which relies on both the web search engine as well as spiders that concentrate on specific sites. Based on this mechanism, many resources and related information can be obtained. We define a series of file formats to clean the redundant information. In addition to this, we build a feedback module to collect user information that could help us to extract useful information. Relied on the information obtained, we build an extensible model to describe both the resources and the related information. This model can be extended based on different scenarios.

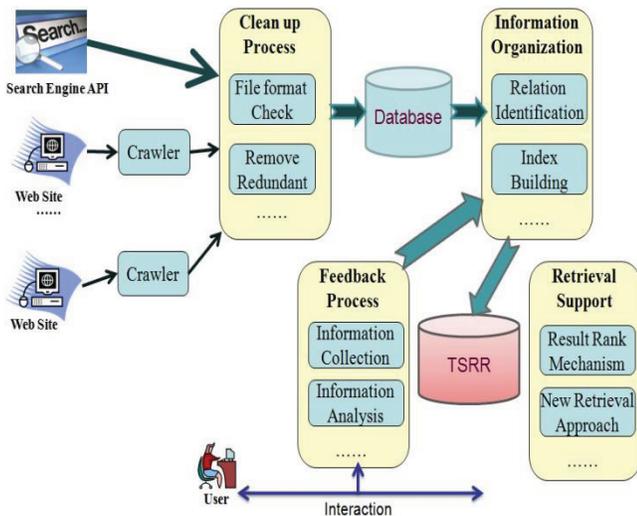


Figure 2. The framework of the software resource search engine

In the resource organization phase, we use a group of algorithms that can identify association relationships among different kinds of information, including the example code recommending algorithm, the similar resource discovery algorithm, the algorithm that links the resources with their developers, and the algorithm that discovers similar texts etc. we provide retrieval support to consider result ranking and to improve the precision and the recall. The example code recommending algorithm extracts related code from the obtained resources and then clusters the code based on their usage of the component and finally ranks the clustered results to provide examples for developers. The similar component discovery algorithm leverages the cooperation relationship among components and makes use of the LSA technique to calculate the similarity between two components.

By doing this, we can greatly reduce the efforts for developers in acquiring desired components and improve the efficiency of software reuse.

#### B. 4.2 Trustworthy software resource management

Trustworthy software resource management comprises four parts: evidence model customization, evidence gathering, evidence trustworthiness management, and trustworthiness

evaluation and classification. Fig. 3 depicts trustworthy software resource management in TSRR.

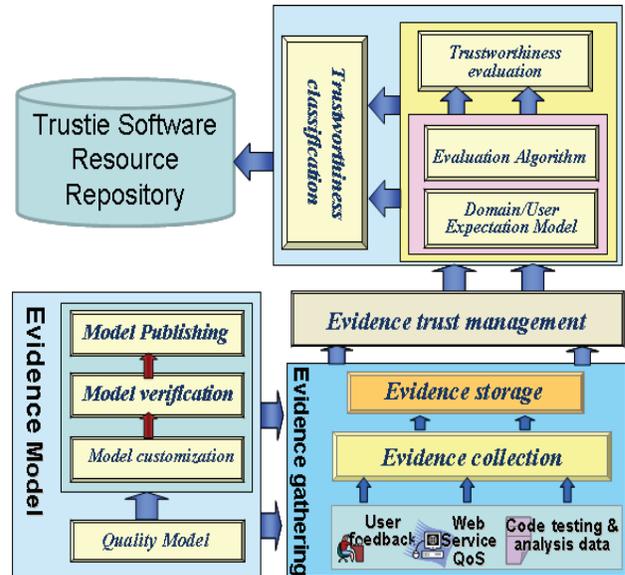


Figure 3. Trustworthy software resource management in TSRR

Evidence model customization is the basis for evidence gathering. It determines what is needed to collect and organize the evidence. As shown in figure 4, the model is a hierarchical model like the 9126 quality model and each node is linked with an evidence type to indicate the source of evidence. The evidence type may be user feedback, resource test data, code analysis data, or Web Service QoS, etc. Considering there may be several evidence models and each model may have the properties of the same meaning, we build synonymous relationships between evidence model properties. Thus it makes full use of evidence and avoids repeated evidence collection. The evidence gathering module collects and stores various kinds of evidence for trustworthy software resource evaluation.

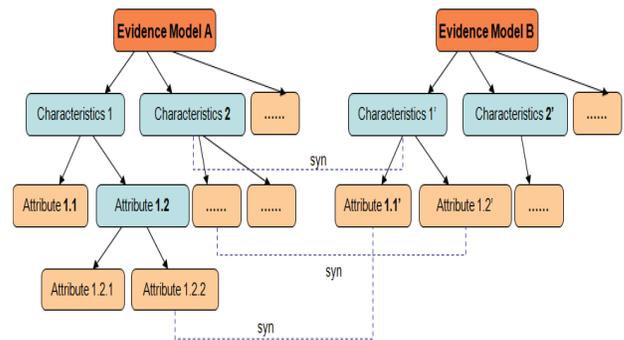


Figure 4. Evidence model with synonymous relationship

Through the establishment of trustworthy relationships between users and evidence submitters, the evidence trustworthiness management module deals with the situation that the collected evidence is false or inaccurate. If a user feels that one piece of subjective evidence (such as the user feedback)

cannot be trusted, he or she can give a percentage degree to this evidence. This degree is the possibility of the evidence that can be considered in trustworthiness evaluation. It should be noted that users cannot give a degree to objective evidence like Web Service QoS.

Trustworthiness evaluation and classification is the core of trustworthy software resource management. It uses gathered evidence and the user-defined expectation model to evaluate the trustworthiness of software resources. As there may be different requirements and restrictions of specific domains, the expectation model is defined by users to describe their expectation of software resources. The expectation model is also a hierarchical model to describe user requirements. We also build mapping from each user expectation model to one or more evidence model. The mapping defines two things. One thing is to define the inclusion relation between properties in user expectation and evidence models. The relation means that one user-expected property may be calculated or derived by one or more properties in an evidence model. For example, the security can be seen as the summation of availability, integrity and confidentiality. The other is to define the weight degree of each relation. The degree represents the importance that user considered. The value of the degree is between 0 and 1, and the sum total is 1. In order to provide a simple and convenient way to tell users which resource is better and trustable, we establish a software trustworthiness classification specification to classify resources. Now, we divide software into five levels (from 1 to 5, the higher means higher trustworthiness) and each resource in the repository is marked with a trustability level according to the results of trustworthiness evaluation.

Considering some evidence is objective like user satisfactory, we adopt fuzzy comprehensive evaluation method to decide the trustability level of the resource. The first step is to normalize the value of every characteristic and attribute in evidence model. Then we create factor set, evaluation set and the weight set to compute the trustability level. The factor set is indicate every quality factor that the resource related, the value of each can be get from the evidence that collect by TSRR. The evaluation set is a judge set from 1 to 5 to indicate the trustability level of resource. The weight set is the weight distribution of each element in factor set that defined by domain/user expectation model. Then we can get a fuzzy matrix to compute the result and give the user recommendations on which one is proper to use.

By doing this, we can greatly reduce the efforts for developers in acquiring desired resources and improve the efficiency and quality of reuse.

### V. 5. CASE STUDY

The scenario of the case is: A service user needs a service to look up information of a book by its ISBN. The user wants the service has trustability level 4 and the degree for availability, response time and satisfaction is 0.4, 0.4, and 0.2. TSRR has already found there are six web services have the similar function of what user required. In addition to this, TSRR has collect these services' QoS data and user evolution information by evidence collection tools. Table 1 shows one of the evidence of these service and we can see their quality are

different. The question is: which one should be the proper one to meet with user's needs?

TABLE I. THE INFORMATION AND QoS OF SIX ISBN WEB SERVICES

ID	Service Provider	Service access address	Availability	Response Time	Satisfaction
WS_ISBN_1	daehosting.com	http://webservices.daehosting.com/services/isbnservice.wso	91.10%	840ms	92.5%
WS_ISBN_2	webservice.x.com	http://www.webservice.x.com/isbn.asmx?WSDL	75.5%	810ms	89.5%
WS_ISBN_3	booksprice.com	http://www.booksprice.com/IsbnConverter.jws?wsdl	99.11%	1060ms	87.5%
WS_ISBN_4	wou.edu	http://student-wou.edu/rwessel/CHAPTER18/ISBN.asmx?WSDL	98.81%	1900ms	85.5%
WS_ISBN_5	xmlme.com	http://www.xmlme.com/WAmazonBox.asmx?wsdl	99.7%	285ms	91.5%
WS_ISBN_6	pickabook.co.uk	http://services.pickabook.co.uk/service.asmx?WSDL	99.72%	368ms	93.7%

Using the way depicted in previous sections, we can get user expectation model and evidence model with weight relationship as figure 5. Then we can use the fuzzy comprehensive evaluation to compute the trustability level of these services. At first we use the Gaussian normalization method to normalize the evidence data. The factor set is made by the value of the evidence, that is  $U = \{Availability, Response Time, Satisfaction\}$ . The evaluation set is  $V = \{1,2,3,4,5\}$  and the weight set is  $A = \{0.4,0.4,0.2\}$ . After that we get a fuzzy matrix to compute each web service's trustability level. The result we get is WS\_ISBN\_5 and WS\_ISBN\_6 are level 4, WS\_ISBN\_1 is level 3, the others are level 2. So we can recommend WS\_ISBN\_5 and WS\_ISBN\_6 to user.

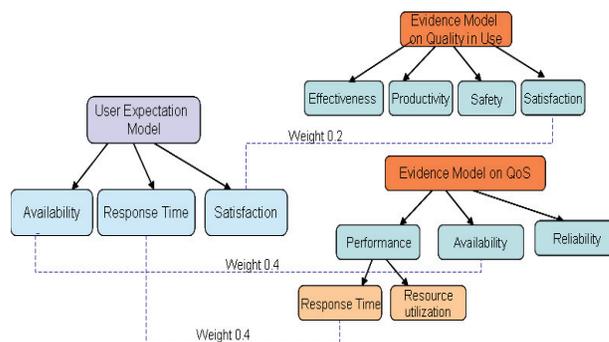


Figure 5. User Expectation model and Evidence model with weight relationship

### VI. CONCLUSION

This paper introduces a software resource repository to support software reuse and sharing. The repository provides publishing, retrieving, classifying, evaluating and managing

functions to support trustworthy software development. Moreover, it provides an Internet search engine to find more resources to be reused. The case study shows the TSRR effectively help users to select what they want. The future work is to study an automatic way and a more accurate method to calculate the trustability level of software resources.

#### ACKNOWLEDGEMENT

This work is supported by the National Basic Research Program of China under Grant No. 2009CB320703, the High-Tech Research and Development Program of China under Grant No. 2007AA010301, 2009AA010307, the Science Fund for Creative Research Groups of China under Grant No. 60821003, the National Natural Science Foundation of China under Grant No. 60803011, 60803010.

#### REFERENCES

- [1] <http://www.ow2.org/view/ActivitiesDashboard/JOnAS>
- [2] <http://www.mysql.com>
- [3] J. M. Morel, J. Faget, The REBOOT Environment, BULL S. A. Rue Jean JAURES, F-78340 LESCLAYES-SOUS-BIOS, France.
- [4] R. C. Seacord, S. A. Hissam, K. C. Wallnau: Agoro-a search engine for component, IEEE Internet Computing, November/December, 1998, pp. 62-70.
- [5] Y. Ye, An Active and Adaptive Reuse Repository System, Proceedings of 34th Hawaii International Conference on System Sciences (HICSS-34), Software Technology Track , Maui, HI, IEEE Press, 2001.
- [6] <http://www.openworldforum.org>
- [7] <http://sourceforge.net/>
- [8] <http://www.componentsource.com/index.html>
- [9] <http://download.cnet.com/windows/>

# A Log-Assisted Approach Enforcing Consistency in the Presence of Exceptions

Nikolas Nehmer  
University of Kaiserslautern, Germany  
Department of Computer Science  
nnehmer@cs.uni-kl.de

## Abstract

*In today's software development processes, exception handling is still considered as an issue of minor importance although, if not handled appropriately, exceptions may seriously harm a software system. In particular, the premature exit of a method due to an exception might leave objects of the system in an inconsistent state. Subsequent calls to those objects will process the inconsistent data and the error will propagate throughout the system. In this paper a novel approach to systematic exception handling in single-threaded applications is presented. A generic runtime mechanism supporting developers in ensuring consistency and fault containment in the presence of exceptions is proposed. Exception related inconsistencies are identified and rolled back during runtime automatically, substantially reducing the harmful impact of exceptions not expected by the developer. Lightweight application logging is used to free developers of the burden to reestablish consistency in case of exceptions manually.*

## 1 Introduction

Due to the advancing use of software there is a strong need for software systems to meet dependability-related requirements. Unfortunately, the presence of errors [1] is a fact of life in today's complex software systems that needs improved methods to deal with.

The development of software tolerating these errors at runtime is one way to achieve improved robustness. Modern programming languages such as C#, C++ or Java provide explicit exception handling<sup>1</sup> programming language constructs as a mechanism for tolerating errors at runtime.

---

<sup>1</sup>Exception handling is a programming language construct or computer hardware mechanism designed to handle the occurrence of conditions that change the normal flow of program-execution. The condition raising the control-flow change is called an exception. In general exceptions are used for signaling error (exceptional) conditions.

However, due to the lack of execution history information error-handling support based on traditional programming language constructs for exception handling alone is insufficient. The premature termination of a method due to an exception might leave the system in an inconsistent state. Subsequent calls to a failed component or retries may process inconsistent data and fail again or silently propagate the error throughout the system. Reestablishing consistency is completely left to the application developer who is responsible for all necessary exception handling related house-keeping (i.e. clean-up actions reestablishing consistency). Applying standard exception handling strategies like retrying the failed method on possibly inconsistent system state seems to be unacceptable.

In this paper a runtime mechanism automatically ensuring consistency in the presence of exceptions is proposed. Software developers are freed of the burden to reestablish consistency for methods left unfinished due to an exception. Lightweight application logging is used to automatically roll back inconsistent system state. The approach exploits isolation guarantees given by single-threaded applications resulting in a lean logging strategy. Based on these mechanisms the harmful impact of unexpected exceptions is reduced significantly. The system model, examples and approach presented in this paper are based on the Java programming model but can easily be adapted to other program languages.

## 2 Exception-induced inconsistencies

In general, exception handling mechanisms are intended to make developers design and build more robust and dependable software systems by separating code regions that handle unusual system behavior from code that deals with normal processing. This separation of concerns leads to consequences for system design. Exceptions are raised (*thrown*) when either semantic constraints are violated or fatal error conditions occur. This causes a control flow change from the instruction that raised the exception to a point where the exception is *caught* and handled (i.e. the

exception handler). In object-oriented systems, exceptions are not necessarily handled within the method of their occurrence. If no appropriate exception handler (according to the exception's type) is found the current method scope is terminated and the exception is implicitly propagated to the caller or recursively up along the call stack. Methods terminated exceptionally might leave an inconsistent system state.

```

1 class DataStructure {
2     private int size;
3     ...
4     public void add(Object o) {
5         size++;
6         ... = new Node(o);
7     }
8     ...
9 }
10 class Node {
11     ...
12     Node(Object o) {
13         if(o == null) {
14             throw new IllegalArgumentException();
15         }
16         ...
17     }
18 }
19 class Demo {
20     main(String[] args) {
21         DataStructure d = new DataStructure();
22         Object obj = getObject(); //might
23             return null
24     try {
25         d.add(obj);
26     } catch (Exception e) {
27         //DataStructure's private size should
28             be decremented
29     }
30 }

```

Listing 1. Simple example

Listing 1 shows a data structure adding and removing object references to an internal node structure (e.g. stack or singly linked list). Null-references are not allowed and therefore a node's constructor throws an `IllegalArgumentException` if a null-argument is passed. When `add()` is called the internal size counter representing the number of elements in the data structure is incremented (line 5) before the node instantiation statement (potentially throwing an exception) is executed (line 12). If the instantiation statement fails the size variable becomes inconsistent. To reestablish consistency by decrementing the private size field in the demo caller's error handling code, internal class details would have to be revealed

to the caller violating information hiding principles. Furthermore, reestablishing the consistency of system state in more complex scenarios gets unmanageable for the system developer.

The current system state of an object-oriented system is defined by the state of all reachable objects, the stack and the instruction pointer referencing the current program instruction at a given point in time. An object is an instance of a class and its state is determined by its instance variables. Local variables are part of the stack. Class variables (also called static fields) belong to the class. Variables either hold primitive data values or they reference objects. A reachable object is defined as an object referenced by class variables, instance variables or local variables from a different reachable object. The current *overall system state* can be illustrated using the notion of an *object graph*.

**Definition 1** An *object graph*  $g$  of an object-oriented system  $s$  represents a snapshot of the systems's overall state at a particular point in time. Nodes in  $g$  represent objects or values of primitive data types. Edges either embody references to values of a primitive data type or non-null references to objects representing the current values of instance, class and local variables of an object at a given point in time. This definition explicitly includes local variables only existing in a system's stack frame represented by a subgraph of  $g$ .

A method in an object-oriented system is a block of code. Assuming a correct realization and initialization of the system, each method call will begin in a consistent system state and leave the system in a consistent state after successful completion. However, if a method execution is terminated by an exception, an inconsistent system state potentially results. The same applies to try-blocks. To formalize the system states before executing a block of code and after returning from execution the notion of a *pre- and post-execution object graph* is introduced (related to [2]).

**Definition 2** Let  $c$  be a class, and  $b$  a block of code (including inlined method calls and try-blocks) within class  $c$ . Given an invocation of code block  $b$  on an object  $o$ , the *pre-execution object graph* for that invocation of  $b$  is defined as the object graph taken just before  $b$  is executed. The *post-execution object graph* for an invocation of  $b$  that returns with an exception is defined similarly, except that the graph is taken just after  $b$  returns with an exception, the current stack frame is already relieved and local variables are discarded.

Developers might want to allow differences between the pre- and post-execution object graph for some cases. For example, a counter variable should log every access to a method, even the ones terminated by an exception. Therefore, the field has to be explicitly tagged by the developer

(e.g. using annotations). Based on the notion of pre- and post-execution object graphs we distinguish *inconsistency* from *potential inconsistency*.

**Definition 3** A system state is defined to be **potentially inconsistent** if a block of code was terminated by an exception and the pre- and post-execution graphs differ. A system state is **inconsistent** if it is potentially inconsistent and at least one of the nodes or edges that changed between the pre- and the post-execution object graph is not explicitly tagged to be allowed to change by the developer.

As derived from the example in listing 1 a simple increment of a counter variable caused an inconsistency in case of an exceptional method termination. Of course, the example can be easily fixed by reordering the increment statement and the node instantiation instruction as shown in listing 2. The `add()` method is now considered to be *failure atomic* [2].

**Definition 4** A block of code *b* of class *c* is **failure atomic** if for all executions of *b* that return with an exception, the pre-invocation and post-invocation system state is equal. A method is *failure non-atomic* if it is not failure atomic.

```
1 class DataStructure {
2     private int size;
3     ...
4     public void add(Object o) {
5         new Node(o);
6         size++;
7     }
8     public void addAll(Object [] oa) {
9         for (x = 0; x++; x < oa.size()) {
10            add(oa[x]);
11        }
12    }
13    ...
14 }
```

**Listing 2. Composability example**

Systems built by following object-oriented design rules are usually created by composing lower level modules. Several calls to methods delivering basic functionality are composed building a new functional module.

**Definition 5** An architecture is considered to be **composable** with respect to a system property *p* if integrating subsystems that fulfill *p* will not invalidate *p* for the overall system.

Composing several failure atomic methods into a new functional module does not necessarily enforce the resulting method to also be failure atomic. The example in listing 2 illustrates the lack of composability (see

definition 5) with respect to the failure atomicity property. In the example several calls to `add()` are composed to a new method `addAll(Object[])`. Imagine a call to `addAll(Object[])` passing an array of three entries with the third one being a null-reference. The call to `addAll(Object[])` will add the first two entries to the internal node structure but will raise an `IllegalArgumentException` with the last constructor call. The manipulated data structure now contains two more objects although the `addAll()` method terminated exceptionally. Writing exception handling code reestablishing consistency for this simple example would be rather complex. The exception handler would have to know how many objects were added to the data structure and remove them accordingly. Removing objects from the data structure might possibly raise additional exceptions forcing the developer to build nested exception handling code. This simple example clearly shows that manual exception handling trying to reestablish consistency becomes unmanageable soon.

Developing highly fault tolerant software implies writing bullet-proof exception handling code and being aware of every exception type possibly raised at every single instruction in an application. But reasoning about *runtime exceptions* such as `NullPointerExceptions` is extremely difficult and often impossible. Writing exception handling code for exceptions expected by the developer is already a complex task as shown above. Obviously, writing exception handlers for *unexpected exceptions* is impossible.

To summarize, there is a strong need for a generic system mechanism simplifying exception handling by supporting automatic rollback of inconsistent system state. Developers should be freed of the burden to reestablish consistency by hand, being able to concentrate on the control flow continuation logic itself. Based on the rollback semantics a default exception handling mechanism is needed to handle unexpected exceptional events.

### 3 Approach

To ensure consistency in the presence of exceptions a generic system mechanism realizing automatic rollback semantics is proposed. Changes to the system state made within a terminated block of code (e.g. a method) have to be rolled back at least to the beginning of that block. Consequently, the system state represented by the pre-invocation object graph of that block has to be restored (except for variable values explicitly excluded by the developer).

One of the main goals of the mechanism is *simplicity*. Developers should not be bothered with additional programming constructs to enforce automatic consistency control. The whole concept should *transparently* integrate into today's programming environments. Furthermore the

approach should be *scalable* – i.e. applicable to varying scopes. The system part guarded by the mechanism should not be limited to a single data structure. Furthermore, the mechanisms should not be restricted to single system parts explicitly denoted by programming language constructs (e.g. introducing a new keyword to tag blocks with “all or nothing” semantics). Rather program and architectural structures that are present anyway should be exploited. As a vision according to the concept of spheres of control [3] a complete application is considered to become transactional.

Even if a complete application was transactional the question remains how far the application should be rolled back and where the control flow should be continued in case of an exception. Assuming an instruction within some method fails, should the method be rolled back and retried automatically? What happens if the retry fails - retry again or propagate the exception to the caller that is rolled back again? Even if this process could be automated to some degree more sophisticated exception handling strategies have to be applied explicitly. After rolling back some block of code, execution has to be continued. Even if rolled back, a retry does not necessarily work. To being able to structure the rollback boundaries and control flow continuation behavior *atomic blocks* (see definition 6) have to be defined as transaction boundaries. In addition to reestablishing consistency by automatic rollback, explicit exception handling strategies (control flow continuation) have to be defined at the transaction boundaries.

**Definition 6** An *atomic block* is a block of code that is guaranteed to either execute completely or terminate exceptionally. In case of an exception the system state is rolled back to the state embodied by the pre-execution object graph of that block (except variables explicitly excluded). An atomic block embodies the “all-or-nothing” principle.

To ensure the *transparency* of the approach the atomic blocks are derived from the program code structure itself and from architectural properties. At every transaction boundary a reasonable handling strategy has to be defined in addition to an automatic rollback behavior, i.e. simple standard strategies such as retry or propagate, or more sophisticated application specific exception handlers. Consequently, the scope of a transaction should not be too wide but not too narrow and fit an application’s logical structure. Try-blocks are perfect matches. At these natural boundaries exception handling strategies are already applied and applying them to data structures ensured to be consistent (by automatic rollback) is perfectly reasonable. On an architectural level component boundaries seem to be perfect transaction boundaries. Components form self-contained functional modules that perfectly fit to the proposed “all or nothing” semantics. They can be automatically derived e.g.

from UML-diagrams. To make components apparent on the code level and detectable by a runtime system mechanism their interface methods are simply tagged by an annotation. Try-blocks and components embody atomic blocks in our approach.

In general, exceptions should be handled as local as possible. Due to the information hiding principle, the further an exception is propagated up along the call stack the more context information is getting lost. In particular, a low level exception should only be signaled to a calling component in well justified cases. Otherwise component internals are revealed to the caller violating information hiding principles. In most cases a caller will not be able to recover from a low level exception anyway.

Consequently, exceptions not handled appropriately within a component are wrapped into a standard `ComponentFailedException` automatically and the component state is rolled back by the underlying runtime system. To allow low level exceptions to pass the component boundary they have to be declared at the component interface explicitly. In general, exceptions that are able to pass component boundaries should be part of the contract between caller and callee. Therefore every exception type possibly signaled by a component has to be declared at the component interface. All exceptions not declared in the component interface are mapped to the standard `ComponentFailedException`. A calling component should guarantee to handle all exceptions declared in the component interface including the standardized `ComponentFailedException`. In Java this can be enforced by the compiler by using checked exceptions crossing component boundaries. Based on that mechanism, unexpected and therefore most likely uncaught exceptions can be handled quite elegantly ensuring component consistency (by automatic rollback) for subsequent calls and simplifying exception handling among components (see simplified example in listing 3).

From a conceptual point of view components are simply some blocks of code defined by their interface methods. Just like method calls, a call to a component, i.e. a call to one of the components interface methods, can be inlined by inlining the block of code. Consequently, components, methods and try-blocks are equally alike except their logical and structural meaning within a program. According to the program structure components and try-blocks form hierarchies of atomic blocks. Try-blocks are either nested into another or into a component. Components are nested into another by invoking each other. According to traditional nested database transactions [4] atomic blocks can be nested as well as defined below.

**Definition 7** A *nested atomic block* is an atomic block contained in an outer atomic block. Each nested atomic block can be rolled back individually.

```

1 ...
2     try {
3         componentInterface(Obj o);
4     } catch (ComponentFailedException e) {
5         // e.g. control flow continuation
6         //relying on consistent system state
7         doSomethingElse(Obj o);
8     }
9 ...
10
11 class MyComponent {
12
13     @ComponentInterface
14     public void componentInterface(Obj o)
15         throws ComponentFailedException {
16         //do something
17     }
18 }

```

**Listing 3. Component example**

Based on these explanations (simplicity, transparency, natural transaction boundaries, exception locality) the exact rollback semantic desired by a developer has to be defined. To ensure the simplicity of the approach, in case of an exception, nested atomic blocks terminated by an exception are rolled back successively until an atomic block with a catch-block corresponding to the exception type raised is found. This means, if an instruction within a try-block fails and the catch-clause handles the appropriate exception type, the system state is rolled back by the underlying system mechanism (to the system state depicted by the pre-execution object graph of the try-block) and control is transferred to the catch-block. If no matching catch-block is found, the exception is propagated up the call stack until either an appropriate handler is found or the component boundary is reached. In the former case, the outer atomic block is rolled back and control is transferred to the corresponding catch-block. In the latter case, the exception is caught automatically, wrapped into a standard `ComponentFailedException` and the underlying runtime ensures rollback of the inconsistent system state. If the contract between caller and callee is fulfilled, the exception is caught in the try-block surrounding the component call and control flow is continued. In fact, every time an exception is caught within a try-block the corresponding atomic block is rolled back. The developer writing the handling code and continuing the program control flow can rely on a consistent system state and fully concentrate on the control flow continuation logic. Figure 2 shows an example for nested atomic blocks.

To ensure rollback semantics based on atomic blocks information on former system state has to be recorded. Two

approaches are standing to reason – checkpointing and logging. Checkpoints in general take a complete snapshot of a recent system state to speed up the restart of a system or subsystem. At the entry point to every single atomic block a snapshot would have to be taken. Consequently, checkpointing would impose a huge and unacceptable overhead due to the frequency of taking snapshots induced by the structure of atomic blocks proposed in this paper. In contrast a logging approach records the history of instructions executed within an atomic block only. A naive approach would be to log a complete execution trace including every single instruction executed, i.e. every single read or write instruction, method call, object instantiation, branch or loop statement, exception throw and so forth imposing an unacceptable overhead.

Traditional transactions fulfill ACID-properties [4]. The approach presented in this paper assumes isolation guarantees (given by single-threaded applications) and focuses on forcing atomicity and consistency. Consequently, the log protocol can be highly optimized, significantly reducing the overall impact on system performance.

The goal is to achieve a minimal set of log entries still ensuring complete rollback capabilities to the entry points of atomic blocks – i.e. reestablishing the state defined by the pre-execution object graph of an atomic block. As a general rule of thumb only instructions changing the system state have to be logged at all. The current system state of an object-oriented system was already defined in chapter 2. In the first place the system state is defined by the values of all variables, i.e. object references and primitive values assigned to class variables, instance variables, method parameters or local variables. Consequently, variable assignment as a means to change the state of a variable is the only instruction type that has to be logged to reestablish a former system state.

Each log entry corresponds to a variable assignment. To ensure roll back semantics log entries have to be precisely assignable to their real world counterparts. Therefore a log entry contains a unique object identifier (or class identifier in case of a class variable) and identifier for the variable that was changed and of course the corresponding value. To further reduce the logging volume before-images can be taken – i.e. instead of storing the newly assigned value the old value is stored. Furthermore, always whole blocks of code have to be rolled back. In consequence, intermediate variable assignments can be ignored. Only for the first occurrence of a variable assignment on an object `o` within an atomic block a log entry has to be created.

Variable assignments perform state transitions modifying either local or global system state. The *local system state* is a subset of the *overall system state* (depicted by a system’s object graph) at a particular point in time. It simply comprises objects instantiated and primitive type variables

declared in the currently executing atomic block – i.e. the state only visible within the current atomic block. More formally the current *local system state* is defined by all nodes in the object graph  $g$  of a system  $s$  created in the scope of an atomic block. The *global system state* represents the system state only visible outside of the current atomic block. The global system state is embodied by the object graph excluding the subgraph representing the local system state.

According to these observations, the amount of instructions to be logged can be further reduced. Not every variable assignment statement has to be logged as some of them only affect the local system state. After unwinding the stack frame or leaving a closed code block (e.g. try-block) local objects and variables do not exist anymore or are detached from the object graph representing the overall system state. Therefore these variables will never be visible to the world outside of the current atomic block. Assignment statements to be logged affect the pre-execution object graph of the current atomic block – the others can be ignored as they are not necessarily needed to reestablish the system state before entering the atomic block. To distinguish variable assignment statements affecting global or local system state, it is essential to analyze whether a changed variable will still exist after terminating the current atomic block. To make that distinction the scope in which a variable was declared (i.e. was created and “lives” in) is the decisive factor. In general, only assignment statements within an atomic block have to be logged. In addition, variable assignments have to be treated differently according to the type of the variable. Instance variable assignments have to be logged if the corresponding object was created outside of the current atomic block. Assignments to local variables have to be logged if the variable was declared outside of the current atomic block. Class variables constitute an exceptional case and have to be logged always.

The decision tree depicted in figure 1 summarizes the properties that make a variable assignment essential to still ensure roll back semantics of an atomic block.

The minimal logging approach is illustrated by an example shown in figure 2. The example shows a Java application in the left column of the table. The subsequent column (assign) exposes the current system state at a given point in time showing current variable values while executing the program. The third column (former) depicts the before-image of a variable (i.e. the old value). The last column (log) represents the values to be logged according to the minimal logging approach.

## 4 Realization

A first prototype of the logging concept described in the previous chapter has been implemented and is almost ready for performance tests. In this chapter the overall implemen-

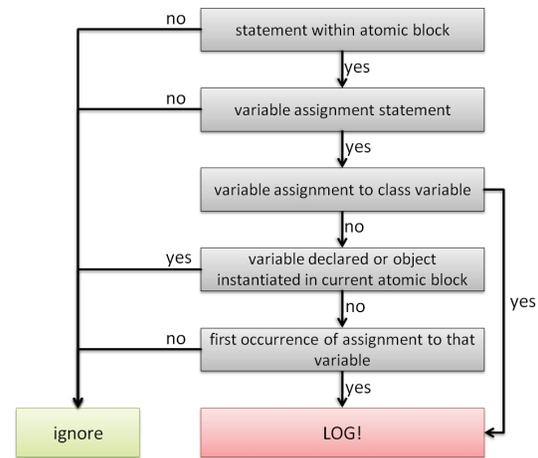


Figure 1. Decision tree

tation approach and architecture of this implementation is sketched briefly.

The logging mechanism is based on dynamically instrumenting Java bytecode using the ASM bytecode framework [5] and Java agent technology [6]. Before a class is loaded into the Java virtual machine (JVM) each bytecode statement to be logged is enhanced with calls to the logging facility. To minimize the time spent with the synchronous calls to the logging facility, the raw log data is queued in a data structure and control is immediately returned to continue the original control flow. The log data is dequeued and processed by a worker running in a separate thread of execution. The worker structures, potentially compresses and serializes the log data. The performance of this implementation benefits from multicore systems exploiting its inherent concurrency.

The implementation of the rollback mechanisms is based on exploiting the Java debugger API. The application is executed in debug mode and an exception breakpoint is defined listening to any kind of exceptional event. If an exception occurs the exception breakpoint is activated and the Java virtual machine is suspended, suspending all running threads and executing the rollback mechanism. To roll back the system state to the entry point of the current atomic block every variable assignment executed within that atomic block affecting the pre-execution object graph has to be revoked. Based on the log entries, the former system state is reestablished using access mechanisms provided by the Java debugger API. These access mechanisms provide direct read or write access to all objects present in the virtual machine. Accordingly, the API offers a simple way of modifying the variable values to recover the former consistent system state. After reestablishing consistency the virtual machine is resumed, the exception is propagated until it reaches an appropriate catch-block and the normal con-

Java code	assign	former	log
class SomeClass {			
int x;	-	-	-
SomeClass(int x) {this.x = x;}	this.x=5	-	-
main() {			
SomeClass s = new SomeClass(5);			
s.iMethod(new Other());			
}			
@ComponentInterface			
public void iMethod (Other obj) {			
Other o = obj;	obj=o	-	-
int y = x + 1;	y=6	-	-
x = localObj.sV;	this.x=7	this.x=5	this.x=5
x = y;	this.x=6	this.x=7	-
//nested atomic block			
try {			
y = y + 3;	this.y=9	y=6	y=6
o.sV = y;	o.sV=9	o.sV=7	o.sV=7
} catch (Exception e) {			
//do some exception handling			
} } }			
Class Other {static int sV=7;}			

Figure 2. Minimal logging example

control flow is continued. The complete prototype is embedded into the Eclipse [7] debugger. The overall architecture is illustrated in figure 3.

Performance penalties from this implementation concept result mainly from running the application in debug mode and maintaining an internal data structure mapping log entries to their object instances. They may be circumvented if the approach was embedded in a virtual machine like the JVM directly. A native JVM implementation would open even more performance optimization possibilities.

## 5 Related work

The general idea of automatically reestablishing consistency in the presence of exceptions has been around for several decades now. The original idea of rollback oriented consistency control was introduced by traditional database transactions [4, 8]. Stroustrup [9] introduced the notion of exception safety embodied in the C++ standard library without proposing a generic system mechanism applicable to all kinds of software modules. In [2] the notion of failure atomicity is introduced and a mechanism automatically detecting failure non-atomic methods is proposed. In most cases these methods are automatically turned into failure atomic methods by establishing wrappers taking snapshots of system states. This approach is not applicable dur-

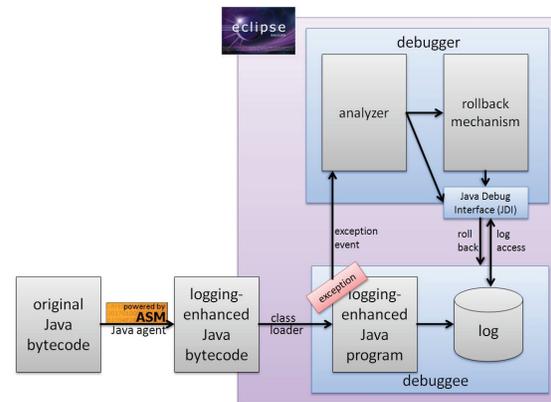


Figure 3. Proof of concept architecture

ing runtime. In recent years, software transactional memory (STM) [10, 11, 12] has been investigated as a concurrency control mechanism. As proposed in several papers [13, 14, 15], STM can be exploited to establish rollback semantics in the case of an exception. STM was originally designed as a concurrency control mechanism. Consequently, the isolation property of ACID transactions has to be maintained. This implies a huge overhead on the logging mechanism compared to a lightweight approach focusing on rollback semantics only. In the field of debugging, so called “back in time” debuggers (e.g. [16, 17, 18]) have been proposed recently. Recording every single instruction in an application enables these debuggers to introduce a “step-back” functionality in addition to the common “step-over” and “step-into” semantics. As back in time debuggers have to record every single statement they introduce additional overhead in contrast to an approach based on atomic blocks.

## 6 Conclusion

In this paper a novel system mechanism ensuring consistency in the presence of exceptions is presented. A lean logging mechanism is proposed to enforce rollback semantics by introducing atomic blocks on the level of architectural components and try-blocks. A first proof of concept implementation has shown the general feasibility of the approach. Performance measurements are under way.

Of course, also the proposed approach has its limitations. If programs induce side effects (e.g. writing to an external file) the changes to external state can’t be rolled back. As explained, the isolation guarantees of single-threaded applications are exploited to minimize the logging overhead while still ensuring rollback semantics. Consequently, the approach is only applicable for the single-threaded case. Adapting the approach to concurrent systems is a topic of future research.

The proposed approach differs from other transactional approaches by exploiting isolation guarantees given by single-threaded applications. The resulting logging protocol is lightweight and imposes a minimal performance overhead on the application. Simplicity and transparency of the approach are enforced by relating atomic blocks to existing program and architectural structures freeing system developers of the burden to manually clean-up inconsistent state in the presence of exceptions.

## Acknowledgements

This work has been funded by the Klaus Tschira Foundation.

## References

- [1] A. Avizienis, J.-C. Laprie, B. Randell, and C. E. Landwehr, "Basic concepts and taxonomy of dependable and secure comput." *IEEE Trans. Dependable Sec. Computing*, vol. 1, no. 1, pp. 11–33, 2004.
- [2] C. Fetzer, P. Felber, and K. Hogstedt, "Automatic detection and masking of non-atomic exception handling," *IEEE Transactions on Software Engineering*, vol. 30, no. 8, pp. 547–560, August 2004.
- [3] C. T. D. Jr., "Data processing spheres of control." *IBM Systems Journal*, vol. 17, no. 2, pp. 179–198, 1978.
- [4] J. Gray and A. Reuter, *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann, 1993.
- [5] "Asm bytecode framework." [Online]. Available: <http://asm.ow2.org/>
- [6] "Java agent technologie." [Online]. Available: <http://java.sun.com/javase/6/docs/api/java/lang/instrument/package-summary.html>
- [7] "Eclipse platform." [Online]. Available: <http://www.eclipse.org/>
- [8] T. Härder and A. Reuter, "Principles of transaction-oriented database recovery," *ACM Comput. Surv.*, vol. 15, no. 4, pp. 287–317, 1983.
- [9] B. Stroustrup, "Exception safety: Concepts and techniques," in *Advances in Exception Handling Techniques*, 2000, pp. 60–76.
- [10] N. Shavit and D. Touitou, "Software transactional memory," *PODC '95: Proceedings of the fourteenth annual ACM symposium on Principles of distributed computing*, pp. 204–213, 1995.
- [11] T. Harris and K. Fraser, "Language support for lightweight transactions," *SIGPLAN Not.*, vol. 38, no. 11, pp. 388–402, 2003.
- [12] T. Harris, S. Marlow, S. Peyton-Jones, and M. Herlihy, "Composable memory transactions," *PPoPP '05: Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming*, pp. 48–60, 2005.
- [13] C. Fetzer and P. Felber, "Improving program correctness with atomic exception handling," *Journal of Universal Computer Science*, vol. 13, no. 8, pp. 1047–1072, 2007.
- [14] T. Harris, "Exceptions and side-effects in atomic blocks," *Sci. Comput. Program.*, vol. 58, no. 3, pp. 325–343, 2005.
- [15] B. Cabral and P. Marques, "Implementing retry featuring aop," in *LADC*, 2009.
- [16] B. Lewis, "Debugging backwards in time," *CoRR*, vol. cs.SE/0310016, 2003.
- [17] G. Pothier, E. Tanter, and J. Piquet, "Scalable omniscient debugging," vol. 42, no. 10. New York, NY, USA: ACM, 2007, pp. 535–552.
- [18] A. Lienhard, T. Gîrba, and O. Nierstrasz, "Practical object-oriented back-in-time debugging," in *ECOOP '08: Proceedings of the 22nd European conference on Object-Oriented Programming*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 592–615.

# An Automatic Failure Mode and Effect Analysis Technique for Processes Defined in the Little-JIL Process Definition Language

Danhua Wang, Jingui Pan  
State Key Laboratory for Novel Software Technology,  
Nanjing University  
Nanjing, China

George S. Avrunin, Lori A. Clarke, Bin Chen  
Department of Computer Science, University of  
Massachusetts, Amherst  
Amherst, USA

**Abstract**—Many processes are safety critical and therefore could benefit from proactive safety analysis techniques that attempt to identify weaknesses of such processes before they are put into use. In this paper, we propose an approach that automatically derives Failure Mode and Effect Analysis (FMEA) information from processes modeled in the Little-JIL process definition language. Typically FMEA information is created manually by skilled experts, an approach that is usually considered to be time-consuming, error-prone, and tedious when applied to complex processes. Although great care must be taken in creating an accurate process definition, with our approach this definition can then be used to create FMEA representations for a wide range of potential failures. In addition, our approach provides a complementary Fault Tree Analysis (FTA), thereby supporting two of the most widely used safety analysis techniques.

*Keywords*—FMEA; FTA; automatic; Little-JIL; safety analysis technique

## I. INTRODUCTION

Failure Mode and Effect Analysis (FMEA) [1, 2] is a bottom-up approach to analyzing and evaluating safety problems in a system or process in an attempt to reduce the occurrence of severe hazards or their consequences. A failure mode is “the way or manner in which a product or process could fail to meet design intent or process requirements” [3]. The potential impacts of a failure are defined as the effects of the failure mode [3]. In addition to identifying failure modes and effects, identifying the causes of an identified failure mode is another benefit of FMEA analysis. When consistently applied to a whole process, FMEA essentially consists of identifying and listing all potential failure modes, assessing the effects on the overall system for each failure mode, and then identifying all potential causes which could lead to each failure mode. Table 1 presents a small example of traditional FMEA worksheet.

FMEA can be automatically applied to a process if the process is modeled in sufficient detail and in a language with precise semantics. For this project, we used the Little-JIL process definition language, which has precise semantic definitions for all its language constructs.

Typically, FMEA is done manually by skilled experts. If the process being evaluated is at all complex, then this approach is known to be time-consuming, error-prone, and tedious. Our approach automatically applies FMEA to processes defined in the Little-JIL language. We assume that a Little-JIL process definition has been developed and validated with great care. Such a process definition can be used to study and evaluate the process [4], as well as to drive simulations [5] or executions. Thus, we assume that the process definition already exists and can be further leveraged by automatically identifying potential failure modes, and then automatically generating the effects and causes for any selected single failure mode.

Our approach overcomes the traditional shortcomings of FMEA when applied to complex processes. Also, our approach provides an integrated view of FMEA and Fault Tree Analysis (FTA), two of the most widely used safety analysis techniques. Experts can then focus their attention on the provided FMEA and FTA information for a process when trying to detect potential hazards and weaknesses.

To evaluate our approach, we have applied it to several detailed process definitions, defined in Little-JIL. We have selected processes from the healthcare domain, since hazards in health care processes can jeopardize the safety of the individuals served by them and even cause death and suffering. Moreover, FMEA has been used extensively by health care organizations to analyze safety problems in their processes to improve their safety [6, 7, 8]. Blood transfusion processes have often been the subject of FMEA [9]. In this paper, we use a simple blood transfusion process as an example.

The rest of this paper is organized as follows. Section II presents related work. Section III provides a brief introduction of the Little-JIL process definition language. Section IV gives a detail description of our automatic FMEA approach, followed by an concluding section.

## II. RELATED WORKS

In [10], an approach is presented to provide automated support for FMEA using model checking with the behavior tree system modeling notation. This approach enables safety analysts to work with high-level models in a notation that is

---

Sponsor: ①State Key Laboratory for Novel Software Technology at Nanjing University. Project Number: KFKT2009A13.

②Department of Computer Science at University of Massachusetts, Amherst.

TABLE I. TRADITIONAL FMEA WORKSHEET EXAMPLE

Failure Mode and Effect Analysis Worksheet								
Process: Simple Blood Transfusion Process								
SEV = How severe is the effect on the customer?								
OCC = How frequently is the cause likely to occur?								
DET = How probable is the detection of the cause?								
RPN = Risk priority number in order to rank concerns; calculated as SEV x OCC x DET								
Process Step	Failure Mode	Effects	SEV	Causes	OCC	DET	RPN	Actions
Obtain patient's blood type	"Blood Type" is wrong	"Blood Unit" is wrong	10	Wrong "Patient Name"	2	6	210	
				"Test patient's blood type" produces wrong "Blood Type"	5	4		
				"Contact lab for patient's blood type" produces wrong "Blood Type"	3	7		

close to natural language, while automating the tedious aspects of FMEA. Another model-based FMEA approach is proposed in [11, 12] by Papadopoulos et al., which realizes the semi-automatic synthesis of FMEAs which builds upon automatic fault tree analysis for system-level hazards. Many approaches are proposed to automate the creation of FMEA information from software [13, 14]. Another approach proposed by Robin Lutz is used to combine FMEA and FTA to analyze the requirements of safety critical software (e.g. spacecraft software) [15]. There have been several published studies demonstrating the benefits of employing FMEA in various domains. For example, Snooke N. et al. describes how model-based simulation can be employed to automatically generate the system-level effects of all possible failures on systems within the aircraft systems [16]. The application of functional modeling to the automatically produce FMEA information for mechanical systems is described in [17]. It is worth mentioning that the approach proposed in [11, 12] is not restricted to particular domains, i.e. applicable to a range of widely used engineering models.

### III. LITTLE-JIL PROCESS DEFINITION LANGUAGE

Little-JIL is “an executable, high-level process programming language with a formal (yet graphical) syntax and rigorously defined operational semantics” [18]. It provides a process modeling method basing on activities, which are defined as steps in Little-JIL processes. Little-JIL processes coordinate the activities of autonomous human or computer agents and their use of resources during the performance of their activities. A Little-JIL process model for blood transfusion is shown in Fig. 1. Here, we only give a brief introduction to the semantics of Little-JIL. Details of the language can be found in [19].

A Little-JIL process definition is a hierarchy of steps, each of which represents a single unit of work. Every step specifies all artifacts and resources it uses in its interface. A step can optionally be preceded by one or several pre-requisite step(s)

or/and be followed by one or several post-requisite step(s). A step without any sub-steps is called a leaf step. Each non-leaf step has a sequencing badge which indicates the execution order of its sub-steps. Artifacts, which are objects such as a medical chart or prescription, are passed between different steps via parameter bindings. There are four parameter types, IN, OUT, IN/OUT and Locals. In Little-JIL, resources are “special kinds of artifacts for which there is contention for access” [19]. Resources are managed by an external resource manager and their acquisitions need to be explicitly specified in step interfaces. Steps may throw exceptions, which can be handled by exception handler steps. Each step in Little-JIL is assigned to an execution agent (human or automated), which is responsible for performing the work associated with a step.

In the Little-JIL process definition, shown in Fig. 1, the root of the process, “Perform in-patient blood transfusion”, is a sequential step, which means its sub-steps “Obtain patient’s blood type”, “Pick up blood from blood bank”, and “Administer blood transfusion”, should be executed from left to right one by one. “Patient Name” is passed to “Obtain patient’s blood type” as an artifact via a parameter binding. Since “Obtain patient’s blood type” is a try step, its sub-step “Contact for patient’s blood type” and “Test patient’s blood type” should be tried from left to right until one of them is executed successfully and returns an artifact “Blood Type” to “Contact for patient’s blood type”. Then this “Blood Type” should be passed to “Pick up blood from blood bank”. After “Pick up blood from blood bank” is completed, the root step “Perform in-patient blood transfusion” should receive “Blood Unit” and then pass it to “Administer blood transfusion”, along with “Patient Name”, another artifact. “Patient Name” is passed to “Find patient location in computer” to get “Patient Bed Location”. Then “Blood Unit” and “Patient Bed Location” are finally passed to “Blood transfusion”.

Each of the leaf steps in Fig. 1 could be further decomposed into sub-steps, but that elaboration is not presented here.

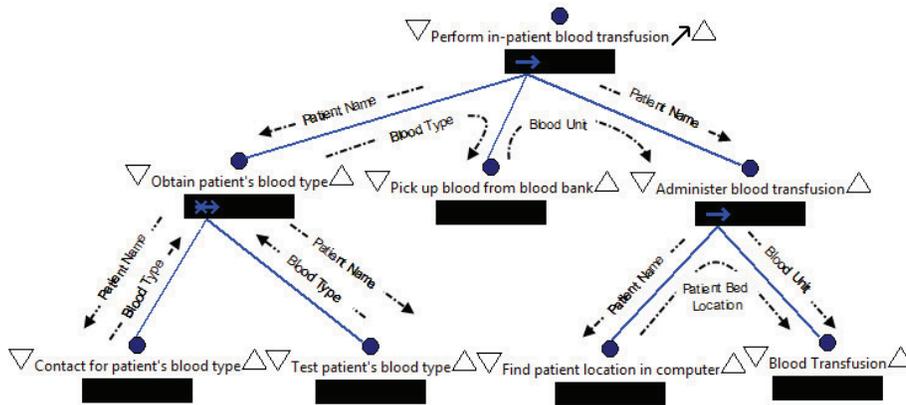


Figure 1. Simple Blood Transfusion Process

#### IV. AUTOMATIC FMEA FOR LITTLE-JIL PROCESSES

Our approach to automatically derive the FMEA information from Little-JIL involves three steps: identify potential failure modes, identify effects for each failure mode, and identify causes for a given failure mode. We describe each step in turn.

##### A. Identify Failure Modes

To automatically generate failure modes from Little-JIL process definitions, one needs to first identify appropriate failure modes for each language construct. In our work, we limit our attention to the failure modes related to artifacts and define two types of artifact-related failure modes:

Type 1: Artifact  $p$  to Step  $S$  is wrong.

Type 2: Artifact  $p$  from Step  $S$  is wrong.

Although not all failures can be associated with these failure modes, a large number of interesting failure modes are artifact-related failure modes or can easily be turned into artifact-related failure modes. For example, if a step is done incorrectly, we would expect that failure to be evident in one or more of the out artifacts associated with that step. When there are no such out artifacts, a hypothetical output could be created to represent the erroneous behavior of the step.

Each step, the basic elements of a Little-JIL process definition, has an interface that specifies the artifacts it uses as parameters along with each ones type. The failure modes related to artifacts for each step are created using the following rules:

- For an IN parameter  $p$  declared in the interface of Step  $S$ , the failure mode “Artifact  $p$  to Step  $S$  is wrong” (Type 1) is generated.
- For an OUT parameter  $p$  declared in the interface of Step  $S$ , the failure mode “Artifact  $p$  from Step  $S$  is wrong” (Type 2) is generated.
- For an IN/OUT parameter  $p$  declared in the interface of Step  $S$ , both the failure mode “Artifact  $p$  to Step  $S$  is wrong” (Type 1) and the failure mode “Artifact  $p$  from Step  $S$  is wrong” (Type 2) are generated.

- A local parameter is a special IN or OUT or IN/OUT parameter with a limited scope such that that associated artifact can only be passed between a step and its sub-steps. Thus, for this case, failure mode(s) can be generated the same as no-local IN, OUT, or IN/OUT parameters.

By generating potential failure mode(s) for each step in a process, all potential failure modes related to artifacts in the process definition are identified. For each of these failure modes, the potential effects need to be identified.

##### B. Identify Potential Effects for Each Failure Mode

The effects derivation algorithm consists of two phases:

1) *Phase1.* Construct the Artifact Flow Graph (AFG) from the unrolled Little-JIL process. The AFG can be easily constructed by traversing the process tree with an algorithm (not shown) that can be done in polynomial time.

The AFG is used to determine whether an artifact is data dependent on another artifact. An AFG is a directed graph  $G_a = \langle P_a, E_a \rangle$ , where  $P_a$  is the set of artifacts in the process and  $E_a$  is the set of edges, which represent the data dependent relationships between artifacts. Suppose both  $p_1$  and  $p_2$  are artifacts, there is an edge from  $p_1$  to  $p_2$  if and only if  $p_2$  is data dependent on  $p_1$ . If there is a parameter binding indicating that  $p_1$  is passed to  $p_2$  or if  $p_1$  is an input parameter and  $p_2$  is an output parameter of the same step, we create an edge from  $p_1$  to  $p_2$  indicating that  $p_2$  is data dependent on  $p_1$ .

Fig. 2 gives the corresponding AFG of the simple blood transfusion process modeled in Fig. 1. Every Node in Fig. 2 indicates an artifact in the process, which is represented as “artifact name (step name)” in the graph. The AFG can be generated directly from a Little-JIL process by traversing the process tree.

2) *Phase2.* Derive FMEA information using the AFG:

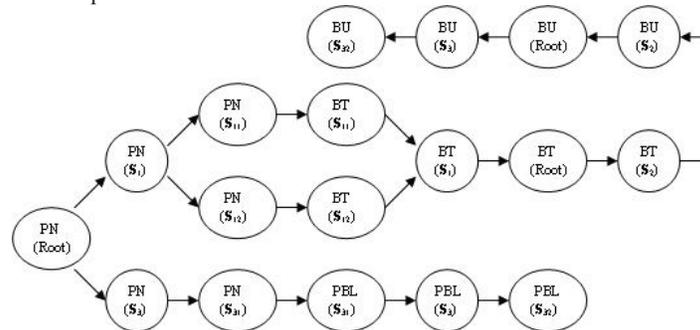
After generating the AFG for a Little-JIL process definition, the effects for each failure mode can be identified. If there is a path from artifact  $p_1$  to artifact  $p_2$  in the AFG, then a fault in  $p_1$  may be propagated to  $p_2$ . Thus, given a Type 1 failure mode (“Artifact  $p$  to Step  $S$  is wrong”) or a Type 2

failure mode (“Artifact p from Step S is wrong”), it is straightforward to determine incrementally the artifacts at which steps that could be contaminated by such a faulty p by traversing the AFG. In other words, a fault in p could be propagated to these artifacts causing them to be faulty. Therefore the fault of these artifacts is defined as the effects of the given failure mode.

A high-level description of the algorithm for deriving the FMEA information is given here:

```
//Initialization:
Initialize the visited AFG nodes vn set to empty;
Initialize AFG nodes nd set to all AFG nodes;
Initialize worklist wl set to empty;
//Main Cycle:
While nd is not empty do
    Remove an AFG node n from nd;
    Add n to wl;
    If vn contains n then continue;
    Maps n to the set of AFG nodes vnto set that directly
    flow to n;
    Replace n with set vnto;
    For each node n' in set vnto
        Add n' to wl;
    End for
    Add n to vnto;
End while
```

Formats of FMEA information may vary based on the needs of the organization and the requirements of the customer



Annotations: Artifacts: PN-Patient Name, BT-Blood Type, BU-Blood Unit, PBL-Patient Bed Location; Steps: Root-Perform in-patient blood transfusion, S<sub>1</sub>-Obtain patient’s blood type, S<sub>11</sub>- Contact for patient’s blood type, S<sub>12</sub>-Test patient’s blood type, S<sub>2</sub>-Pick up blood from blood bank, S<sub>3</sub>-Administer blood transfusion, S<sub>31</sub>-Find patient location in computer, S<sub>32</sub>-Blood Transfusion.

Figure 2. AFG of the Simple Blood Transfusion Process

▲ Obtain patient's blood type
▲ Artifact "Blood Type" from "Obtain patient's blood type" is wrong
▲ Artifact "Blood Type" to "Perform in-patient blood transfusion" is wrong
▲ Artifact "Blood Type" from "Perform in-patient blood transfusion" is wrong
▲ Artifact "Blood Type" to "Pick up blood from blood bank" is wrong
▲ Artifact "Blood Unit" from "Pick up blood from blood bank" is wrong
▲ Artifact "Blood Unit" to "Perform in-patient blood transfusion" is wrong
▲ Artifact "Blood Unit" from "Perform in-patient blood transfusion" is wrong
▲ Artifact "Blood Unit" to "Administer blood transfusion" is wrong
▲ Artifact "Blood Unit" from "Administer blood transfusion" is wrong
▲ Artifact "Blood Unit" to "Blood transfusion" is wrong

Figure 3. Part of FMEA Tree View Result for the step "Obtain patient's blood type"

[1]. Familiar formats are FMEA tables or FMEA worksheets. In our approach, we proposed a new representation, call an Effect Tree, which provides a way to organize the failure modes and the effects of a process into a tree view. The top level of an Effect Tree lists the step name of each process definition step. The second level lists the failure modes of each step. The third level lists the effects of each failure mode. The next and all subsequent levels list the effects resulting from the failure of their parent. This expansion continues until there is no subsequent effect that can be propagated, according to the AFG.

Fig. 3 shows one failure mode of step “Obtain patient’s blood type” and its effects. It indicates that wrong “Blood Type” produced by “Obtain patient’s blood type” could lead to the wrong “Blood Type” being sent to “Pick up blood from blood bank”, and subsequently results in the wrong “Blood Unit” being returned and then passed to “Perform in-patient blood transfusion”. Finally this could result in wrong “Blood unit” being provided to “Blood transfusion”. The artifact flow paths in the AFG that represent the fault propagation paths can be tracked by expanding the tree nodes in the Effect Tree.

Inspecting all effects of each failure mode should help identify effects that could result in significant damage. For the blood transfusion example, there are two effects that deserve more attention: “Blood Unit” to “Blood Transfusion” is wrong, “Patient Bed Location” to “Blood Transfusion” is wrong. The first one indicates that the wrong blood unit is transfused to the patient, and the second indicates that the blood unit is transfused to the wrong patient. Both of these could have

serious consequences.

### C. Identify Causes for a Given Failure Mode

In Fault Tree Analysis (FTA), severe consequences, such as the two listed above, are considered hazards. These identified hazards can then be treated as the TOP-events of a fault tree. Using the fault tree, FTA tries to determine which combinations of events must transpire for the hazard to actually occur. In our approach, we proactively identify possible hazards of a process using FMEA, and then use FTA to determine what events must occur for each hazard to arise.

The fault tree is a graphic model of the various parallel and sequential combinations of faults (events) that will result in the occurrence of the predefined undesired event (top event) [20]. Logical gates (e.g. AND, OR) are used to represent the interrelationships between events and the top event. The FTA approach involves two steps, deriving a fault tree and analyzing the fault tree. Deriving a fault tree starts with the top event, which is then further developed. All intermediate and necessary events that may lead to the top event are connected to the top event using appropriate gates. These new events will then be developed if they are not primary events. This procedure continues until all new events are primary events. Once a fault tree has been derived, both qualitative and quantitative analysis can be applied to analysis it. Through analyzing a fault tree, all minimal cut sets (MCSs), that is minimum combinations of events that would cause the top event to occur, can be found and their probability calculated. MCSs of a fault tree indicate whether the process is exposed to single points of failure or combinations of high-probability events. Subsequent changes may need to be made to the process to remove these weaknesses. Fig. 4 presents the generated fault tree for one of the hazards identified in the simple blood transfusion process using FMEA.

By generating the fault tree for the effects which may lead to a hazard, and doing FTA, process modifications may be recommended to try to prevent the hazard from occurring. Previous work has addressed how to automatically derive a Fault Tree from a Little-JIL process definition automatically

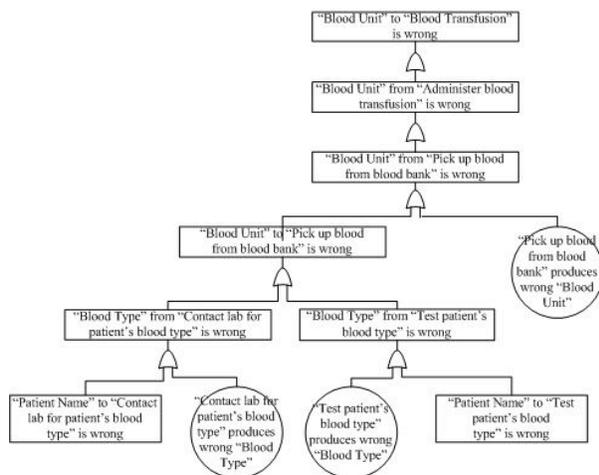


Figure 4. An example of generated fault tree

and thus, we only give a brief introduction of the approach here. For a detailed description of it, see [21].

The fault tree derivation starts with the given TOP-event, which is an intermediate event representing the given hazard (in our work, it is a given failure mode or effect). Then the intermediate event will be developed – all immediate and necessary events that could lead to this event are identified and connected to this event using appropriate gates. Those new events themselves may be intermediate events and need to be developed further. This process continues until all leaf nodes in the fault tree are primary events that do not need to be developed further as determined by the analyst. To automatically derive a fault tree from a Little-JIL process definition, there are two issues that have to be addressed: how to automatically extract fault tree events that could lead to a given event and how to connect them to this event using the appropriate gates. For the first issue, a few types of events are predefined which can be easily identified from the process definition. To address the second issue, a collection of templates are defined based on the Little-JIL process definition language. Different templates are used to develop different types of events.

## V. CONCLUSIONS AND FUTURE WORK

FMEA is an inductive technique for analyzing and evaluating potential reliability problems in a process or system. It is well accepted and applied in various kinds of industries especially safety critical processes, such as medical safety processes [22, 23, 24]. In this paper, we present an approach that can be used to automatically generate FMEA information from a Little-JIL process. Since Little-JIL steps have simple uniform interfaces, failure modes related to artifacts can be automatically generated. Effects of single failure mode can be identified by traversing the AFG of the process. The automatic FTA approach proposed in [21] is employed to generate fault trees for hazards identified by carefully checking FMEA information.

Performing fault tree analysis for all potential failure modes and effects in a process might be a huge undertaking and sometimes a waste of time and energy. The effects which may cause hazards should be identified and then further evaluated. We suggest that after generating all potential failure modes and their potential effects for a process, the effects of each failure mode should be examined carefully. If the effect is critical, a fault tree needs to be generated to find out the possible causes. By evaluating the possible causes of such effects, actions can be recommended to avoid the hazard. With our approach, such recommended changes would be made to the process definition which would then be reevaluated to assure that the problem had been effectively addressed.

Our proposed approach focuses on the problems that can arise though artifacts. Erroneous artifacts are probably the most common way in which problems will propagate though a process. There are other ways that this could happen however, such as though the use of erroneous resources or the faulty execution of a step that nonetheless does not contaminate the output parameters of that step.

After being acquired, Little-JIL resources can be passed as parameters like other artifacts [19]. Similarly, the artifacts produced by one step might be subsequently used as resources in another step. Therefore, faulty resources and resource passing between different steps might lead to fault propagation in the process. Also, as described in Section III, agents are treated as a special kind of resource in Little-JIL, and therefore errors introduced by agents should also be taken into account. Without considering resource faults leads to incomplete FMEA result. In other words, we need to determine the dependences between parameters and resources. Thus, one part of our future work should be finding out how resource fault affects the fault propagation in Little-JIL process definitions.

Another limitation is that subsequent steps could be dependent on a previous step being done correctly, where the failure to do is not reflected in an artifact. If we create a hypothetical output artifact to represent the erroneous step behavior, this artifact would not propagate erroneous information beyond this step. Therefore, we may want to find a way to add fault propagations introduced by erroneous step behavior to generate the FMEA information. This information would be overwhelming, however, unless it is guided by the analyst or some other external information.

In summary, automatically generating FMEA information from Little-JIL processes addresses the major weaknesses of traditional FMEA approaches. This information can be examined to determine which are the events of major concern, and these events can then be used to drive FTA. Thus, our approach aims to coordinate FTA and FMEA, two complimentary approaches, to help improve processes.

#### REFERENCES

- [1] Department of Defense, Procedures for Performing a Failure Mode, Effects and Criticality Analysis, MIL-STD-1629, Washington D.C., 1980.
- [2] N.G. Leveson, Safeware: System Safety and Computers, vol. 20, Published by Addison-Wesley, 1995.
- [3] Potential Failure Mode and Effects Analysis (FMEA), Automotive Industry Action Group (AIAG), 4th ed., 2008.
- [4] B. Chen, G.S. Avrunin, and E.A. Henneman, et al., "Analyzing medical processes," International Conference on Software Engineering(ICSE), Germany, 2008, pp. 623-632.
- [5] M.S. Raunak, L.J. Osterweil, A. Wise, L.A. Clarke, and P. Henneman, "Simulating patient flow through an emergency department using process-driven discrete event simulation," Software Engineering in Health Care(SEHC), Canada, 2009, pp. 73-83.
- [6] E. Stalhandske, J. DeRosier, R. Wilson, and J. Murphy, "Healthcare FMEA in the veterans health administration," <http://www.va.gov/ncps/SafetyTopics/HFMEA/PSQHarticle.pdf>, unpublished.
- [7] Failure Mode and Effects Analysis (FMEA): A framework for proactively identifying risk in healthcare, 1<sup>st</sup> ed., vol.6. Toronto ON: ISMP Canada, 2006.
- [8] Example of a Health Care Failure Mode and Effects Analysis for IV Patient Controlled Analgesia (PCA)," <http://www.ismp.org/Tools/FMEAofPCA.pdf>, unpublished.
- [9] J. Burgmeier, "Failure mode and effect analysis: an application in reducing risk in blood transfusion," Jt Comm J Qual Improv, 2002.
- [10] G. Lars, L. Peter, and Y. Nisansala, and M.H. Lee, "An automated failure mode and effect analysis based on high-level design specification with behavior trees," International Conference on Integrated Formal Methods (IFM), 2005.
- [11] Y. Papadopoulos, D. Parker, and C. Grante, "Automating the failure modes and effects analysis of safety critical systems," In: Int. Symposium on High-Assurance Systems Engineering (HASE), IEEE Computer Society, 2004, pp. 310-311.
- [12] Y. Papadopoulos, and M. Maruhn, "Model-based automated synthesis of fault trees from Matlab-Simulink models," Int'l Conf. on Dependable Systems and Networks(DSN'01), 2001, pp. 77-82.
- [13] C. Price, and N. Snooke, "An automated software FMEA," Proceedings of the International System Safety Regional Conference(ISSRC), Singapore, 2008.
- [14] H. Hecht, and R. Menes, "Software FMEA Automated and as a Design Tool," [http://www.sohar.com/proj\\_pub/download/Software\\_FMEA\\_Auto\\_As\\_Design\\_Tool.pdf](http://www.sohar.com/proj_pub/download/Software_FMEA_Auto_As_Design_Tool.pdf), unpublished.
- [15] R.R. Lutz, and R.M. Woodhouse, "Requirements analysis using forward and backward search," Annals of Software Engineering, Special Volume on Requirements Engineering, vol.3, 1999, pp. 459-475.
- [16] N. Snooke, C. Price, and C. Downes, and A. Carol, "Automated failure effect analysis for PHM of UAV," Proceedings of the International System Safety and Reliability Conference (ISSRC), April 2008.
- [17] N. Hughes, E.X Chou, C.J. Price, and L. Mark, "Automating mechanical FMEA using functional models," Proc 12th Int. Florida AI Research Soc. Conf(FLAIRS-99), 1999, pp. 394-398.
- [18] A. Wise, A.G. Cass, B.S. Lerner, E.K. McCall, L.J. Osterweil, and S.M. J Sutton, "Using Little-JIL to coordinate agents in software engineering," Automated Software Engineering Conference(ASE), Grenoble, France, 2000, pp. 155-163.
- [19] Little-JIL 1.5 Language Report, Department of Computer Science, University of Massachusetts, Amherst, 2006, unpublished.
- [20] W. E. Vesely, Fault Tree Handbook, U.S. Nuclear, Regulatory Commission, 1981, pp. 34-44.
- [21] B. Chen B, G.S. Avrunin, L.A. Clarke, and L.J. Osterweil, "Automatic fault tree derivation from Little-JIL process definitions," Proc. of the Int'l Software Process Workshop and Int'l Workshop on Software Process Simulation and Modeling (SPW/ProSim), 2006, pp. 150-158.
- [22] B. Duwe, B.D. Fuchs, and J.H. Flaschen, "Failure mode and effects analysis application to critical care medicine," Critical Care Clinics, vol(21), pp. 21-30.
- [23] M. Apkon, J. Leonard, L. Probst, L. Delizio, and R. Vitale, "Design of a safer approach to intravenous drug infusions: failure mode effects analysis," Qual Saf Health Care, vol(13), 2004, pp. 265-271.
- [24] W. Adachi, A.E. Lodolce, "Use of failure mode and effects analysis in improving the safety of i.v. drug administration, Am J Health-Syst Pharm, vol(62), 2005, pp917-920.

# Secure ad-hoc routing protocol

Thouraya Bouabana-Tebibel, Rym Nesrine Guibadj, Sara Mehar  
National School of Computer Science  
Algiers, ALGERIA

**Abstract**—Presently, the main concern of ad hoc routing protocols is no longer to find an optimal route to a given destination but to find the safe route free from malicious attackers. Several secure ad hoc routing protocols, proposed in the literature, are based on public key cryptography which drawback is to consume much more resources and decrease consequently network performances. In this paper, we propose a secure routing scheme that combines the hash chains and digital signatures to provide a high level of security while reducing the costs of hop-by-hop signature generation and verification. The proposed protocol is analyzed using the NS-2 simulator.

*Keywords*—component; routing; secure; MANET; hash chains

## I. INTRODUCTION

MANET or Mobile Ad hoc Network is a set of wireless mobile nodes, forming a temporary network without the use of any fixed infrastructure. Each node acts as a router (relay) and data packets are forwarded from node-to-node towards their destination in a multi-hop fashion. This architecture makes the network more vulnerable. Ad hoc routing protocols have been designed to be more and more efficient without keeping security in mind. This makes them vulnerable to a variety of attacks which affect the reliability of data transmission. The present question is no longer to find an optimal route to a given destination but the safe route free from malicious attackers.

In fact, most ad hoc routing schemes provide no security system. All entities can participate in routing and there are no barriers for a malicious node to cause traffic disruptions. The attacker wants essentially to affect the routing process, in order to control the network and destroy routing operations [18,16]. He achieves his objectives by: message alteration, message fabrication, message replay and impersonation.

In [19] a classification of insider attacks against mobile ad-hoc routing protocols is presented. IT includes route disruption, route invasion, node isolation, and resource consumption.

Many solutions are proposed to secure the most important routing protocols against those attacks [3,5,8,10,14,15,25]. But they remain incomplete, blocking only a subset of attacks among all those well-known for the damage they cause to the networks. On the other hand, each secure scheme defines an

appropriate environment of execution and presupposes a number of satisfied hypotheses to ensure its successful running. Indeed, protocols based on cryptography require a mechanism of key distribution and management [16]. Furthermore, when efficient, the used techniques are often too expensive in time calculation and memory space.

As a compromise, protocols based on reputation [7] integrate a new metric, the level of reliability of the route, to select the path towards destination. This reduces considerably the solution cost by diminishing calculation intensity.

As for intrusion detection systems, they can reduce the risks of intrusion but cannot completely eliminate them [20]. They also, sometimes fail with application of solutions as punishment of selfish nodes or location of malicious nodes which continuously change identity [27].

In terms of reliability, most of solutions rely on asymmetric cryptography and certificates delivered on line by authorities of certification. Message authentication and integrity are realized using digital signature [30]. When applied at each hop, they degrade the system performance.

The aim of our work is to protect the AODV protocol routing messages by using strong cryptographic functions and keeping in mind as main objective, minimization of complex calculation burden. This will be achieved by means of two essential mechanisms. The first one relies on hash chains which consume a little time for their generation and require a minimal storage space. The second one is digital signature that reinforces authentication and ensures integrity, and non-repudiation of messages. The latter is only applied on the source and destination nodes to reduce the latency.

The remainder of the paper starts with a brief description of the reactive routing protocol AODV. Section 3 deals with the core of our secure routing scheme SR\_AODV. We simulate in section 4 the performance of the proposed protocol using NS-2 simulator. In Section 5, we discuss works related to ours. We conclude by motivating our work and showing its novelty and relevance versus related works.

## II. AODV PROTOCOL

AODV (Ad hoc On-demand Distance Vector) is a routing protocol based on Distance-Vector routing algorithm [21]. It is a reactive routing protocol involving route construction only when data are available for transmission. It also uses destination sequence numbers, such as DSDV protocol, which

help to control freshness of the sent information in order to construct correct routes. AODV is based on two phases: route discovery and route maintenance.

When source node A needs to determine a route to a destination node B, it broadcasts a request message (RREQ : Route REQuest). As these requests spread through the network, intermediate nodes store reverse routes back to the origin node. Once the request reaches the destination, or an intermediate node with an active route, it generates a reply message (RREP: Route REPLY). The answer is sent unicast to the source following the reverse path, already built by the intermediate nodes.

RREQ packet format is shown in fig. 1. BroadcastId is a sequence number used as unique identifier for each RREQ. It sets distinction between a new RREQ and an already processed RREQ. The other fields are source and destination IP addresses (IDs and IDd) and source and destination sequence numbers (NSs and NSd). The Hop Count is the number of hops from origin to destination.

BroadcastID	IDs	NSs	IDd	NSd	HopCount
-------------	-----	-----	-----	-----	----------

Figure 1. RREQ packet format

RREP packet format is shown in fig. 2. It is composed of the same fields as RREQ excluding the BroadcastID and including the Lifetime which is the longest time authorized to be spent on the route.

IDs	NSs	IDd	NSd	HopCount	Lifetime
-----	-----	-----	-----	----------	----------

Figure 2. RREP packet format

AODV provides two methods to detect the presence of neighboring nodes and thus to detect the link break. In the first method, a node determines connectivity as long as it hears packets from its neighbors. Indeed, “HELLO messages” are periodically sent by each node to inform the neighbors that the link is still active. The second method is based on information directly received from the MAC sub-layer. If a link breaks within an active route, the node involved before the link break may choose to repair locally the link or deliver an error message (RERR: Route ERRor) listing the unreachable destinations. Thus, a new route discovery phase should be established by the source node [21].

### III. SR\_AODV SOLUTION

#### A. Basic assumptions

The protocol is based on the following assumptions:

- a packet sent from node A is received by the latter one hop neighbor B before a third node C replays the packet to B.
- a trusted Certification Authority performs the pre-distribution of both private key and X509v3 certificate

[6] to each member of the network through a physical contact. The conventional certificate X509v3 contains the public key, the identity of the certificate owner and other fields. All these fields are encrypted by means of a digital signature integrated to the certificate.

#### B. Proposed scheme

Secure protocols are based on complex cryptographic solutions; therefore they suffer from computation extra costs. This gives rise to the need for new schemes. In our contribution we propose security mechanisms that take into account the limited resources of nodes. These mechanisms are not based on unrealistic assumptions such as availability of an always-online security infrastructure (trusted third-party).

Our approach is inspired from Lamport authentication algorithm [12] used in remotely accessed computer systems. Authentication described by Lamport was designed for a client/server architecture where the management is centralized. In Lamport authentication, a server randomly chooses a password  $H_n$ . Afterwards, he applies to  $H_n$   $n$  times one-way function “h” to get  $n$  passwords ( $H_{n-1}, H_{n-2}, \dots, H_1, H_0$ ) called One Time Password sequence or OTP, for short.

$$H_n \rightarrow h(H_n) = H_{n-1} \rightarrow h(H_{n-1}) = H_{n-2} \rightarrow h(H_{n-2}) = H_{n-3} \dots \rightarrow h(H_1) = H_0$$

We were attracted by the effectiveness of hash functions because they reduce the high costs caused by traditional cryptographic mechanisms. Thus, we combined the use of hash chains authentication with digital signatures to achieve a satisfying security level. We adopt notations of table I.

TABLE I. NOTATIONS

Symbol	Signification
$SK_A, PK_A$	Private key. Public key of node A
$(d)SK_A$	A message encrypted with the private key of A
$[d]SK_A$	A message signed with the private key of A
$Cert_A$	A certificate belonging to node A
$ID_A$	Node A identifier
$H_j^A$	The $j^{th}$ element of the hash chain of node A

The security process we propose is divided into two phases: initialization phase and authentication phase.

##### 1) Initialisation phase

As discussed in basic assumptions, a trusted certification authority performs the pre-distribution of one certificate and one private key to each member of the network. Each entity, identified by  $ID_A$  constructs its own One Time Password sequence OTP. In order to ensure the provenance authenticity of the last value  $H_0^A$  while transmitting it to all one-hop neighbors of node A, we propose what follows. Node A first, encrypts  $H_0^A$  and then transmits the clear ( $H_0^A$ ) and encrypted ( $(H_0^A)SK_A$ ) values as well as node A identity ( $ID_A$ ) and certificate ( $Cert_A$ ) to all one-hop neighbors, refer to (1). Each

neighbor decrypts the encrypted  $H_0^A$  using the public key of node A transmitted within  $Cert_A$ . If the decryption succeeds,  $H_0^A$  integrity and origin are proved true. So,  $H_0^A$  value is saved in a new entry of the Neighbors table which keeps the  $H_0^A$  value of each neighbor. The comparison fails if either the sender authenticity or  $H_0^A$  integrity is compromised. In both cases, the message is ignored.

The one-hop neighbors also send the last values of their own hash chain and certificates to node A, see (2). At the end of this step, each node knows the  $H_0$  values of its one-hop neighbors.

$$A \rightarrow \text{Broadcast HELLO\_Auth: } \{ID_A, (H_0^A) SK_A, Cert_A\} \quad (1)$$

$$V \rightarrow A \text{ REPHELLO\_Auth: } \{ID_V, (H_0^V) SK_V, Cert_V\} \quad (2)$$

## 2) Authentication phase

When a node S needs to know a route to some destination D, and such a route is not available, it broadcasts a route request RREQ, see (3).

$$S \rightarrow \text{Broadcast RREQ: } \{\text{BroadcastID, IDs, IDd, NSs, NSd, } [\text{BroadcastID, IDs, IDd, NSs, NSd}]SK_S, Cert_S, H_i^S\} \quad (3)$$

This request contains the same basic AODV protocol fields excluding the hop count field, see figure 1. We add the source node certificate  $Cert_S$  in case of large-scale networks where nodes have not necessarily the public keys of all the network members.

The fields of the request are non-mutable. They are signed with the private key of S and accompanied by a hash value  $H_i$  ( $1 < i < n$ ). To not reuse a password already revealed, an index  $i$  is incremented within the node at every use.

The generated packet is then broadcasted on the channel. When a node receives it, it starts checking the  $H_i^S$  value to ensure that the packet comes from a legitimate node. To do so, it applies several times the hash function on  $H_i^S$  until it reaches  $H_0^S$ , the password initially transmitted by the neighbor and stored in the Neighbors table. If the receiver does not reach  $H_0^S$  after  $n$  iterations, it infers that the message is fabricated by a malicious node. So, it rejects it without any process. Otherwise the message is accepted. The intermediate node builds the reverse route to be crossed by the route reply packet, replaces the received  $H_i$  by a new password of his own chain, and finally broadcasts the RREP, see (4).

$$I \rightarrow \text{Broadcast RREP: } \{\text{BroadcastID, IDs, IDd, NSs, NSd, } [\text{BroadcastID, IDs, IDd, NSs, NSd}]SK_S, Cert_S, H_i^I\} \quad (4)$$

Eventually, the message is received by the destination D which verifies the source signature and then responds with a RREP. The digital signature authenticates the source and destination nodes and also ensures the non message replay. The first control is made on IDs and IDd fields when the second is performed on NSs and NSd fields.

In (5) J denotes the last intermediate node of the path, connecting S to D. The destination node sends the response to J according to the following formula (5) :

$$D \rightarrow J \text{ RREP: } \{\text{IDs, IDd, NSs, NSd, } [\text{IDs, IDd, NSs, NSd}] SK_D, Cert_D, H_i^D\}$$

RREP fields are signed by the destination node and value  $H_i^D$  is associated to the packet. Each node sending the response is authenticated along the path using  $H_i$ . The reverse route construction is exposed to the risk of a diversion launched by an attacker who responds instead of the destination node. This attack is detected thanks to the destination certificate which includes the destination identity. If the latter doesn't match with the destination identity invoked by the source node, one can deduce an intrusion attempt. As for the digital signature, it authenticates the source and destination nodes and controls the message replay. Finally, once the destination signature checked, the source node updates its routing table with the new path.

In [28] an approach comparable to ours, called a *Zero Common Knowledge authentication* was proposed. It differs from ours in the use of  $h(H_i)$ . In this approach, the receiver checks the value  $H_i$  by calculating only  $h(H_i)$  and testing the relationship  $h(H_i) = H_{i+1}$ . If checked then the node identity is proved true. This approach supposes the storage of the latest password which may put in check the control process in case of lost messages.

## IV. SIMULATION AND TESTS

In order to evaluate the routing protocol performance, one often uses simulation. In fact, it would be very costly, or even impossible, to establish a network for testing purposes. An ad hoc network simulation does not take much time, and it keeps us closer to the real use of the routing protocol. These two major advantages help us to better see the behavior of the protocol in different scenarios and evaluate its performance.

We carry out simulations using the NS-2 simulator [17]. We choose NS-2 due to its popularity among academic researchers [13,23]. In addition, it already supports a verified version of AODV. Simulations are hold considering a network of size 670 m x 670 m, composed of 20 nodes. We perform these simulations with the parameters defined in table II.

TABLE II. Simulation Parameters

Parameter	Value
Antenna	OmniAntenna
MAC layer type	IEEE 802.11
Radio propagation model	Two Ray Ground
Bandwidth	1Mb
CBR traffic	4 packets/s
Packet size	512 bit
Pause time	30 ms
Transmission range	250 m
Simulation time	200s

The nodes move according to the RWP mobility model (Random Waypoint Model). This model has become a standard in wireless networks research. It provides several scenarios where the mobile entities randomly move in the simulation area. For each experiment, we created several scenarios for traffic and mobility with the parameters set out above. Each time, we vary the speed between [0, 20m/s] and evaluate one of the following metrics:

1) *EED Average end to end delay*: this parameter gives average time required to transmit a data packet from the source node to the destination node.

2) *APL Average path length*: is calculated using the hop count field. It is often used as a metric for choosing the best path to route data.

3) *RL Routing load (the routing overhead)* : this parameter gives us information about the number of control packets, generated by the protocol for the path establishment and route maintenance.

To evaluate the SR\_AODV performances, we carry out our experimentations on three protocols: ARAN (Authenticated Routing for Ad-hoc Networks) [23] that has been chosen for its robustness and its high level of security, AODV and SR\_AODV.

ARAN is a secure protocol, implementing asymmetric cryptography. It uses a trusted Certification Authority called CA to generate certificates. Before entering the Ad-hoc network, each node requests a certificate from the CA. In ARAN, each node signs the discovery packets and route reply messages before retransmitting them. Each node verifies the previous node digital signature and then replaces it with its own. The cryptographic operations cause additional delays at each hop thus increasing the route acquisition latency. Only the destination can answer the Request packet. When the source receives the RREP, it verifies the destination signature. This allows an end-to-end authentication between the source and destination. However, the latency increases especially for long paths.

Fig. 3 shows that the increase in the movement speed leads to a rather large increase of the end-to-end delay. Indeed, the nodes movement involves frequent link failure in the established paths. Nodes are forced to rebuild invalid routes. Thus, delivery of data packets is delayed. We note that the required delay for ARAN is much higher than that of AODV and SR\_AODV.

Indeed, SR\_AODV established routes faster than ARAN: the time spent to check the hash values in SR\_AODV is insignificant, compared to the time needed in a certificate checking or a digital signature. We can therefore say that the processing of the digital signature only by the two path ends (source and destination) reduces the delay of packets transfer.

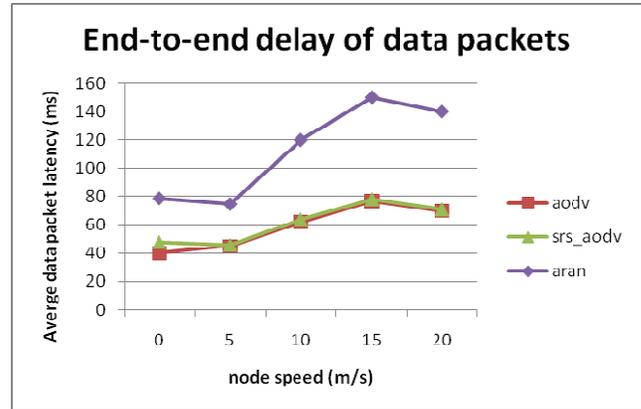


Figure 3. End to End Delay (EED)

The hop count field was omitted from the control packet. Paths are established using the fastest response (not necessarily the shortest path). Therefore, the routes set by SR\_AODV are expected to be longer than the AODV's. However, the curves of the three protocols are very similar (refer to Fig. 4). Indeed, authors of [23] found that the performance of data routing does not significantly decrease if we neglect the metric "hop count". This is confirmed through the results of our simulations. In fact, it may happen, in many cases, that the shorter routes are congested and unsafe.

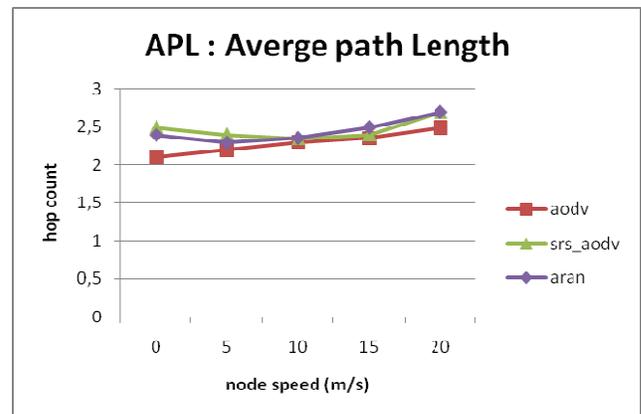


Figure 4. Average path length (APL)

Fig. 5 shows the size of control packets used to discover routes. We note that the three protocols apply for the rule: the packets need increase with a higher speed. The space overhead, caused by SR\_AODV and ARAN is higher than AODV, because of the size of certificates and digital signatures. SR\_AODV saves space compared to ARAN as only a single certificate and signature are used, unlike ARAN where two certificates and two signatures are required.

In a small scale network, we can assume that each member has the public keys of all other members. Thus, the space reserved for the certificate will be omitted from the control messages which improve the performance of our SR\_AODV solution.

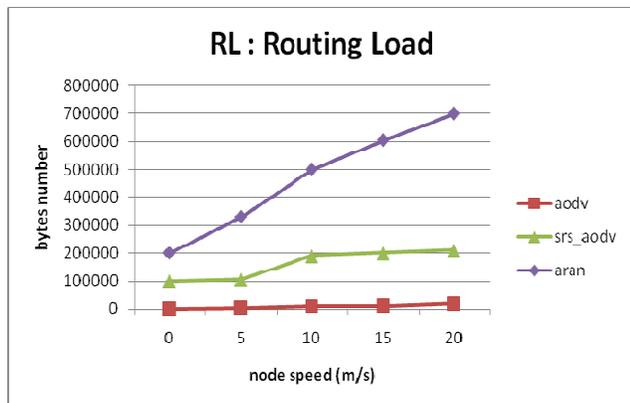


Figure 5. Routing Load (RL)

## V. RELATED WORK

Many studies were oriented towards the analysis of attacks against the AODV protocol. In [19] a systematic analysis of insider attacks against AODV is presented. The authors identify a number of attack goals, and then study how to achieve these goals through misuses of the routing messages. The analysis results reveal several classes of insider attacks, including route disruption, route invasion, node isolation, and resource consumption.

A large variety of solutions has been proposed to secure the AODV routing. As first defense line, we may find the classical mechanisms, using symmetric or asymmetric cryptography, digital signatures and hash chains. Intrusion detection systems and reputation mechanisms make a second defense line [22,26]. They succeed in taking suitable decisions on the network routing, as excluding malicious nodes, by monitoring and analyzing the nodes behavior.

Early works in this field are especially interested in authentication. Secure AODV or SAODV [29] was one of the first secure extensions of AODV protocol [13]. SAODV assumes that each ad hoc node has a pair of public/private keys. The node also holds the certified public keys of all network members. Two mechanisms are used at each hop to secure the AODV messages: digital signatures to ensure integrity of the message non-mutable fields, and hash chains to secure the hop count information, the only mutable information within the messages.

SAODV has some weaknesses. It is unable to detect fake RREP transmitted by compromised intermediate nodes. Moreover, the proposed solution entails an additional cost resulting from the accumulation of signature and hash operations at each hop. SAODV signatures require a

processing power that might be excessive for the ad hoc entities. Furthermore, the possession of certified public keys of all members is not obvious for large scale networks and due to the lack of memory resources.

To get better performance while maintaining the secure routing, Adaptive SAODV or A-SAODV was developed based on the SAODV implementation. Many analysis and comparison studies are made on the performance and impacts of AODV, SAODV and A-SAODV in a free-attack simulation environment [4]. The simulation results show the performance impact of security implementation into the original AODV after the implementations of SAODV and A-SAODV into the networks.

Other protocols have emerged as improvements to SAODV, FLSL (Adaptive Fuzzy Logic Based Security Level Routing Protocol) [9] and SAR (Security Aware Ad-hoc Routing) [18], for instance. In FLSL, a new attribute called *security level* is introduced in the format of the control messages to denote the reliability and dependability of certain mobile hosts or routes. The *security level* is used by source and destination nodes to determine the most secure and shortest route. As for SAR, it can discover a path with desired security attributes. The path found by the SAR protocol is not necessarily the shortest, but the safest of all the paths.

AODV and most of the on demand ad hoc routing protocols use single route reply along reverse path. Rapid change of topology causes that the route reply could not arrive to the source node. To avoid this, a new technique which tries multiple route replies is proposed in [24] and [11]. The proposed technique is called reverse AODV or R-AODV.

Latest researches in the area are conducted to secure ad hoc networks against grouped attacks. In [1] Awerbuch et al. propose ODSBR, the first on-demand routing protocol for ad hoc wireless networks that provides resilience to attacks caused by internal individual or colluding nodes. The protocol uses an adaptive probing technique that detects a malicious link after log n faults have occurred, where n is the length of the path. Problematic links are avoided by using a route discovery mechanism that relies on a new metric that captures adversarial behavior. Later, Awerbuch and Scheideler claim in [2] that the biggest threats appear to be join-leave attacks, used to isolate honest peers in the system, and against which no provably robust mechanisms are known so far. In this paper he showed that, on a high level, a scalable DHT can be designed that is provably robust against adaptive adversarial join-leave attacks.

## VI. CONCLUSION

The purpose of this paper is to present a new scheme to secure the AODV protocol. We showed that the proposed scheme made of hash chains and end-to-end digital signature provides a high security level at a very low cost.

The initialization phase always raises the problem of secure distribution of keys and passwords, particularly on large scale systems when the by hand distribution becomes unrealizable.

We proposed a remote pre-distribution carried out in an efficient and secure manner. We resorted afterwards to the use of one way hash chains to authenticate the control message senders during the route discovery. This authentication is principally useful while crossing the route from the source towards the destination. It guarantees the route drawing with legitimate nodes. Furthermore, the use of key-chain scheme is very well suited to pervasive computing devices since it requires nearly no computational power, very low bandwidth and memory storage.

At destination, the request message integrity is checked using a digital signature. This end-to-end checking ensures a fast source and destination authentication as well as a non message replay control.

Route diversion launched by an attacker who substitutes oneself for the destination node during the reverse route construction is detected by means of a digital signature. The latter protects against infiltration of internal and external malicious nodes.

The proposed solution can be extended to treat other attacks. It will be also, interesting to validate this work by formalizing the specification and verification of the SR\_AODV protocol.

#### REFERENCES

- [1] B. Awerbuch, R. Curtmola, D. Holmer, C. Nita-Rotaru, and H. Rubens, "ODSBR: An on-demand secure Byzantine resilient routing protocol for wireless ad hoc networks," *ACM Trans. Inf. Syst. Secur.*, vol. 10, no. 3, 2007.
- [2] B. Awerbuch and C. Scheideler, "Robust random number generation for peer-to-peer systems," *Theoretical Computer Science, Elsevier*, vol. 410, no. 6-7, 28 February 2009, pp. 453-466.
- [3] M. Burmester and B. de Medeiros, "On the Security of Route Discovery in MANETs," *IEEE Transactions on Mobile Computing*, vol. 8 no. 9, September 2009, pp. 1180-1188.
- [4] D. Cerri and A. Ghioni, "Securing AODV: The A-SAODV Secure Routing Prototype", *IEEE Communications Magazine*, February 2008.
- [5] R.Curtmola and C. Nita-Rotaru, "BSMR: Byzantine-Resilient Secure Multicast Routing in Multihop Wireless Networks," *IEEE Transactions on Mobile Computing*, vol. 8, no. 4, April 2009, pp. 445-459.
- [6] S. Eichler and C. Roman, "Challenges of Secure Routing in MANETs: A Simulative Approach using AODV-SEC," Technical Report: LKN-TR-2. Technische Universität München, Germany, 2006.
- [7] S. Galice, M. Minier, and S. Ubéda, "A trust protocol for community collaboration," In *IFIPTM, vol. 238 of IFIP*, 2007, pp. 169-184.
- [8] Y-C. Hu, A. Perrig, and D. B. Johnson, "Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks," *Wireless Networks*, vol. 11, n°. 1-2, Springer, 2005, pp. 21-38.
- [9] L. Jin, Z. Zhang, and H. Zhou, "Performance comparison of AODV,SAODV and FLSL routing protocols in mobile ad hoc network," *4th IEEE Consumer Communications and Networking Conference, CCNC 2007*, Jan. 2007, pp. 479-483.
- [10] S. Jung, B. Lee, E. Talipov, M.W. Ahn, and C. Kim, "Effects of Valid Source-Destination Edges for Node-Disjoint Multipaths on AD HOC Networks," MSV 2008, Las Vegas, USA, 2008, pp. 308-313.
- [11] C. Kim, E. Talipov, and B. Ahn, "A Reverse AODV Routing Protocol in Ad Hoc Mobile Networks", LNCS 4097, pp. 522 – 531, 2006.
- [12] L. Lamport, "Password Authentication with Insecure Communication," *Communication of the ACM*, vol. 24 , pp. 770-72, 1981.
- [13] Y. Lin, A. Rad, V.W.S Wrong, and J. Song, "experimental comparison between SAODV & AODV routing protocols," *1st ACM workshop on Wireless multimedia networking and performance modeling*. ACM Publisher, 2005, pp. 113-122.
- [14] H. Luo, J. Kong, P. Zerfos, S. Lu, and L. Zhang, "URSA: Ubiquitous and robust access control for mobile ad hoc networks," *IEEE/ACM Transactions on Networking*, vol. 12, n°. 6, Dec. 2004, pp. 1049-1063.
- [15] W. Mallouli, B. Wehbi, and A. R. Cavalli, "Distributed Monitoring in Ad Hoc Networks: Conformance and Security Checking," *The 7th International Conference on AD-HOC Networks & Wireless*, September 10-12, Sophia Antipolis, France, 2008.
- [16] A. Mishra, "Security and quality of service in ad hoc wireless network," Cambridge University Press, New York. 2008, pp. 3-106.
- [17] NS Manual. VINT Project. 2008. [http://www.isi.edu/nsnam/ns/doc/ns\\_doc.pdf](http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf)
- [18] S. Yi, P. Naldurg, and R. Kravets, "Security aware ad hoc routing for wireless networks," *2<sup>nd</sup> ACM int. symp. on Mobile ad hoc networking & computing*, Long Beach, USA. 2001. ACM Publisher. Pp. 299-302.
- [19] P. Ning and K. Sun, "How to Misuse AODV: A Case Study of Insider Attacks against Mobile Ad-Hoc Routing Protocols," *Ad Hoc Networks, Elsevier*, vol. 3, no. 6, November 2005, pp. 795-819.
- [20] J.-M. Orset, B. Alcalde, and A. R. Cavalli. "An EFSM-based intrusion detection system for ad hoc networks," *ATVA, LNCS*, 2005, pp.400-413.
- [21] C. Perkins, "Ad hoc On-Demand Distance Vector (AODV) Routing," INTERNET-DRAFT. IETF RFC 3561. 2003.
- [22] A. A. Pirzada, Chris McDonald, and Amitava Datta. "Performance Comparison of Trust-Based Reactive Routing Protocols," *IEEE Transactions on Mobile Computing*, vol. 5, no. 6, June 2006.
- [23] K. Sanzgiri, B. Dahill, B.N. Levine, C. Shields, and E. M. Belding-Royer, "Authenticated routing for ad hoc networks," *10th IEEE International Conference on Network Protocols*, Paris, France, 2002.
- [24] E. Talipov, D. Jin, J. Jung, I. Ha, Y. Choi, and C. Kim, "Path hopping based on Reverse AODV for Security," *APNOMS 2006, LNCS 4283*, 2006, pp. 574 – 577.
- [25] W-J. Tsauro and H-T. Pai, "A New Security Scheme for On-Demand Source Routing in Mobile Ad Hoc Networks", *IWCMC'07*, August 12-16, 2007, Honolulu, Hawaii, USA, pp. 577-582.
- [26] C-Y.H. Tseng, P. Balasubramanyam, C. Ko,R. L.J.Rowe, and K. Levitt, "A Specification-based Intrusion Detection System for AODV," *1st ACM workshop on Security of ad hoc and sensor networks*, Fairfax, Virginia. *SIGSAC*, ACM, 2003, pp. 125 – 134.
- [27] C-Y. H. Tseng, "Distributed Intrusion Detection Models For Mobile Ad Hoc Networks," PhD Thesis, University of California, 2006.
- [28] A. Weimerskirch and D. Westhoff, "Zero Common-Knowledge Authentication for Pervasive Networks," Tenth Annual International Workshop on Selected Areas in Cryptography SAC 2003, 2003.
- [29] M. G. Zapata, "Secure Ad hoc On Demand Distance Vector (SAODV) Routing", Mobile Ad Hoc Networking Working Group, Internet Draft, September 2005.
- [30] M. G. Zapata, "Key Management and Delayed Verification for Ad Hoc Networks", *Journal of High Speed Networks*, vol. 15, no. 1, Jan. 2006, pp. 93-109.

# A Pattern Methodology for Modeling Network Forensic Investigations in Converged Tactical Environments

Juan C. Pelaez

U.S. Army Research Laboratory  
APG, MD 21005, USA

[juan.c.pelaez@arl.army.mil](mailto:juan.c.pelaez@arl.army.mil)

*Abstract-Voice over IP over Wireless (VoIPoW) has had a strong effect on tactical networking by allowing human voice and video to travel over existing packet data networks along with traditional data packets. The U.S. Army is using IP telephony on high level units and it is expected that this technology will be fully implemented all the way from combat units to division level within the next lustrum. The implementation of wireless VoIP technology in the battlefield brings both network architecture and forensic challenges to the warfighter. This paper discusses the main challenges associated with security investigations in converged networks. Further, this research attempts to improve tactical Internetworking security by proposing a VoIP pattern system in order to create a UML Network Forensic Model for a simplified VoIP tactical environment. The pattern system will allow examiners to specify, analyze and implement network security investigations for different architectures.*

*Keywords-Voice over IP, tactical networks, semi-formal patterns, network forensics, network architecture.*

## I. INTRODUCTION

The US Army has adopted several cutting edge technologies in order to converge all tactical systems, including intelligence, surveillance and reconnaissance (ISR), unmanned aerial vehicle (UAV), sensor and intelligence systems, on a standard Internet Protocol backbone.

Voice over Internet Protocol (VoIP) has had a strong effect on tactical networks by allowing human voice and video to travel over existing packet data networks along with traditional data packets. Among the several issues that need to be addressed when deploying this technology, security is one of the most critical.

VoIP over wireless (VoIPoW) networks is becoming the most popular system for mobile communication in the world. However, studies of the security of wireless VoIP networks are still in their infancy. Wireless devices are commonly used by terrorists, and it is therefore necessary for network analysts to understand which evidence can be obtained from the VoIP system after an attack has occurred.

The increase in VoIP wireless devices provides a unique challenge for network investigators. While an attack on a wired network is investigated by tracing it back to a physical location, no physical access is required when a wireless medium is attacked, making it more difficult to extract evidence in this case.

Network forensics investigations in VoIP environments require real-time evaluation and analysis, in contrast to the

traditional method used in law enforcement, in which the victim's device is taken off-line after an attack has occurred.

This paper discusses the main challenges, technical considerations and tools associated with forensic investigations in converged networks. The goal is to help network investigators to identify actual intrusions, collect better evidence, reduce analysis time, and help to stop attacks against the VoIP network.

Additionally, this paper discusses network forensics in VoIPoW based on the technology that already exists, and identifies some of the issues that will have to be resolved in the future. To effectively analyze the network forensic challenges in VoIPoW, this paper starts by giving an overview of VoIP. Then an analysis of the network architecture challenges that tactical users face when implementing this wireless VoIP technology is presented. Further, the network forensic challenges on tactical networks are analyzed based on the Digital Forensics Research Workshop framework. Finally, the VoIP pattern system is introduced in order to create a semi-formal network forensic model for a simplified VoIP tactical environment. This paper ends with some conclusions and some ideas for future work.

## II. VOIP OVERVIEW

VoIP is defined as the transport of voice over Internet Protocol based networks, where any data network that uses IP, such as the Internet, intranets and Local Area Networks (LAN) can be used to establish this service.

VoIP has a strong effect on global communications by allowing human voice and video to travel over existing packet data networks along with traditional data packets. Therefore this technology will replace the traditional Public Switched Telephone Networks (PSTN) system, soon.

Also, VoIP over Wireless (VoIPoW) which is considered a typical application in IP telephony, is becoming the most popular system for mobile communication in the world.

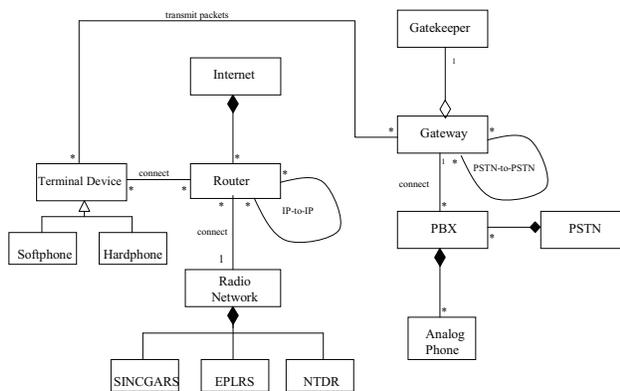
The convergence of voice, video and data in the same network brings both advantages and disadvantages to users, and security is one of the most critical issues that need to be addressed when deploying this technology.

### A. VoIP on Tactical Networks

In addition to combining voice, video and data onto the same network, VoIP has the benefit of being able to traverse different types of radio networks.

The U.S. Army is using VoIP on high level units (i.e. battalion, brigade and division). According to the Army's Communications-Electronic Research, Development and Engineering Center (CERDEC) in New Jersey, it is expected that this technology will be fully implemented all the way from combat units to division level by 2016.

Figure 1 shows a class diagram describing how an IP telephony system, the PSTN network and radio networks integrate together. This model shows how it becomes possible to place a call from a regular telephone number, or from a SINCGARS radio, to a PC running an H.323 client (and vice versa). The PBX which supports the standard phone (caller), formats caller and callee numbers and forwards them to the VoIP gateway via PSTN network. The gateway takes the voice call from circuit-switched PSTN and places it on the IP network. The gateway queries the gatekeeper via the IP network with caller/callee numbers (note that the voice packets do not go through the gatekeeper; only the call signaling goes through it) and the gatekeeper derives a routing number based upon service logic. Finally the gateway routes the call to called party (callee).



**Figure 1** - Class Diagram for a simplified Tactical Internet architecture

### III. NETWORK ARCHITECTURE CHALLENGES

The implementation of wireless VoIP technology in tactical networking brings both network architecture and forensic challenges to users.

This analysis starts with a comprehensive understanding of both the signaling and the standard protocols used today for providing wireless access in VoIPoW.

H.323 and the Session Initiation Protocol (SIP) are the two standard protocols used today for signaling and call control in VoIP, and they are essential for providing total access and for supporting IP-based services.

In a tactical environment, it is necessary to structure the components and procedures required for delivering multimedia communication services (i.e. voice, video and

data) across packet-based networks (e.g. Internet, intranets and Local Area Networks (LANs)). The H.323 protocol is rather complex and requires a combination of components to perform its functions. The solution to this problem is affected by the following forces:

- Converged environments have a large variety of users and require the use of multiple signaling protocols. Examiners usually conduct investigations in architectures that support both SIP and H.323 calls. Therefore this network forensic model must be able to operate in a converged environment using multiple existing and potential signaling protocols.
- Likewise in a tactical environment, Wi-Fi and WiMax (Worldwide Interoperability of Microwave Access), also known as the IEEE 802.11 and IEEE 802.16 respectively, are the two standard protocols used today for providing wireless access in VoIPoW. In large-scale tactical networks, interworking between these standards is common since the WiMax purpose is to expand the range of wireless systems access.
- It is essential to define a high-level specification of the VoIP architecture that can be used to conduct forensic investigations in a tactical environment. In this context, it is necessary to analyze the interoperability with other multimedia service networks and terminals. Terminal devices in disparate networks (e.g. softphone communicating with an analog phone) communicate frequently.

### IV. NETWORK FORENSIC CHALLENGES

Several models are used for investigation in forensic science. In order to analyze the network forensic challenges on tactical networks we chose the framework from the Digital Forensics Research Workshop (DFRWS) because it is a comprehensive approach and is more oriented to these research goals. The DFRWS model depicts the sequential process for digital forensic analysis [DFRWS01]. These steps are shown in Table 1.

The initial phase or the Identification of potential digital evidence (i.e., where might the evidence be found) is covered by the Intrusion Detection Systems (IDS) and in some sense by the attack patterns. The Preservation phase involves acquiring, seizing, and securing the digital evidence; making forensic images of the evidence; and establishing the chain of custody. This research focuses on the middle phases of the forensic process (i.e. the Collection Examination and Analysis of the evidence) where the presented patterns will provide network investigators a structured method to: understand VoIP forensic challenges; collect more and better evidence; and, reduce the analysis time in converged networks.

The Presentation phase involves the legal aspects of the forensic investigation – presenting the findings in court and corporate investigative units by applying laws and policies to the expert testimony and securing the admissibility of the evidence and analysis. This phase is outside of the scope of this research, but it must be considered in order to create a comprehensive model.

IDENTIFICATION	PRESERVATION	COLLECTION	EXAMINATION	ANALYSIS	PRESENTATION
Event/Crime Detection	Case Management	Preservation	Preservation	Preservation	Documentation
Resolve Signature	Imaging Technologies	Approved Methods	Traceability	Traceability	Expert Testimony
Profile Detection	Chain of Custody	Approved Software	Validation Techniques	Statistical	Clarification
Anomalous Detection	Time Synch.	Approved Hardware	Filtering Techniques	Protocols	Mission Impact Statement
Complaints		Legal Authority	Pattern Matching	Data Mining	Recommended Countermeasure
System Monitoring		Lossless Compression	Hidden Data Discovery	Timeline	Statistical Interpretation
Audit Analysis		Sampling	Hidden Data Extraction	Link	
		Data Reduction		Spatial	
		Recovery Techniques			

**Table 1** - DFRWS Digital Investigative Framework [DFRWS01]

### A. Collection Challenges

In this phase, the problem can be summarized in one question: How to efficiently collect digital attack evidence in real-time from a variety of VoIP components and networks? The solution to this problem is affected by the following forces:

- General security mechanisms, such as firewalls and Intrusion Detection Systems (IDS), cannot detect or prevent all attacks. They are unable to stop/detect unknown, internal attacks, and attacks that come in the body of the messages (e.g. steganophony attacks [Pei09b]). It is indispensable to analyze how an attack happened in order to counter it in the future.
- A real-time application, like VoIP, requires an automated collection of forensic data in order to provide data reduction and correlation. Current techniques dealing with evidence collection in converged networks are based on post-mortem (dead forensic) analysis. A potential source of valuable evidence (instant evidence) may be lost when using these types of forensics approaches.
- Even though there are a number of best practices in forensic science, there are no universal processes used to collect or analyze digital information. A systematic structure is needed.
- A forensic investigator needs forensic methods with shorter response times due to the large volume of irrelevant information and increasingly complex attack strategies [Wan05].

### B. Analysis Challenges

In the analysis phase the problem is: How to analyze evidence identified and extracted by the VoIP Evidence Collector in order to discover the attack source and other characteristics of the attack? The solution is affected by the following forces:

- Two of the most costly, time-consuming, and human-intensive tasks are the analysis and reconstruction of attacks in a compromised system.

- Because the amount of data generated by VoIP networks is significant, storing network data for forensic analysis may be complicated.
- In a converged environment an automated technique is required to locate the attackers and reconstruct their criminal actions.
- Criminals may also attempt to cover the traces of their malicious work by using anti-forensic methods to manipulate and tamper with the evidence or interfere directly with the network forensic investigation [Jah08].
- The main disadvantage for the forensic examiner is that encrypted packets are difficult to analyze.
- The forensic analysis process must guarantee data preservation and integrity.
- Attacks in converged networks are becoming more frequent and more complex to counter.
- A method is required for reusing network forensic knowledge and documenting forensic investigations.
- Forensic incidents in VoIP are often faced by examiners who do not have experience executing investigations or using appropriate forensic tools.

## V. NETWORK MODELS USING PATTERNS

The purpose of this paper is to generate a comprehensive pattern system based on a collection of architectural, attack, security and forensic patterns, providing best practices for all IP telephony systems. The goal is to analyze and to understand network forensic investigations in a tactical VoIP converged environment using existing methods for this basis.

The pattern system will allow examiners to specify, analyze and implement network security investigations for different architectures. The pattern system will make use of UML (Unified Modeling Language) [Boo98] to generate these patterns.

This research will present effective ways for network investigators to implement the use of network forensics as a secure and convenient method of collecting digital evidence in a VoIP environment.

### A. VoIP pattern system

This research approach proposes the use of four different types of patterns:

- Architectural patterns analyze existing VoIP architectures in IP telephony. These patterns are used for modeling tactical architectures using the UML language. This research presents a pattern based methodology where patterns are used for high-level specification of the VoIP system.
- Attack patterns which are defined as a systematic description of the steps and goals of an attack, and ways to defend this system. An attack pattern template is used in order to describe how to document and organize

generic attack patterns. An attack pattern catalog is also provided.

- Security patterns focus on the security mechanisms and standards to stop attacks against the VoIP system. From the list of attacks one can figure out what security patterns are necessary to prevent these attacks.
- Forensic patterns focus on capturing, recording, and analyzing information collected on VoIP networks from several intrusion detection, auditing, and checking points. These patterns will help network investigators to understand which evidence can be obtained from the VoIP system after a specific attack.

Figure 2 shows a pattern system diagram which focuses on integrating these four types of patterns (i.e. Architectural, Attack, Security and Forensic) in order to create a semi-formal network forensic model for a simplified VoIP environment. The combination of these patterns [Fer07a], presents effective ways in which network investigators can more effectively implement the use of network forensics as a secure and convenient method of searching, collecting and analyzing digital evidence in a tactical wireless VoIP environment. The central part of this pattern system is the VoIP network forensic model. The usefulness of this model will directly depend on a comprehensive VoIP pattern system.

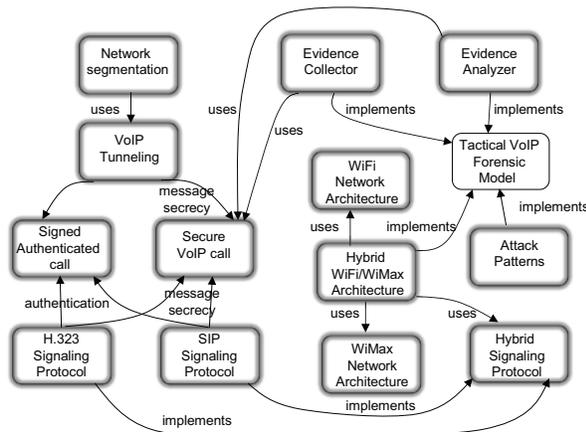


Figure 2 - VoIP Pattern System

The objective is to develop an architectural pattern-based methodology that can be used to guide the design of wireless VoIP systems and products as well as to simulate these systems. In other words, a system of IP telephony patterns is intended to describe how to secure and analyze attacks in VoIP.

### B. Network Forensic Model

In our technical approach we use a distributed attack sensing and warning system realized as a series of network-based collection sensors to acquire relevant forensic information so that intrusion analysts can perform effective analysis.

In [Pel09a] we proposed a new type of pattern, the *forensic pattern*. This pattern represents a systematic approach to network forensic collection and analysis of data. We use forensic patterns to provide a comprehensive solution to previously mentioned network forensic challenges in converged tactical environments.

#### 1) VoIP Evidence Collector

This pattern defines a structure and demonstrates a process for collecting VoIP attack packets on the basis of adaptively setting filtering rules for real-time collection. The collected data is sent to a network forensics analyzer for further analysis. This data is used to discover and reconstruct attacking behaviors.

In VoIP forensic investigations, collection sensors will be deployed in the converged environment, thus reducing the manual collection of data. In order to capture all voice packets entering or leaving the system, these hardware devices will be attached in front of the target servers (e.g., call server) or sensitive VoIP components. These sensors will also be used by the IDS to monitor the VoIP network.

In order to maintain efficiency when capturing network traffic, we select the data to save, such as source and destination addresses and ports, and protocol type. The evidence collector can then extract all or selected voice packets (i.e., request or response) over the VoIP network by applying a filter. The data collected by the evidence collectors is stored in a storage area network (SAN) cluster and will be used to perform the corresponding forensics analysis. We can also use network segmentation techniques [Fer07b] to monitor the voice virtual local area network (VLAN) traffic independently from data VLAN traffic, although the two share the same converged network.

*a) Structure:* Figure 3 shows the UML class diagram of the evidence collector (modified from [Ren05]). The **evidence collector** is attached to hosts or network components (e.g. **call server**) at the node where we need to collect evidence in a **VoIP network**. Forensic data is collected using **embedded sensors** attached to key VoIP components or Network Forensic Analysis Tools (**NFAT**). VoIP components that are monitored can provide forensics information once an attack occurs. The evidence collector should be designed to extract forensic data and securely transport it (i.e., hash and encrypt) to the **forensic server** using a VoIP secure channel [Fer07b]. The forensic server combines the logs collected from the target servers and the VoIP network and stores them in its database to allow queries via command user interfaces. When attached to a terminal device, the evidence collector captures the **network traffic** to record the whole procedure of the intrusion and can be used to reconstruct the intrusion behavior [Ren05]. The evidence collector is also able to filter out certain types of evidence to reduce redundancy.

*b) Implementation:* After collecting the desired forensic data, the evidence collectors will send two types of data to the network forensics server, depending on the function performed. If the sensor is attached to a key VoIP compo-

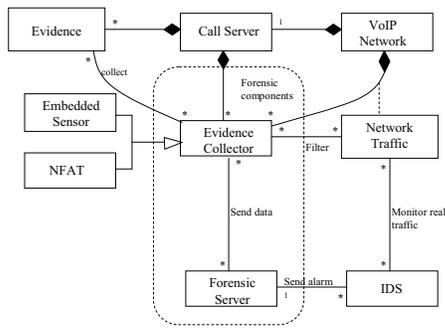


Figure 3 Evidence Collector Class Diagram

ment, it will collect logging system and audit data; otherwise (i.e., attached to a terminal device) it will act as packet sniffers do (with the Network Interface Card (NIC) set to promiscuous mode) or NFAT tools extracting raw network traffic data (e.g., entire frames, including the payloads, are captured with tcpdump). These data are used to discover and reconstruct attacking behaviors. After each attack against the VoIP network, the forensic data collected from key components and attacking sources may include logging data. The following data may also be useful to discriminate calls and call types:

- Terminal device information
  - Numbers called
  - Source and destination IP addresses
  - IP geographical localization
  - Incoming calls
  - Start/end times and duration
  - Voice mail access numbers
  - Call forwarding numbers
  - Incoming/outgoing messages
  - Access codes for voice mail systems
  - Contact lists
- VoIP data
  - Protocol type
  - Configuration data
  - Raw packets
  - Inter-arrival times
  - Variance of inter-arrival times
  - Payload size
  - Port numbers
  - Codecs

## 2) VoIP Evidence Analyzer

This pattern defines a structure and process to analyze the collected forensic data packets. It also presents a method of investigating an alleged IP attack scene and tracing back attackers.

In our approach we combine (i.e., pre-process and store) all forensic logs and network traffic captured by the evidence collector into a forensic data repository (database and files) and analyze them using techniques such as log correlation and normalization [For04]. Logs are processed and converted into a simple format and then compared with the set of predefined misuse and attack patterns to identify possible security violations [Ren05]. The raw traffic data must also be converted into a readable format and stored in a separate database.

The evidence analyzer then performs automated inference based on the evidence database and presents results to the forensic investigator. The analysis process involves using automated methods to sift through large amounts of acquired data and extract and identify data of particular interest [Gra05].

a) *Structure:* Figure 4 shows a class diagram describing how an IP telephony (as in figure 1) and a forensic system integrate together. This model shows the three primary forensic components: the **evidence collector**, the **forensic server** and the **network investigator**. The evidence collector is attached to a host that may be attacked in a VoIP network (e.g., **gatekeeper**).

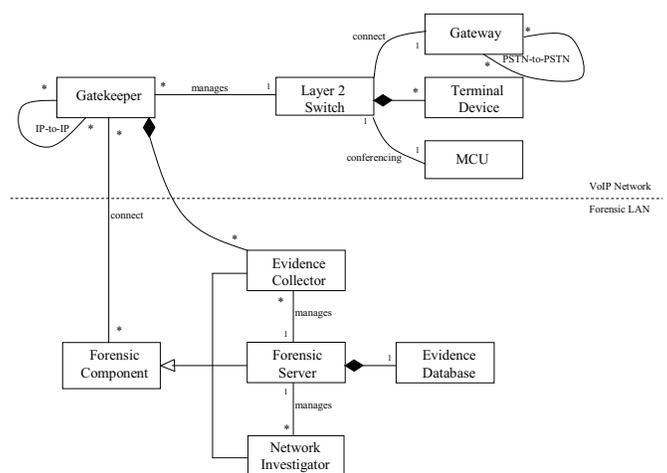


Figure 4 Class diagram for a VoIP network forensics system

The main function of the forensic server is combining the logging information collected from the VoIP network and its key components, and storing it in the evidence database. On the other hand, the network investigator acquires information about attackers and their sources by using techniques such as IP traceback and packet marking and by mapping topology to geographic locations so as to conduct further investigations.

b) *Implementation:* After the IDS gives the alert, the network forensics server sends a command to the network investigator (the response is in real-time). The network investigator receives information from the forensic server about sensitive spots on the VoIP network. Then the network investigator surveys the network in order to obtain

useful information, such as the attacker location and phone numbers. The network investigator will also scan the network for mapping topology to find, for example, a false proxy server, or traceback the location of the attacker [Ren05]. Finally, the network investigator sends the scan and survey result to the forensic server using a VoIP secure channel [Fer07b]. This result will include such information as the topology of the network, the IP address, the MAC address, the possible geographic location of the IP.

The network forensics server can also analyze the attack behavior by replaying the attacking procedures. Network forensics tools can reorganize the packets into individual transport-layer connections between machines [Ren05]. With the appropriate tools, investigators can capture the packets and decode their voice packet payloads in order to analyze VoIP calls.

The forensics server provides correlations of forensics data in order to discover the attack behavior. This process will provide network investigators a better way to monitor voice traffic data and correlate events from VoIP security mechanisms (e.g., IDS).

To construct the given same events, it is necessary to correlate the different format logs to a single-layer data format by time, IP, and User ID. This task is known as normalization [For04]. Correlation in forensics is based on the knowledge of previous attacks gained by historical methods, geographical location, strength of signal, and the behavior of the attacker. Likewise, attack patterns [Fer07a] will provide prior knowledge of known exploits. VoIP correlation rules correlate events taken from multiple VoIP source devices, including call managers, IP PBXs, and voice gateways [Hic07]. These correlation rules will detect, for example, theft of service attempts as well as DoS attacks against VoIP servers.

## VI. CONCLUSIONS AND FUTURE WORK

In conclusion, VoIP will become more ubiquitous in the near future, with the probability of becoming the most popular system for mobile communication, therefore it is important to study the mechanisms and tools for forensic analysis on converged networks.

The forensic information found in VoIP systems has a great potential to be used as evidence. Forensic patterns value provide the most when UML models are reused on similar investigations.

This research presents effective ways in which network investigators can more effectively implement the use of network forensics as a secure and convenient method of collecting digital evidence in a VoIP environment.

This research focused on the functionality offered by these semi-formal UML patterns. The main contribution is the creation of a comprehensive pattern system to be used in forensic investigation processes. The usefulness of a VoIP network forensic model will directly depend on this VoIP pattern system. These are the first steps toward a methodology for modeling network forensics. Future work

will include the generation of additional forensic patterns including IDS versions of this approach.

## ACKNOWLEDGEMENTS

Special Thanks to Robert Reschly and Duane Wilson for their many helpful comments and discussions regarding this research.

## REFERENCES

- [Boo98] G. Booch, and J. Rumbaugh. "The Unified Modeling Language User Guide", Addison-Wesley Pub Co; 1st edition, Boston (September 30, 1998).
- [DFRWS01] Digital Forensics Research Workshop. "A Road Map for Digital Forensics Research 2001." *Digital Forensics Research Workshop 6 November* (2001): <http://www.dfrws.org>
- [Fer07a] E. B. Fernandez, J. C. Pelaez, and M. M. Larrondo-Petrie. "Attack patterns: A new forensic and design tool." *Procs. of the Third Annual IFIP WG 11.9 International Conference on Digital Forensics*, Orlando, FL, Jan. 29-31, 2007.
- [Fer07b] E.B.Fernandez, J.C. Pelaez, and M.M. Larrondo-Petrie, "Security patterns for voice over IP networks", *Procs. of the 2nd IEEE Int. Multiconference on Computing in the Global Information Technology (ICCGI 2007)*, March 4-9, Guadeloupe, French Caribbean.
- [For04] D. Valentino Forte, The Art of Log Correlation - Tools and Techniques for Correlating Events and Log Files, IR Italy Project, 2004.
- [Gra05] T. Grance, S. Chevalier. "Guide to Computer and Network Data Analysis: Applying Forensic Techniques to Incident Response (Draft)." *Recommendations of the National Institute of Standards and Technology*. August, 2005.
- [Hic07] A. Hickey. "VoIP security monitoring gets proactive." *SearchVoIP.com*, 25 Jan 2007.
- [Jah08] H. Jahankhani, E. Beqiri. "Memory-Based antiforensic tools and Techniques." *International Journal of Information Security and Privacy*, Volume 2, Issue 2.
- [Pel09a] J.C. Pelaez and E.B. Fernandez. "VoIP Network Forensic Patterns." *Fourth International Multi-Conference on Computing in the Global Information Technology (ICCGI 2009)*. Cannes, France, August 23-29, 2009.
- [Pel09b] J.C. Pelaez. "Using Misuse Patterns for VoIP Steganalysis." *Third International Conference on Secure Systems Methodologies Using Patterns (SPattern'09)*. Linz, Austria, August 31- September 04, 2009.
- [Ren05] W. Ren, H. Jin. "Distributed Agent-based Real Time Network Intrusion Forensics System Architecture Design." *Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA'05)*. March, 2005.
- [Sch06] M. Schumacher, E.B.Fernandez, D. Hybertson, F. Buschmann, and P. Sommerlad, Security Patterns: Integrating Security and Systems Engineering, Wiley publishing, New York, 2006.
- [Wan05] W. Wang, T. Daniels. "Building Evidence Graphs for Network Forensics Analysis." *Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC 2005)*. September 2005.

# P2PsecT: Peer-to-peer Security Testbed

A research platform for peer-to-peer security

Eduardo Segura, Xiao Su  
Computer Engineering Department  
San Jose State University  
San Jose, USA  
esegura@computer.org, xiao.su@sjsu.edu

**Abstract**— As peer-to-peer (P2P) systems grow in popularity, their benefits begin to appeal to more traditional users, like the military and the enterprise. However, Information Assurance on P2P is currently performed on a case-by-case basis. The solutions proposed, although necessary, have been isolated, focusing on the specific problems at hand for a particular P2P architecture. This paper discusses the benefits that a security-centric simulation platform would provide. While other P2P simulators have long been available, none of them has a focus on security. To this end, we propose a components-based framework to enable experimentation with different security parameters. By using components with well-defined interfaces, we aim to minimize the effort required to use a different strategy for a function, be it routing a message, or finding a peer. The ultimate goal is to be able to compare the systems side-by-side. Our objective is for this framework to provide useful comparison metrics, thus establishing a common test bed for the different security models that exist today for P2P.

*Keywords*-Security, System Design, Testbed, Peer-to-Peer

## I. INTRODUCTION

There is considerable interest in the security aspects of peer-to-peer (P2P), and in security in general. The DETER testbed [10] is a prominent example of this interest.

To tackle P2P security research, scientists have created taxonomies [1][2][3], based on different criteria. These taxonomies are then used to classify related security problems into specific categories, e.g. “trust abuse” [2][3].

Wallach [1] explores and classifies some of the common functions in P2P systems, enumerating possible weaknesses in each category.

A different classification is proposed in [2], where the authors use the concept of “rational attacks”. These attacks are used to identify different sets of related “attack vectors”.

Another framework is proposed in [3], where the authors analyze different security aspects of existing P2P systems. They perform their study by defining an “adversary model”, which is then used to classify the different vectors available.

We believe that these classifications are invaluable to understand what tools are available to an attacker. But they need to be fleshed out as software. We thus propose our platform: P2PsecT. The objective is to provide the building blocks for any P2P system, focusing on defining an architecture

of components that deal with the different classes of threats, proposed in the work above.

## II. PROPOSED PLATFORM

Our platform provides a playground where developers can implement the specific security-sensitive functions they need, and then study how those affect their P2P systems. In this section, we will review the high-level functionality offered by each of the layers.

### A. Basic P2P primitives

This subsystem encapsulates the bulk of the usual peer-to-peer transport functionality. We use it to decouple our components from the particular P2P transport used.

As we will revisit later on, this approach has proved to be quite challenging. To begin with, different P2P networks use different calling conventions. It is quite difficult to develop an abstraction layer that would work over several networks.

Another issue arises with the need to exchange data between different components. The solution we considered is to have several data mapping algorithms.

### B. Security Providers

Any module that encapsulates a piece of functionality directly related to security, we call it a “Security Provider”.

Take, for example, RoutingProvider. Depending on her needs, the implementer will have to code the behavior for her specific routing algorithm. Otherwise, one of the options provided by the framework could be selected.

An issue arising from this architectural design is the cross-component incumbency of concepts. For instance, a routing algorithm could make routing decisions based on the particular representation of a node’s Id. One such routing algorithm is Uinta[4]. The consequence of this design decision is that it is now hard to decouple the routing algorithm from the identification mechanism.

### C. Application-facing services

This interface has to be flexible enough as not to rely on the specific mechanism enforced by the underlying platform, while enabling access to as many of its functions as possible.

These are, however, conflicting factors. On one hand, we have the functionality provided by one particular platform. On the other hand, we have the portability that is sacrificed by the use of those ready-to-use functions.

We have thus settled on reusing as much functionality provided by the underlying platforms as possible. Although this might not be ideal, it is the most practical for this work.

### III. FRAMEWORK DESIGN

As we saw in the previous section, our framework is organized around components. These components are in charge of enforcing different security functions of the P2P architecture. In this section, we will briefly introduce four of those security components. For each one of them we will provide a high level overview and a description of the main responsibility.

#### A. Identification

The purpose of this component is to provide secure identifiers. Several strategies can be used to create an object's id. Each one of those has different characteristics, presenting different sets of benefits and challenges. It is up to the user of the framework to decide, depending on specific environment conditions, which strategy should be used to generate the ids.

#### B. Routing

A major issue that needs to be addressed, if we are to create a reliable P2P network, is to provide secure routing primitives. In order to achieve this goal, several routing algorithms have been proposed. We intend to abstract the rest of the framework from the particular implementation of the routing component. Hence the need to have a specific provider for it.

#### C. Replication

Replication is the mechanism by which a given resource, e.g.: a file, is redistributed to other peers in the network. There might be several reasons why a P2P application would want to offer this functionality. We are not interested on those reasons, but on the fact that this vector can be used to launch an attack on the network. It is thus necessary to provide a component that provides security for this function.

#### D. Confidentiality

A basic requirement of any secure network is to be able to transmit confidential data. It is also a requirement to ensure that the data has not been modified during transit. It is thus imperative that we provide a layer of indirection over specific cryptographic functions. In order to study the impact of different algorithms we define a cryptographic API, which will be used to encrypt, decrypt, sign, and verify the information that is sent over the P2P network.

### IV. SAMPLE INSTANTIATION

In this section we will show how we built a very simple, insecure P2P application and how we improved on it using our framework.

#### A. First, insecure version

We found one simple application as part of the JXTA samples that matched what most P2P applications do at a basic level: message exchange.

After a small modification, we ended up with a simple application that had all the moving parts needed to represent a more complex P2P application. The client is fairly simple: It tries to connect to the server and transmit a short message. The

peer's server thread should then acknowledge reception by printing a short message to the standard output.

#### B. Why is this version vulnerable

To keep our analysis short, we will not try to discover all possible attack vectors. Instead, we will focus on those vulnerabilities that will be addressed later by our improved version.

First, and perhaps most important for this scenario, all communication between peers in this sample application is done in clear text.

Another concern is that there is no way to guarantee the anonymity of communications. Imagine the situation where a 'whistle blower' wants to criticize the activities of her boss. In order to provide a guarantee that she won't suffer retaliation, anonymity must be enforced.

Finally, note that in this example, we are relying on the identification mechanism provided by default in JXTA. This component generates IDs in a pseudo-random fashion. This particular approach of identification tokens leaves the P2P application vulnerable to Sybil attacks.

In Fig. 1, we see output from this simple application. As we will show in the following section, this output varies quite a bit between versions, thanks to the packet being encrypted during transmission. We will also show how we achieve anonymity.

#### C. New version: impact on security

At this point we have a simple, naïve implementation of a P2P application, vulnerable to several types of attacks. Here we will study what has changed in the new version, specifically regarding the functionality that provides security measures and how they change the scenario depicted above.

The first issue that we reviewed for the previous version was encryption. On the new one, we have provided a simple form of encryption.

Our next task is to provide anonymity during transmission by using a specific implementation of the routing component.

This takes us to our next step: How do we provide end-to-end anonymity between any two pair of nodes? One possible solution is to use Onion Routing [5], which uses successive layers of encryption. Now the nodes do not know where a packet is going, but only the next hop along its route.

A good way to visualize what has changed between the insecure version and this new one is to review the actual output of an execution. In Fig. 2 we show the output log from a typical test run.

What we see there is the packet as it transits through the network. Whenever a peer receives a packet, it will print its contents, and will then decrypt the payload, thus removing a layer of encryption from the 'onion'. Once the payload has been decoded it is sent to a different host.

```
Waiting for msgs on input pipe
-----Begin Message-----
Message Size : 87
Element PipeTutorial : 28
[Mon Aug 04 06:17:03 PDT 2008]
-----End Message-----
Message received at : Mon Aug 04 06:17:03 PDT 2008
Message created at : Mon Aug 04 06:17:03 PDT 2008
```

Figure 1. Text output from the insecure application

```

-----Begin Message-----
Message Size : 230
Element securep 2p : 174
[<ori><src>1</src><dst>1</dst><payload>=psj?=tsd?2=0tsd?=etu?2=0etu?=
qbzmpbe?>qtke@>ute@3>1ute@>fuv@2>1fuv@>rc{nqcf@KHooc#Zruog$>
Irc{nqcf@>1qtke@=0qbzmpbe?>=0psj?</payload></ori>]
-----End Message-----
Message received at : Mon Aug 04 20:13:26 PDT 2008
Message created at
<ori><src>1</src><dst>1</dst><payload>=psj?=tsd?2=0tsd?=etu?2=0etu?=
qbzmpbe?>qtke@>ute@3>1ute@>fuv@2>1fuv@>rc{nqcf@KHooc#Zruog$>
Irc{nqcf@>1qtke@=0qbzmpbe?>=0psj?</payload></ori>
message sent
-----Begin Message-----
Message Size : 176
Element securep 2p : 120
[<ori><src>1</src><dst>1</dst><payload>=psj?=tsd?2=0tsd?=etu?1=0etu?=
qbzmpbe?>Jgnnq?Yqtnf#=#0qbzmpbe?>=0psj?</payload></ori>]
-----End Message-----
Message received at : Mon Aug 04 20:13:26 PDT 2008
Message created at
<ori><src>1</src><dst>1</dst><payload>=psj?=tsd?2=0tsd?=etu?1=0etu?=
qbzmpbe?>Jgnnq?Yqtnf#=#0qbzmpbe?>=0psj?</payload></ori>
message sent
-----Begin Message-----
Message Size : 122
Element securep 2p : 66
[<ori><src>1</src><dst>0</dst><payload>Iifmmp!Xpsme" </payload></ori>]
-----End Message-----
Message received at : Mon Aug 04 20:13:26 PDT 2008
Message created at
<ori><src>1</src><dst>0</dst><payload>Iifmmp!Xpsme" </payload></ori>
Application received this data: Hello World!

```

Figure 2. Debug output from the new version

## V. ANALYSIS OF RESULTS

In this section, we analyze attack vectors, what framework components have been defined to address those vectors, and enumerate what threats are mitigated with the current implementation.

### A. Possible attack vectors

The framework, being a security-specific one, has to provide countermeasures against most threats. In addition to that, it should provide extension points to allow for future enhancements. Table I reflects what we believe are some of the

most important issues affecting P2P applications today. We will later provide a tentative solution for the items in each category.

### B. Managing the threats: How to manage those attack vectors

After reviewing what threats should we focus on, we have to decide how we deal with them. To reduce the complexity of the problem, we first focus on finding what threats can be grouped together. To be able to do this, we define a criterion for classification, explicitly stating what characteristics we want to prioritize.

Table I shows that classification. For each class, we propose a component that could take responsibility for implementing a solution for it.

### C. Current Status: What are we managing today?

As we expected, it turns out there are many subtle trade-offs across components. Consider the choice of a particular identification strategy for assets stored in the network: It may deeply limit what options are then available for replication of that asset.

On the flip side, that was one of the main goals of our work: To identify what alternatives, and their trade-offs, are available to a developer when creating a P2P application, and how those options affect the resulting software. The developer should be able to replace any component with an alternative, and be able to study, compare and draw conclusions about how that change impacts the system.

Table II offers a summary of components currently in use. We do not include the components that make up the interface to the application layer, or to the P2P platform below, because those are considered as infrastructure for the purpose of this work.

TABLE I. SURVEY OF ATTACK VECTORS

Attack	Component	Example of attack	Benefit obtained by attacker
Excuses	Policy enforcer	Attacker claims that he is unable to provide resources.	Attacker doesn't need to share resources
Picking on newbie	Policy enforcer	Attacker may abuse new peers who have to pay dues first	Attacker doesn't need to share resources
Fudged books	Policy enforcer	Accounting records are modified	Attacker appears to share resources
Forged evidence	Policy enforcer	Attacker produces proof of collaboration in the system	Attacker appears to have shared resources
Accounting	Policy enforcer	Attacker manages to stop the auditing process	Cannot determine if attacker collaborated
Group deception, local honesty	Policy enforcer	A group of colluding nodes modify their accounting records as to show collaboration	The colluding nodes look like they have collaborated, but have not.
Elusion	Maturation	Attacker gets a file or any resource then immediately leaves.	Attacker benefits, without giving back
Sybil	Identity manager	The attacker abuses the system, but it is able to disassociate himself with the identity that has the bad reputation	The attacker looks, after getting a new id, like an innocent peer.
DoS	Several	Attacker disrupts service by joining and leaving repeatedly	Attacker is able to disrupt the network
Eavesdropping	Confidentiality	Attacker is able to look into the data transmitted by other nodes in the network	Attacker can gain vital information to gain privileges
Censorship	Replication	Attacker is selected to hold the replicas of the objects that the system is supposed to distribute	Attacker can deny retrieval of an object, or infect it with malware.
Censorship	Routing	An attacker can locate himself in a way where he controls access to a given resource, by controlling all routes to a node	The attacker can deny access to objects or nodes

TABLE II. COMPONENTS AVAILABLE IN OUR FRAMEWORK TODAY

Component	Implementation Status	Lessons Learned	Challenges for future undertakings
Confidentiality	Simple algorithm has been provided. Reliable encryption uses Java API	Implementing an abstract interface that can accommodate many crypto algorithms is hard	Provide more cryptographic algorithms, keep those algorithms updated over time, extend the interface to accept new kinds.
Routing	Infrastructure is in place, with two sample routing strategies	Flexibility is paramount when dealing with this component. Routing can depend on many things, like IDs	Some routing algorithms are heavily dependant on the structure of the IDs. This can make it hard to keep the coupling of those components low.
Identity Manager	One type and one generator are in place. Framework provides interface for adding more	IDs are tightly coupled to the underlying P2P platform. It is hard to not used a similar schema used by the hosting platform	Some other components, like replication, might be heavily dependent on the internal structure of an ID. It can prove difficult to separate them.
Replication	This component has been modeled, implementation is partial	Replication is applicable only when there are discrete items to be distributed. Not as useful for CPU time	Some replication strategies offer good features, but at the cost of a compromise in some other component

D. Conclusion

By no means is the development of a security framework an easy task. The challenges are many and across the board. Let us review some of them:

If there is an exploitable bug in the framework, the whole purpose of its very existence can then be questioned. Special care must be taken to design and develop this platform.

If the framework lacks features, its usefulness will be limited. The endeavor needs to be ongoing, not a one-time effort. By making this a community-wide, open-source project, the resulting platform can be made comprehensive, including algorithms and mechanism that might not be available otherwise.

If the framework is not maintained, it can become obsolete very fast. Even modern cryptography will be easier to crack with the advent of more powerful computers.

In spite of all these challenges, there are many things to be gained: easier implementation and testing of P2P systems, allowing students to try different parameters and interactions between different components, and to study the weaknesses of a system in a controlled environment.

VI. RELATED WORK

Other projects, like PEPERS [11], have already identified the potential benefits of creating a simulation testbed for P2P security.

Another related project is the work of [6], which studies the security features of large P2P grid systems. They also propose a framework, only that they focus on one particular implementation, grid architectures.

There is also a project called Phyllo [7], but only seems to address routing strategies.

A different approach is found in JXTA [8]. This package is not a framework, but a specification of what the communication protocols between the peers should be. This work does not focus on security, instead providing a common platform for building P2P applications.

Finally, we have GUNet [9]. GUNet is a secure P2P framework. It does not, however, focus on security. It has many interesting features, like anonymous transmission, but it does not provide as much flexibility to modify key P2P components, like Id generation.

VII. ROOM FOR IMPROVEMENT AND RESEARCH OPPORTUNITIES

To begin with, the framework needs more components. Developing more algorithms and strategies will improve the framework, as weaknesses will surely be identified.

Another possible improvement is to provide an interface to a different P2P platform. Currently the framework is built using JXTA. To compare different P2P infrastructures, we need to build more components for the lower layers, in charge of translating data types, adapting function calls, and enabling communication in general.

As a final point, we need to validate the framework itself. For example, our framework is mostly event-driven. Such a bias should be addressed, if it is found to limit instantiations.

REFERENCES

- Wallach, D. (2002). Survey of P2P Security Issues. Proc. of ACM ISSS.
- Nielson, S. J., Crosby, S., & Wallach, D. S. (2005). A Taxonomy of Rational Attacks. Proceedings 4th Int. Workshop on Peer-to-Peer Systems (IPTPS), (pp. 36–46). Springer Berlin: Heidelberg.
- Sit, E. & Morris, R. (2002). Security Considerations for Peer-to-Peer Distributed Hash Tables. Proc. of Fourth IPTPS: Cambridge, MA.
- Jin, H., Xu, J., Zou, B. & Zhang, H. (2005). Uinta: A P2P Routing Algorithm Based on the User's Interest and the Network Topology. Lecture notes in CS. (pp. 238-249). Germany: Springer-Verlag.
- Dingledine, R., Mathewson, N., & Syverson, P. (2004). Tor: The second-generation onion router. Proceedings of the 13th USENIX Security Symposium, (pp. 303–320). Berkeley, CA: USENIX Assoc.
- Ellahi, T. N., Hudzia, B., Mcdermott, L., & Kechadi, T. (2006). Security Framework for P2P Based Grid Systems. Proceedings of The Fifth International Symposium on Parallel and Distributed Computing, (pp. 230-237). Washington, DC: IEEE Computer Society Press.
- Heinbockel, W. & Kwon, M. (2005). Phyllo: A Peer-to-Peer Overlay Security Framework. In First Workshop on Secure Network Protocols (NPSec), (pp. 43-48).
- Li Gong. (2001). Project JXTA: A technology overview. Technical report, SUN Microsystems. Retrieved February 28, 2008 from <http://www.jxta.org/project/www/docs/TechOverview.pdf>.
- Bennett, K., Grothoff, C., Horozov, T., Patrascu, I. & Stef, T. (2002). GUNet-A truly anonymous networking infrastructure. Retrieved February 28, 2008 from <http://gnunet.org/download/main.ps>.
- T. Benzel, R. Braden, D. Kim, C. Neuman, A. Joseph, K. Sklower, R. Ostrenga, and S. Schwab. Experience with DETER: A Testbed for Security Research. In Proceedings of Tridentcom, March 2006.
- Mobile peer-to-peer security infrastructure, <http://www.pepers.org>. Last accessed on Mar. 7, 2010

# Model-Driven Development of Java Enterprise Applications

Andre Pflueger, Wolfgang Golubski, Tobias Haubold

Zwickau University of Applied Sciences, Informatics, Zwickau, Germany

E-mail: andre.pflueger@fh-zwickau.de

## Abstract

*MDSO is highly regarded and already used in industry. This paper presents an approach for the model driven development of enterprise applications with minimum modeling effort and a maximum of configuration possibilities for the software developer. The approach uses the GeneSEZ framework with the concept of model annotations. It uses UML profiles with stereotypes and tagged values which can almost be used like the known Java annotations.*

## 1. Introduction

Nowadays Model Driven (MD) approaches are highly regarded, visible by the increasing number of acronyms: Model Driven Architecture (MDA), Model Driven Software Development (MDSO), Model Driven Engineering (MDE), Model Driven Test Development (MDTD), etc. MD approaches are successfully applied in software projects [1, 2] and are an appropriate instrument for developing large business applications. A formal model, e.g. the Unified Modelling Language (UML), is evaluated to generate source code (model-to-text transformation). In order to focus on the domain a Platform Independent Model (PIM) is preferred. Platform and technology specific information can be added e.g. by characterizing UML elements using the profile mechanism [3]. The result is a Platform Specific Model (PSM), e.g. a persistence model.

The Java Enterprise Edition (Java EE) [4] provides technologies to build enterprise applications. These technologies are described as specifications defining functionality and characteristics which a software product has to provide in order to implement this specification. There are three basic specifications for enterprise applications using the well-known three layer architecture:

- Java Persistence API (JPA) for persistence,
- Enterprise JavaBean (EJB) for business logic and
- JavaServer Faces (JSF) for presentation.

This paper introduces an approach to build Java EE applications with the help of MDSO and model annotations.

These annotations allow the software developer to enhance the domain model by as few as possible technology specific information and get a technology compliant model. Missing information are completed without losing the ability for a detailed configuration. This process requires an easily accessible meta model provided by the Generative Software Engineering Zwickau (GeneSEZ) framework [5]. It provides an own meta model and uses openArchitectureWare [6] for model-to-model and model-to-text transformations.

Section 2 describes an example illustrating the concept of the model annotations explained in section 4. Section 3 gives a brief overview of the Java Persistence API (JPA) and Enterprise JavaBean (EJB) profiles. Section 5 summarizes benefits and discusses some practical experiences. The related work is mentioned in section 6 before section 7 concludes the paper and gives future prospects.

## 2 Example

The example used in this paper is a simplified bank scenario shown as a UML class diagram in figure 1. Banks can have customers and possess address information. The customers can have three types of bank accounts mainly differing in their interest rates. The checking account is used for daily cash flow with a low interest rate whereas the other two accounts offer higher interest rate for saving money. The `CustomerBean`, `BankBean` and `CommunicationBean` contain the business logic like CRUD operations or email communication.

This example contains uni- and bidirectional associations with different multiplicities and generalization relationships plus operations distributed on different classes.

## 3. The JPA and EJB profiles

The JPA and EJB specifications define annotations to include metadata into the source code (*source code annotations*). The introduced stereotypes are designed in dependence of these annotations. They enable the software developer to annotate model elements and are therefore called *model annotations*. This allows the software developer to

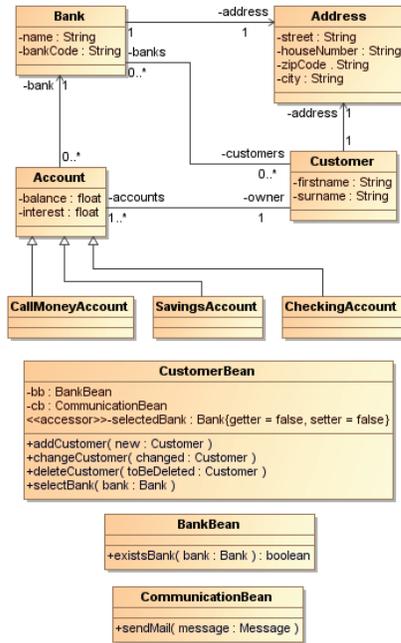


Figure 1. UML class diagram of the bank scenario

use the stereotypes on the model level in the same way he uses the annotations on source code level, and to create a complete specification compliant model. This model is the foundation for the GeneSEZ framework to generate the appropriate source code. The profiles are realized by the profile mechanism of the UML. Every stereotype name starts with the prefix *jpa* or *ejb3* to avoid naming conflicts. The profiles use enumerations as data types where possible in order to maximize type safety and provide default values.

## 4. Model annotations

This section describes the approach of model annotations with their goals and the way of creation. First, a general description is provided, followed by examples according to the specifications JPA and EJB.

### 4.1. General approach

The main goals of model annotations are:

- They can almost be used like source code annotations.
- They create a complete specification compliant model with minimum effort but still in-depth.
- Technology specific information are minimized to focus on the enterprise application domain.

Two different ways have been unfolded to create the stereotypes with their tagged values (the model anno-

tations). On the one hand there are source code annotations mapped one-to-one to the model annotations, e.g. *ejb3Local*. On the other hand there are reasonable summarizations of source code annotations mapped onto one model annotation, e.g. *jpaPersistentEntity* and *ejb3SessionBean*. Reasons for such summarizations are (1) they often appear together or in combination or (2) they have the same context and very similar or equal attributes.

The properties of the source code annotations are mapped onto the tagged values of the corresponding stereotype, ensuring that every configuration possibility provided by the specification is available for the software developer on model level. The concept of configuration by exception is adopted for the model annotations and allows the use of tagged values like source code annotation properties.

The UML model contains much information which can be used to reduce modeling effort for the software developer. According to a specific semantic context the software developer can provide either (1) no specific information at all, (2) partial information or (3) all information.

### 4.2. JPA model annotations

In order to exemplify the introduced approach some model annotations of the JPA profile are presented in context of the bank example (see section 2). Figure 2 shows a JPA compliant class diagram of this example. The missing classes are handled in section 4.3. The stereotype *entity*<sup>1</sup> characterizes an element to be persistent and is an equivalent for *jpaPersistentEntity*. This single stereotype is sufficient to map the classes in figure 2 to a relational database (first variation of providing information). This stereotype can be used in an early phase of the software development process, using MDA with JPA model annotations and the results can be reused in implementation phase to create e.g. a prototype which can be configured in more detail without creating a new model or changing the used stereotypes.

In the following three examples of additions and modifications are listed to show how much information are already available in the UML model or provided by default values:

- An attribute with name *id* is added and annotated with *jpaPrimaryKey*.
- Association attributes are provided with *jpaAssociation* and the corresponding enumeration literal of *jpaAssociationType*.
- A no argument constructor as well as getter and setter are added to every entity.

Listing 1 shows the generated source code for the class `Account`. Due to our experiences (see section 5) manual

<sup>1</sup>UML standard obtained Jacobson classification ([Jac92])

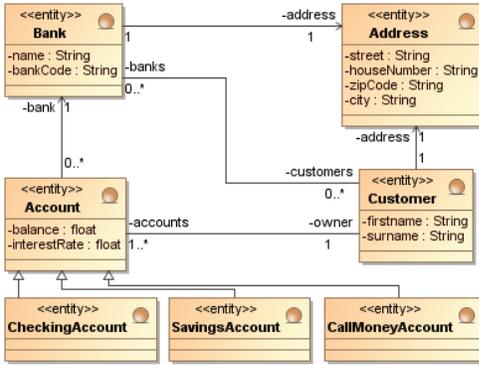


Figure 2. persistence model of the bank example

changes to the source code of entity beans are rare. Therefore programming effort to create entities using model annotations is very low.

```

@Entity
@Table(name = "tbl_account")
@Inheritance(strategy = SINGLE_TABLE)
public class Account implements Serializable {
    @ManyToOne(cascade = {})
    private Customer owner;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    public Account() {
    }
    //getter and setter [...]
}
  
```

Listing 1. source code of class Account

An example for the earlier mentioned summarization reason<sup>2</sup> is the model annotation *jpaPersistentEntity* causing the generation of the source code annotations *@Entity* and *@Table*. We know no JPA specific case in which the information about the database table is necessary without characterizing the class as an entity.

The association Bank-Customer exemplifies how omitted information are completed. First, multiplicities on both sides of the association are evaluated, 0..\* and 1..\*, leading to the *jpaAssociationType* ManyToMany for both association attributes. Afterwards the direction is determined: bidirectional. JPA needs to know the owner of this association which is not provided by the model. Therefore the class which is evaluated first becomes the owner. The software developer can change this behavior by assigning *true* to the tagged value *mappedBy* of the stereotype *jpaAssociation*. The field name for the corresponding JPA information is detected from the model data.

Despite this completion the software developer can add specific information to the association attributes, e.g. the cascade type. This will not affect completion and allows a more detailed configuration.

<sup>2</sup>source code annotations appear often together or in combination

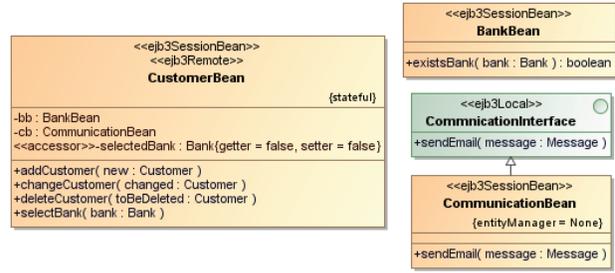


Figure 3. session beans for the bank example

### 4.3. EJB model annotations

CustomerBean, BankBean and CommunicationBean contain the business logic for the introduced bank example. Therefore they are characterized as EJB session beans in figure 3.

Session beans need business interfaces. Using the EJB profile it is unnecessary to model them. By default a local business interface is added for every session bean, e.g. the BankBean. This can also be done by using the model annotation *ejb3Local*. A remote business interface is added if the stereotypes *ejb3Remote* is found, e.g. CustomerBean. Of course it is possible to explicitly model the business interface of a session bean, e.g. CommunicationInterface.

Session beans are accessed by their business interface. By applying the source code annotation *@EJB* the dependency injection of the EJB container can be used. However, there is no business interface in this model which could be used as data type for the attribute *bb*. In such cases the software developer uses the session bean itself as data type and GeneSEZ converts it to the corresponding business interface data type, *IBankBeanLocal* for *bb* (see listing 2).

```

@Remote
@Stateful(name = "CustomerBean")
public class CustomerBean implements ICustomerBeanRemote {
    @EJB
    private MailBeanInterface cb;
    @EJB
    private IBankBeanLocal bb;
    private Bank selectedBank;
    @PersistenceContext(type = EXTENDED)
    private EntityManager entityManager;
    public CustomerBean() {
    }
    //operations [...]
}
  
```

Listing 2. source code of class CustomerBean

In order to combine EJB and JPA the model annotation *ejb3SessionBean* owns the tagged value *entityManager*. By default an attribute of data type *EntityManager* with a normal persistence context is added to a stateless session bean, e.g. BankBean. According to the context session

bean no information beside *ejb3SessionBean* has to be provided by the software developer. If there is no need for an entity manager, the modeler can simply choose the enumeration literal *None* (*CommunicationBean*). The default entity manager for a stateful session bean has an extended persistence context (see listing 2).

## 5. Benefits and experiences

The introduced approach has the following benefits for the software developer:

- reduced modeling effort using configuration by exception and summarizations
- in best case only one model annotation is needed to create an entity or session bean
- annotate model elements almost like using Java annotations, no extra training for software developer necessary
- modeling effort is reduced by combination of profiles
- level of detail of the model adjustable according to development phases
- supports agile development like prototyping or Scrum

The two profiles have been used in practical projects with different scales. Smaller projects contain about 60 to 90 classes to implement the business logic for a regular website. The reference project with about 150 classes supports students in administration and management of collegiate organisations and the coordination of sports groups. The project is in beta phase and about 250 students work with it. About 5000 students will have access to the final version.

Another application field is education. The profiles in combination with GeneSEZ have been applied in practical exercises. In the first exercise a rudimentary model is created which can be configured in more detail in further exercises. The code generation reduces programming effort, avoids demotivating errors and allows a more detailed introduction to the technologies.

## 6. Related Work

In [7] a UML profile for JPA is introduced. The profile is also based on stereotypes but is more complex and focusses only on persistence layer. The realization is completely different. Furthermore any experiences in applying the MD-JPA approach to real world examples are missing.

The fornax platform [8] provides several cartridges for code generation based on openArchitectureWare and the UML meta model which increases the effort for model-to-model transformations, see [5]. The JPA and EJB cartridge are still under development and functionally rudimentary. The Hibernate cartridge creates XML based configuration

files instead of annotations and provides only little support for the modeler to reduce modeling effort.

AndroMDA [9] uses a Java implementation of UML as meta model for code generation. For meta model adjustments and information hiding the concept of so called meta facades based on the facade pattern is used. AndroMDA provides several cartridges amongst others Hibernate and EJB. A cartridge for JPA is not available.

## 7. Conclusion and further work

The paper shows an approach to develop Java EE enterprise applications with minimum modeling effort and minimal technology specific information using model annotations. The introduced JPA and EJB profiles allow software developers with different knowledge levels a comfortable and efficient way of modeling the persistence and business logic. By adopting and refining the concept of configuration by exception the software developer can leave out information indirectly available in the model or by default values. This sharpens the business view. Nevertheless the software developer can configure the technologies in every detail. The default behavior can be easily adapted to new experiences and customized to company specific settings.

In the future development the JPA and EJB profiles will be adjusted to the new Java EE6 specification[4]. To reach the overall goal, i.e. modeling whole enterprise applications with minimum effort, the developing of profiles for JSF, Seam[10], context and dependency injection (CDI), Dependency Injection for Java and Bean Validation[4] have already been started. Among other things these developments will allow us to create new source code annotations which can also be used as model annotations.

## References

- [1] Stephen J. Mellor, Scott Kendall, Axel Uhl, and Dirk Weise. *MDA Distilled*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 2004.
- [2] Markus Voelter Tom Stahl. *Model-Driven Software Development - Technology, Engineering, Management*. Wiley Publishing, Inc. USA, 2006.
- [3] Homepage of the object management group (omg). <http://www.omg.org/>.
- [4] Homepage of java enterprise edition. <http://java.sun.com/javaee>.
- [5] *A Pragmatic UML-based Meta Model for Object-oriented Code Generation*, 2009.
- [6] Homepage of openarchitectureware. <http://www.openarchitectureware.org/>.
- [7] *MD-JPA Profile: A Model Driven Language for Java Persistence*, 2009.
- [8] Homepage of the fornax platform. <http://fornax-platform.org/>.
- [9] Homepage of andromda. <http://andromda.org/>.
- [10] Homepage of the seam framework. <http://www.seamframework.org/>.

# A documentation approach for the self-adaptive system design

Wenhui Zhu

*Software Quality Research Laboratory  
University of Limerick, Co. Limerick, Ireland  
Software Institute, School of Electrical Engineering and  
Computer Science, Peking University, Beijing, China  
zhuwh04 at sei dot pku dot edu dot cn*

David Lorge Parnas

*Software Quality Research Laboratory  
University of Limerick, Co. Limerick, Ireland  
McMaster University, Canada  
David dot Parnas at UL dot ie  
parnas at mcmaster dot ca*

## Abstract

*During the self-adaptive software system design, one of the deficiencies of past research is that past behaviors are always applied to predict future behaviors. As the past is not always a good predictor of the future, such strategies may cause poor performance while the behavior changes. In this paper, we introduce a document-driven approach for developing a self-adaptive software system to enhance its qualities such as performance. In our approach, how to use precise mathematical documentation to build adaptive systems is shown, and we also offer the documents that are used to bind on developers. As a result, this approach ensures that the adaptive behavior is based on reliable information rather than heuristics. Especially, we apply the "plug-and-play" idea; therefore our approach is a good solution for situations where new versions of components may be "plugged into" an existing system and that system must adapt itself to the new component. To demonstrate two different possibilities, we illustrate our approach by using two case studies.*

## 1. Introduction

Self-adaptation is used to guarantee the satisfaction of a system's functionalities during the environmental changes or requirement variations. A self-adaptive software system evaluates its own behaviors and changes behaviors when the evaluation indicates that it is not accomplishing what the software is intended to do, or when a better performance is possible [2]. Due to the increasing functionalities of modern software systems, the need of self-adaptive software appears to be growing [1], and how to build a self-adaptive software system causes very much attention.

Many approaches have been proposed to build self-adaptive systems. They can be generally classified into

conceptual methods and implementation methods. All conceptual methods and some of implementation methods focus on enhancing the system's functionalities by introducing self-adaption; however, they use past behavior to predict future events, which is an inherently unreliable approach since by the nature it is basically heuristic. The other part of implementation-based methods aims at finding a better self-adaptive functionality or implementation rather than using self-adaptive features for system's quality improvement. In this paper, we want to improve the quality of original system functionalities by self-adaptation. We propose a document-driven approach that uses guaranteed information rather than relying on heuristics and statistics.

Documentation should systematically and precisely describe the details of the system analysis and design. Each document must bind those who carry out the subsequent design unless those who accepted the original document agree to change it. Good document helps programmers to divide a system into modules, to specify interfaces, and even to describe the variables of the system. Our document-driven method helps engineers to extract the essential elements of a self-adaptive system and overcome the problem of uncertainty during the system design. This paper also highlights how to integrate the documentation technique with self-adaptation. As a result, our approach can assist software developers to build a self-adaptive system.

Our approach has four major activities in the design phase. Each activity obtains, produces or modifies one or more associated documents. Then, based on those documents, adaptation will be applied to reach the optimized results during implementation.

To clearly illustrate this method and be practical, we assume the system to be built has limited resources and system performance has to be improved by self-adaptation. According to our examples, only performance related quantifiable attributes are

considered when the adaptation objectives are set, e.g., lowest time cost or the smallest space. This constraint allows those attributes to be traced in the document, evaluate their value when necessary, and use them in the implementation. In addition, our examples, only two types of self-adaptation are presented: (1) a system adjusts some part according to the other part that has changed; (2) a system has several candidate components or modules. Without changing the partial system, it chooses the most suitable components or modules to achieve a pre-defined criterion (e.g. the shortest time). Although the demonstration is simple, our ultimate objective is to extend this methodology to a common adaptive control system implementation.

The rest of the paper is arranged as follows. Section 2 is a brief introduction to related research work by other groups. Section 3 introduces the document-driven approach. Section 4 presents the details of the proposed method. Section 5 analyzes the experiment results. A discussion of future work is in section 6.

## 2. Related Work

Various architecture-based adaptation frameworks have been proposed during the past few years [5][6][7]. The objectives of these approaches are formalism and modeling, and adaptation mechanisms, which include Rainbow framework [5], Knowledge-Based Architectural Adaptation Management approach (KBAAM) [6], Software Architecture based Self-Adaptation (SASA) [7], a reflective architecture by Walter et al [13] and ArchStudio for self-adaptation of C2 style system [8]. Moreover, different domain also developed their corresponding dynamic framework, such as Darwin [9][19], ArchWare [12], Weaves [18], Willow [17], CASA [16], and CR-RIO [23]. In addition, some other perspectives focus on autonomic implementation, such as database optimization [21], network server provisioning [22], and workload optimization in web servers [23].

Even though previous self-adaptation research suggested various solutions, most approaches assume that a prediction can be made correctly based on past behaviors. Our approach focuses on how to build up the specification of the status of the available adaptation options. By reading such specification at runtime, the system is able to decide which adaptation is chosen in order to gain certain objectives.

## 3. Documentation

### 3.1 Design Documents

We use three kinds of documents in the analysis and design phase: Module Guide, Module Interface Specification and Module Internal Design Document [10]. The details of them are shown below.

### 3.1.1 Module Guide (MG)

This document helps the programmers to find the modules that are relevant to them. It can be written using tables (see Table 1) [14], and includes modules within a hierarchy - the highest level (system) to the lowest level (individual module). The Module Guide table contains each module's name, the design descriptions that is hidden in the module, and the sub-modules' names. It provides the reviewers and maintenance-programmers what programs will be affected by a proposed change.

**Table 1. Module Guide Table**

Module Name	Secret Hidden	Possible Changes	Sub-Modules
-------------	---------------	------------------	-------------

### 3.1.2 Module Interface Specification (MIS)

The Module Interface Specification defines the interface between modules. It tells programmers what is needed to build into a module and also lets other developers know what can be expected of a module. The document contains information for both those who build the module and those who use it.

### 3.1.3 Module Internal Design Documentation (MIDD)

A MIDD contains internal design decisions. This decision information is known by those who review the systems. This document also offers the programmers an overview of the code and helps those who maintain the system to modify the source code more easily.

The three documents above are based on the document-driven approach, which is elucidated in the paper by Parnas [11][15].

### 3.2 Trace Function Method (TFM)

The TFM is a way to describe MIS based on earlier algebraic, axiomatic and trace-based approaches. The TFM is intended to be used as a description of components with a hidden data structure that should not be mentioned in a specification. More detailed information can be found in [11].

## 4. Design Process

Figure 1 shows the whole design process. During the analysis and design phase, MG, original MIS, parameterized MIS and MIDD are created along with the design activities. Since TFM can help us to precisely address behaviors of variables that are specified by MIS and the adaptation mechanism requires the information of the target components, at this stage, TFM is applied to collecting the necessary information for self-adaptation development. Meanwhile, the specification parameters of the MIS(s) are also created, so is the adaptation strategy, which contains the instructions about which MIS(s) and parameter(s) should be read. Finally, a system is implemented based on the design. Optimizing the

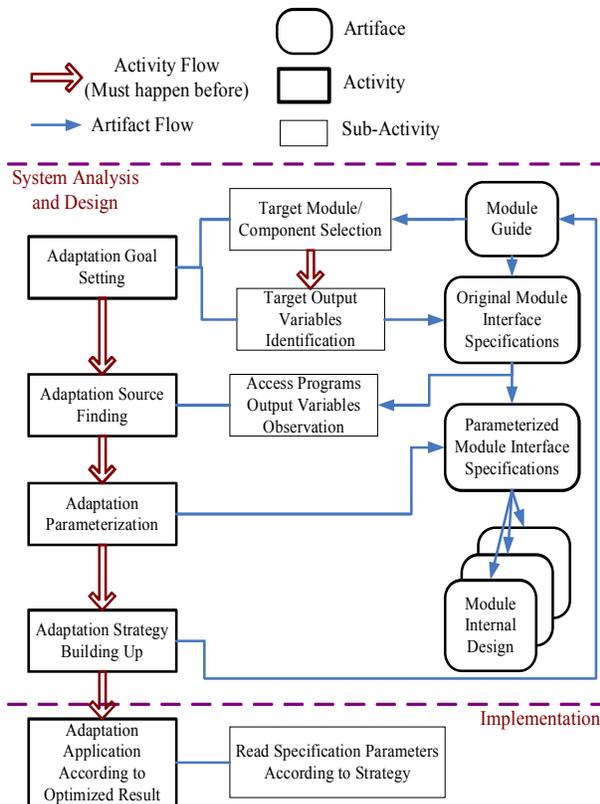


Figure 1. Design process

results that come from the pre-defined specification and the strategy will perform the self-adaptation. (Arrow in Figure 1 means “must happen before”) We classify the modules of the system in two categories – “triggering module” and “non-triggering module”. Triggering modules include the modules or components whose changes will cause the adaptation. Non-triggering modules are those modules or components whose behavior will not cause any adaptation.

Our cases realize two kinds of self-adaptations. One adaptation shows how non-triggering modules are affected by the change of triggering ones. The other one indicates that triggering modules will not be affected, instead, for the sake of the adaptation target, triggering modules just choose one of change options that the modules has. However, both types have to read the specification parameter values and apply the adaptation according to the adaptation strategy.

#### 4.1 Adaptation Design

Among all MIS tables, only access program, output variables and specification parameter tables (see table 2, 3, and 4) are used to build the system’s adaptation mechanism. Variables filled in each table provide developers significant system design information and clues to trace according to the pre-set adaptation target.

Since the value of each output variable described by TFM is a function of the history of the events – not like simply listing the program variables, variable value change traceability can be easily performed by TFM. Those variables listed in the abbreviated event descriptor in the access program table (Table 2) are the one whose values are changed by the events (in this case, the access programs). Thus access program table helps us easily to find the events that affect the variables according to their relations.

To explain this tracing procedure, we can take a simple example, i.e., our adaptation goal is to make the system perform certain function within minimum time. Therefore, any variable associated with “time” that is not negligible becomes our clue. First, each MIS’s output variable table is examined and those MISs that have “time” variables (see Table 3) are located. Then, by looking at the access program table of each selected MIS, those programs where “time” is inside its abbreviate event descriptor are detected, which means those access programs change the value of “time”. Table 2 shows that PUSH, POP and SWAP programs contain the output variable “time”. After those time-related access programs are found, the time related parameters can be located in the specification parameter table; that is, “poptime”, “pushtime” and “swaptime” (see Table 4).

After all the needed information is collected, we can build up the adaptation strategy that uses associated component’s specification and parameters to tell the system how to achieve the objective function according to the runtime inputs.

Table 2. Access program table

Program Name	Value1	Value2	in	Abbreviated Event Descriptor
PUSH			<string>	(PGM: PUSH, 'in', 'top', 'time', 'depth', 'exc')
POP				(PGM: POP, 'top', 'time', 'depth', 'exc')
TOP		<string>		(PGM: TOP, 'Value2', 'top', 'exc')
DEPTH	<integer>			(PGM: DEPTH, 'Value1', 'exc')
SWAP			<integer>	(PGM: SWAP, 'top', 'time', 'exc')
ELEM			<integer>	(PGM: ELEM, 'elem', 'exc')

Table 3. Output variables table

Variable Name	Type	Data Type component	Range
time	<integer>	{0,1,...,9}	{0..4294967295}
top	<string>	{ASCII 9,..., ASCII 122}	length(top)€{1..width}
depth	<integer>	{0,1,...,9}	{0..4294967295}
elem	<string>	{ASCII 9,..., ASCII 122}	length(elem)€{1..width}
exc	{none, range, depth, empty, size, forbidden}		
Value1	<integer>	{0,1,...,9}	{0..4294967295}
Value2	<string>	{ASCII 9,..., ASCII 122}	length(Value2)€{1..width}

**Table 4. Specification parameter table**

Parameter Name	Type	Value	Range
width	<integer>		{1...1073741824}
maxDepth	<integer>		{1...65536}
popTime	<integer>		{1...3600000000}
pushTime	<integer>		{1...3600000000}
swapTime	<integer>		{1...3600000000}
availSwap	<boolean>		{true, false}
availElem	<boolean>		{true, false}

**4.2 Adaptation Strategy Application**

The adaptation strategy is applied after the program has been implemented but before algorithms run. It states the MIS parameters and system inputs which adaptation needs, and has three major steps to accomplish the procedure.

- Read the corresponding MIS parameters and input data. From the example in section 4.1, the system reads the MIS(s) of triggering modules to get the minimum time cost; thus, the value of pop time and push time and the input (e.g.: string size).
- Evaluation according to the Strategy. To evaluate the performance of system (like time cost), objective function is setup and collects data as the input during the calculation.
- Get the optimized solution. Since we have the calculation results that show the performance under different circumstances, it is easy to tell which one is the proper solution.

**4.3 Summary of Adaptation Design**

The above process indicates a prior condition; therefore, those observed attributes (i.e., time) have to be expressed by parameters and variables in the module and component. Those variables should also be essential and existing variables of the program so they cannot be created just for fitting the adaptation (see time in section 4.1 example). This implies that our approach is a very precise method for design a system that self-adaptation to increase quantifiable qualities.

**5. RESULT**

Our case studies only consider the attribute “time”. In each case study, we will illustrate our implementation according to the format – case scenario, objective, evaluation and result.

These two case studies also represent two types of self-adaptations – adaptation inside the component and candidate component selection.

**5.1 String Reversal Case**

**5.1.1 Scenario**

This case mimics the computer hardware. The whole system is like a computer and the stack is like a plug-and-play device. When the stack is “plugged” into the system, the system uses the stack’s specification that has been read beforehand and decides

how it would perform. The stack receives the string with certain speed. The system uses the stack to reverse the string. Since the stack can be different “shapes”, i.e., “tall thin” shape or “short fat”, we need to pack the string into packages accordingly and use the packages as the element to be manipulated in the stack.

**5.1.2 Objective**

The goal of this case is to get the minimum time cost of string reversal. Five elements, pop time, push time, stack width from the stack specification, and the arriving speed from the string, are used to calculate the time cost and they can directly obtained. However, we have to compute the packing time according to different package sizes.

**5.1.3 Evaluation**

When the time cost is computed, two cases may happen due to different string arriving speed.

- (1) Fast:  $T_c = n \cdot (T_{ph} + T_{pk} + T_{pp})$
- (2) Otherwise, slow:  $T_c = n \cdot (T_{wp} + T_{ph} + T_{pk} + T_{pp})$

Where  $T_c$  is time cost,  $T_{ph}$  is push time,  $T_{pk}$  is packing time,  $T_{pp}$  is pop time,  $T_{wp}$  is package waiting time and  $n$  is the number of packages.

The goal of the evaluation is to look for the best package size for the application.

**5.1.4 Case Result**

The testing machine is an iMac G4 with a 700 MHz processor. Table 5 shows the fixed packing time that are used in the experiment.

To get a more convincing result, we repeat 10 times and each time we run the experiment on three stacks. Even though every time our experimental values are slightly different, the adaptation judgment is the same. Hence, we just provide one of our results (Table 6, Table 7, Table 8 and Table 9) to illustrate the calculation.

**Table 5. Fixed packing time of each length**

Length (Char*)	Fixed Packing Time(ms)
2	0.0172
4	0.0094
8	0.013
16	0.0268
32	0.1066
64	0.2158

\* 1 Char = 4 bytes

**Table 6. Information got from MIS**

	Stack		
	No.1	No.2	No.3
Width (Char)	128	32	64
maxDepth (Number)	512	256	1024
popTime (ms)	0.005	0.2	0.0005
pushTime (ms)	0.008	0.05	0.0005

**Table 7. Information got from Input**

	Test Data		
	No.1	No.2	No.3
String Size (Char)	3831	2813	4594
String speed (Char/ms)	9	5	6

**Table 8. Comparative Values**

Package Size (Char)	Reversal Time (ms) of Test Data		
	No.1	No.2	No.3
2	31.0617	216.0355	13.3151
4	14.4275	109.4961	1.5929
8	9.3649	55.5625	4.1621
16	5.9918	31.1026	4.3044
32	7.3386	21.9744	5.0486
64	9.7449		9.9680
128	17.6581		

**Table 9. Minimum Time Cost**

	Test Data		
	No.1	No.2	No.3
Minimum Time Cost (ms)	5.9918	21.9744	1.5929
Best Package Size (Char)	16	32	4

From Table 5 and 6, the total time of pop and push of No.1 test is 0.013 milliseconds (0.005+0.008=0.013) which is approximately equals to the time cost for reversing a string of size 2 char (0.0172 milliseconds). However, if we compare time cost of a 32 and a 16 char string, the 32 char string did not consume twice times as what the 16 char string did. Instead, the difference is approximately 5 times. As a result, adaptation suggested using the 16 char as the pack size.

Similar to No.1 test, the sum of pop time and push time of no.2 is 0.25 milliseconds, which is much bigger than the reversing time cost of 2 char string (0.0172). Hence, this consequence indicates the biggest size – 32 char is the best candidate pack size for the stack.

The most right column of Table 6 shows that the sum of pop and push time of No.3 test is 0.001 milliseconds. This is much smaller the 2 char string’s reversing time cost (0.0172). Therefore, (see Table 5) the best choice of pack size is 4 char since no matter how small the sum of pop and push time is, the time need of 4 char string (0.0094) is smaller than 2 char string (0.0172).

## 5.2 Swap Stack Case

### 5.2.1 Scenario

The second case assumes the system is given various stacks as candidates and each of them has its own characteristics. For example, some of them have a bigger "instruction set" which means the stack has extra commands besides common ones like pop and push, i.e., a “swap” command that would exchange the top two elements. For the stack without the “swap”

command, two pop actions plus two push actions will fulfill the job if we want to swap top two elements. In our experiments, the system needs to choose one of stacks to store strings or numbers under the condition that top element is kept the smallest.

### 5.2.2 Objective

This case also use minimum time cost as the objective; however, to reach the goal, several different parameters have to be provided. To check if the stack has enough space to put the strings/numbers into, we need width and depth from the specification parameters. Besides, pop time, push time, swap time, the availability of swap instruction, and availability of element instruction (element instruction returns the value of the second top element) are also specification parameters. They all are required for satisfying the objective.

### 5.2.3 Evaluation

There are two kinds of time cost calculations for each stack (Ts). Both depend on the availability of the swap instruction and element instruction.

(1) They are available:  $T_s = n * T_{ph} + m * T_{sp}$

Where,  $T_{sp}$  is the swap time,  $n$  is the number of strings,  $m$  is the swap times during the whole storage.

(2) Not available:  $T_s = n * T_{ph} + 2m * (T_{ph} + T_{pp})$

If the swap instruction and the element instruction are not available, the stack has to pop the top two elements and then push them into the stack with reversal sequence.

After getting all the parameters’ values from each stack’s document, the system will evaluate each stack’s time cost and then choose the best-fit stack based on evaluation.

### 5.2.4 Case Result

From input, we know that the string group consists of 100 strings, and each one is 64 char long (see Table 10). Obviously, stack 3 is too small for the group since the stack width 8 is much shorter than the string length. Comparing stack 1 with stack 2 (see Table 11),  $300n + 900m$  microsecond is shorter than  $1000n + 2000m$  microsecond no matter what  $n$  and  $m$  are. Thus we choose stack 1 as the best option to store the strings.

**Table 10. Information got from Input**

Each String Size (Char)	Number of Strings
64	100

**Table 11. Information got from the MIS**

	Stack 1	Stack 2	Stack 3
Width (Char)	1024	128	8
maxDepth	256	128	2096
popTime (us)	300	1000	200
pushTime (us)	300	1000	200
swapTime (us)	900	1200	800
availSwap	true	false	true
availElem	true	false	true

**Table 12. Result of the Adaptation**

	Stack 1	Stack 2	Stack 3
Result	Minimum time cost	Instruction not available	Not enough space

## 6. Conclusion & Future Work

Our main contribution is to propose a systematic design process that uses precise documentation to build up a system with self-adaptation in order to improve past research's deficiencies. Thus, this approach avoids the reliability problem of prediction based on past data. Our case studies demonstrate two different adaptation mechanisms: component self-adjustment and component candidate selection.

However, our implementation still needs to be improved by considering more complicated situations. For example, in future research, since we only chose time attribute in our cases, we can include other attributes, other than time. For those hard-to-be-quantified attributes, such as usability, should also be considered. We also need to turn our attention to problems in which there may be conflicting goals. Finally, we would like to apply to a practical system by putting all possible factors together and validate the results.

## 7. Reference

- [1]. Peter Norvig, David Cohn, "Adaptive software", PC AI, Volume 11, Number.1, January 1997, Page 27-30.
- [2]. Robert Laddaga, "Self Adaptive Software SOL BAA 98-12", DARPA Broad Agency Announcement on Self-adaptive Software, DARPA/ITO, January 1998.
- [3]. David L. Parnas, "Requirements Documentation: A Systematic Approach", SQRL research report, University of Limerick, <http://www.csis.ul.ie/Research/ParnasLectures/Default.HTM>
- [4]. J.S. Bradbury, J.R. Cordy, J. Dingel, and M. Wermelinger. "A survey of self-management in dynamic software architecture specifications". In WOSS '04, pp. 28-33, ACM, New York, NY, 2004.
- [5]. David Garlan, Shang-Wen Cheng, An-Cheng Huang, Bradley Schmerl, and Peter Steenkiste, "Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure," Computer, Vol.37, No.10, Oct 2004,
- [6]. John C. Georgas and Richard N. Taylor, "Towards a Knowledge-Based Approach to Architectural Adaptation Management", 2002 ACM SIGSOFT Workshop on Self-Managed Systems (WOSS'04), Newport Beach, California, October 2004.
- [7]. Yali Zhu, Gang Huang, Hong Mei: "Quality Attribute Scenario Based Architectural Modeling for Self-Adaptation Supported by Architecture-Based Reflective Middleware". APSEC, November 2004.
- [8]. E.M. Dashofy, A. van der Hoek, and R.N. Taylor. Towards architecture-based self-healing systems. In WOSS '02, pp. 21-26, ACM Press, November 18-19, 2002.
- [9]. J. Magee and J. Kramer. Dynamic structure in software architectures. In FSE 1996, pp. 3-14, 1996.
- [10]. David L. Parnas, Madey, J., "Functional Documentation for Computer Systems, Engineering" published in Science of Computer Programming, Volume 25, Number 1, October 1995, Page 41-61.
- [11]. David L. Parnas Sergiy A. Vilkomir, "Precise Documentation of Critical Software", HASE'07, Volume 00, September 2007, Page 237-244.
- [12]. R. Morrison, D. Balasubramaniam, F. Oquendo, B. Warboys, and R.M. Greenwood. An active architecture approach to dynamic systems co-evolution. In ECSA, LNCS 4758:2-10. Springer, September 24-26, 2007.
- [13]. Walter Cazzola, Ahmed Ghoneim, and Gunter Saake. "System Evolution through Design Information Evolution: a Case Study". IASSE, 145--150, July 2004.
- [14]. Kathryn L. Heninger. "Specifying software requirements for complex systems: New techniques and their applications". IEEE Transactions on Software Engineering, SE-6(1):2--12, January 1980.
- [15]. Ying Jin, David Lorge Parnas. "Defining the meaning of tabular mathematical expressions". Science of Computer Programming, 2010
- [16]. A. Mukhija and M. Glinz. A framework for dynamically adaptive applications in a self-organized mobile network environment. In ICDCSW '04, pp. 368-374, 2004.
- [17]. A.L. Wolf, D. Heimbigner, A. Carzaniga, K.M. Anderson, and N. Ryan. Achieving survivability of complex and dynamic systems with the Willow framework. In Proc. of the Working Conf. on Complex and Dynamic Systems Architecture, December 12-14, 2001.
- [18]. M.M. Gorlick and R.R. Razouk. Using Weaves for software construction and analysis. ICSE, pp. 23-34, May 1991.
- [19]. D. Sykes, W. Heaven, J. Magee, and J. Kramer. From goals to components: a combined approach to self-management. In Proc. of the ICSE SEAMS' 08, pages 1 - 8, 2008.
- [20]. G. Valetto and G. Kaiser. Using process technology to control and coordinate software adaptation. In Proc. Of the ICSE ' 03, pages 262 - 272, Washington, DC, USA, 2003. IEEE Computer Society.
- [21]. V. Markl, G.M. Lohman, and V. Raman. LEO: An autonomic query optimizer for DB2. IBM Systems Journal, 42(1):98-106, 2003.
- [22]. L.W. Russell, S.P. Morgan, and E.G. Chron. Clockwork: A new movement in autonomic systems. IBM Systems Journal, 42(1):77-84, 2003.
- [23]. B. Solomon, D. Ionescu, M. Litoiu, and M. Mihaescu. A Real-Time Adaptive Control of Autonomic Computing Environments. In Proc.U2TS'2006, 2006.

# Designing Aspects with Use Cases: A Case Study

Junhua Ding, Christopher R. Westbrook, M. N. H. Tabrizi

Department of Computer Science  
East Carolina University  
Greenville, NC, United States of America  
{dingj, crw0620, tabrizim}@ecu.edu

**Abstract**— A practical and useful Aspect-Oriented Software Development (AOSD) approach should have the capability to smoothly accommodate existing successful software development methods. In this paper, we propose an approach for designing aspects based on use case modeling and object-oriented methods. Our approach establishes a guideline for early identification of aspects from use case models, and the incorporation of aspects into object-oriented methods. We introduced domain modeling with the extension of aspects into the AOSD to bridge the gap between use case modeling and object-oriented design. We have devised how to model aspects in the conceptual class model, specify aspect interactions through system sequence diagrams, define the weaving of aspects and their base modules in the operation contracts, and modularize the aspects during design modeling. As compared to existing AOSD approaches, our approach is more practical and easier to use for not only the development of new software systems, but also the maintenance of legacy software systems. The approach is illustrated through analysis and a case study of an in-house archival, data extraction, assessment, and preservation system.

**Keywords**- *aspects, use cases, object-oriented design, domain modeling*

## I. INTRODUCTION

Crosscutting concerns are software units that are scattered and tangled throughout multiple modules in software [7]. Aspect-Oriented Software Development (AOSD) focuses on identification and modularization of the crosscutting concerns across a software system into manageable modules [18]. The identification and modularization of these crosscutting concerns, referred to as aspects in AOSD, are each essential tasks during the software development life cycle. In order to successfully incorporate aspects into the software design, a guideline must exist from initial requirements analysis to software design that provides a traceable method of forming aspects. Most existing AOSD approaches do not work well with current successful development approaches like use case modeling [10] and object-oriented methods [3]. Legacy experience or system models cannot be easily reused in those AOSD, and developers cannot be easily adapted to the new approach.

A practical and useful AOSD approach should incorporate aspects into existing methods with minimal changes. Use case modeling has been widely used for software development as an easy to use requirements analysis and specification approach; in addition, object-oriented design is the most successful software design approach. It is a natural idea to create an AOSD based on use case modeling and object-oriented

methods. Use case modeling is used by a wide variety of project participants including software developers, business analysts, project managers and even customers.

Introducing aspects explicitly into use case modeling only makes the requirements analysis and specification more complex. Since aspects are a design concept, it is difficult to understand use case models with aspects since an actor never considers aspects. It is also very difficult to analyze the use case model with aspects since aspects are isolated from use cases as separate modules. Composing aspects along with the use cases is not easily automated due to the informal nature of the specification of use cases. Use case modeling should not be changed with the inclusion of aspects for AOSD just like use case modeling has not been changed with the inclusion of objects in object-oriented methods. Instead, a systematic method and tool set are needed to find aspects in use case models [2]. Then those aspects found are transformed into aspects in design models.

Because the gap, in aspect development, between use case modeling and object-oriented design is fairly large, the aspects identified in use case modeling cannot be smoothly modeled in the object-oriented design. In this paper, we introduce domain modeling with the extension of aspects into the AOSD to bridge the gap. We have devised an iterative approach that establishes the guideline for early identification of aspects from use case models, modeling aspects in the conceptual class model, specifying aspect interactions through system sequence diagrams, weaving of aspects to base modules in the operation contracts, and modularizing the aspects during design modeling. As compared to existing AOSD methods, our approach includes the advantages of aspect-oriented, use case modeling and object-oriented methods. The approach is more practical and easier to use for not only the development of new software systems but also the maintenance of legacy software systems. The approach is illustrated through analysis and a case study of an in-house, documentation management and computing system – the Archival Data Extraction, Assessment, and Preservation (ADEAP) system, which is a role-based system with the purpose of providing a valuable resource for researchers to archive and analyze very large datasets of historical documents, primarily hand written documents. Researchers use the system to analyze handwritten documents from user submitted algorithms.

The contributions of this paper can be summarized as the following:

- Development of an approach for designing aspects with use cases. Software requirements are specified as use case models, and an approach is developed to find aspects. Software design is modeled as object-oriented models with aspects, and domain modeling is extended with aspects to facilitate the transformation from the use case model to the object oriented design.
- Introduction of the association mechanism into aspects in the software domain model so that the composition of aspects with analysis classes can be rigorously defined. Composition of aspects with analysis classes is not clearly defined in many existing AOSD methods since analysis classes do not define operations yet [17]. Through introducing the association mechanism into aspects in the domain model, the identification, modularization and composition of aspects can be smoothly transformed from use case models into software design models.
- Illustration of the effectiveness of the approach through a case study of a real world application —the ADEAP system.

The rest of this paper is organized as follows: Section 2 presents the approach for analyzing aspects, and section 3 discusses the design of aspects. Section 4 discusses related work and section 5 describes the summary and future work.

## II. ASPECT-ORIENTED ANALYSIS

In this section, we begin presenting our iterative approach to incorporate aspects across the phases of the software development process. First, we identify the crosscutting concerns from the use cases. Once the aspects are discovered, they are then conceptualized in the domain model. Next, the aspect behavior is merged into the system sequence diagrams and weaved to methods in the operation contracts. In Section III, this aspect analysis is used to create aspects at the design level.

### A. Identifying Crosscutting Concerns in the Use Case Models

Use cases contain the behavior of a system and it's from this behavior that crosscutting concerns can be identified. When examining the use cases, the crosscutting concerns are any behaviors, or actions, that are described as occurring in multiple parts of the system. In AOSD, the early discovery of aspects is essential to their traceability and management [15]. Developers need a way to transform uses cases from text to an analysis resource capable of isolating the crosscutting concerns from the non-crosscutting behaviors. Each use case is comprised of a main success scenario and one or more alternative scenarios; consequently, these scenarios create different system event flows. One way to represent these flows is to construct flowcharts that consist of the actions and decisions occurring within the different scenarios. By using flowcharts, the behaviors demonstrated within the use cases are separated into individual processing blocks and identifying the crosscutting behaviors becomes a simple task of recognizing the processing blocks that demonstrate a behavior existing at multiple locations.

Finding crosscutting concerns can be summarized as the following:

- Construct one or more flowcharts for each of the scenarios in the use cases.
- Manually examine the flowcharts to see which processing blocks contain a reoccurring behavior.
- From the reoccurring behaviors identified, establish an aspect for each behavior that is difficult to manage without separating it into its own module.

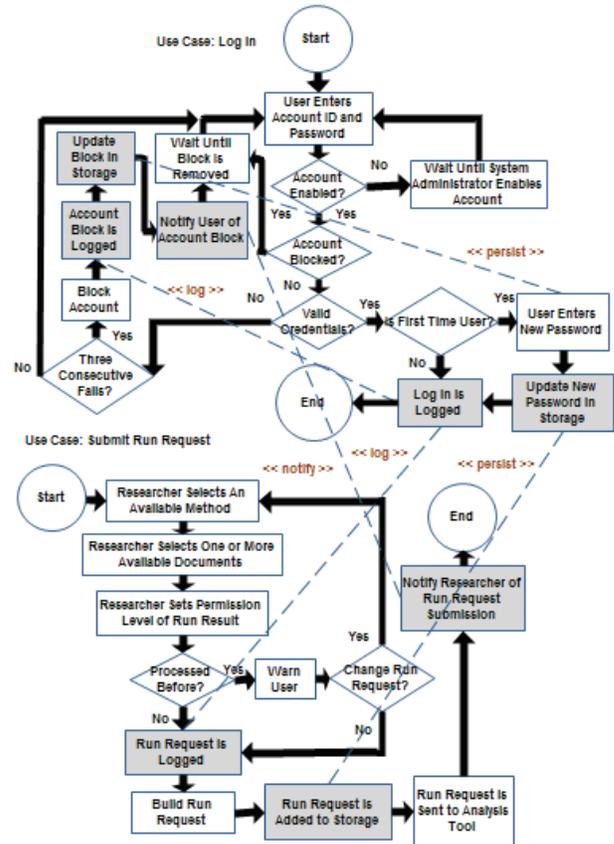


Figure 1. Identification of crosscutting concerns from use cases

In Figure 1, the *Log In* and *Submit Run Request* use cases are represented as flowcharts in order to discover some of the major crosscutting behaviors impacting the system. The shaded rectangles signify the processing blocks in the flowcharts that demonstrate a reoccurring behavior. If a behavior exists in more than one location, a connection is made via a dashed line along with a label to identify the behavior as a crosscutting behavior. From the flowchart diagrams, three crosscutting behaviors are identified: *notify*, *persist*, and *log*. A crosscutting behavior may exist in the flows of multiple use cases, multiple flows of a single use case or even just a within a single flow of a use case. For instance, the log behavior can be classified as a crosscutting behavior since it occurs twice in the flow of the *Log In* use case, but it also crosscuts the multiple use cases since it occurs in the *Submit Run Request* use case. From the discovered crosscutting behaviors, three aspects are identified: *Notification*, *Persistence* and *Logging*.

Although it is important to identify as many of the crosscutting behaviors as possible so that an aspect is not overlooked, it may not be necessary to designate all of the crosscutting behaviors as aspects. A behavior that occurs only a few times, especially in larger systems, may be manageable enough to not require its separation into an aspect. The goal of AOSD is to isolate, into modules, the duplicate portions of the system that make it difficult to manage, maintain or evolve and a crosscutting concern that is minimally crosscuts the system may not need to be an aspect.

Our approach for finding the crosscutting behaviors manually is sufficient for identifying the major aspects of a system since the use case flowcharts highlight the repetitive behavior. The crosscutting behaviors that impact the system the most are the main concern and drive for the creation of aspects. During the early stages of development, some of the crosscutting behaviors may be overlooked or ignored if the behavior does not surface until the future stages of development or because of the inexperience a developer has in recognizing the behavior. This is why our approach stresses the iterative development of aspects to allow developers a chance to revisit previous stages of development in the instance a crosscutting behavior is identified after the initial requirements analysis.

### B. Analysis and Modeling of Aspects in the Domain Model

The domain model, also referred to as the conceptual class model, is a crucial component in connecting requirements analysis and design modeling [13]. Without the domain model, a gap exists between requirements analysis and design modeling. In AOSD, it is important to include aspects as part of the conceptual modeling since excluding them will generate this same gap in the development of aspects. As discussed in [14], aspects cannot be disregarded during the development of the domain model.

The main problem a developer faces when incorporating aspects into the domain model is how to establish the connection between conceptual classes and aspects. Since methods do not exist in the conceptual classes of the domain model, defining the join points needed for a pointcut of an aspect becomes a difficult task. Conceptual classes must be joined to aspects in a way that does not sacrifice the simplicity of the domain model. Any additions to the domain model must avoid introducing new constructs that change the base structure of the domain model or add physical associations that complicate the visual representation.

The first step in resolving this problem is to determine the existence of join points in the domain model. A join point during this phase of development can be defined as the point at which the aspect and the domain model meet. The domain model is comprised of only two main components: conceptual classes and their association to one another. For this reason, a join point relates to the domain model via the conceptual classes and associations. Since the associations belong to the conceptual classes and alone have little meaning, the conceptual classes serve as the key join points.

In order for a conceptual class to be a join point for an aspect, the behavior demonstrated by the aspect must crosscut

the conceptual class. Similar to how the conceptual classes are identified through category lists and by isolating noun phrases existing in the use cases, the conceptual classes that are crosscut by an aspect are identified by utilizing the use cases to create a list of the candidate conceptual classes that demonstrate the behavior of the aspect. For example, in the *Log In* use case, there is a requirement that states, “Anytime a Non-computing Researcher logs in to the system, the action must be logged”. From the use case, *Non-computing Researcher* is identified as a conceptual class that initiates the log behavior; therefore, the *Logging* aspect crosscuts the *Non-computing Researcher* conceptual class.

Although the conceptual classes work as join points, it is often possible to further refine the point at which it connects to the domain model by taking into account the associations of the conceptual classes. A conceptual class may have several associations and it is possible that none or only some of the associations play a part in exhibiting the behavior of the aspect. Discovering if an association has a role in establishing the behavior of an aspect in the domain model once again involves analyzing the uses cases.

The association relationship for conceptual aspects can be summarized as the following:

- Manually examine the uses cases to determine which of the conceptual classes demonstrate a given aspect behavior.
- If a conceptual class does demonstrate an aspect behavior, look at its associations in the domain model to see if they can further refine the join point.
- The association is important to the join point if removing it from the domain model also requires removing an instance of the aspect behavior described in the use cases.

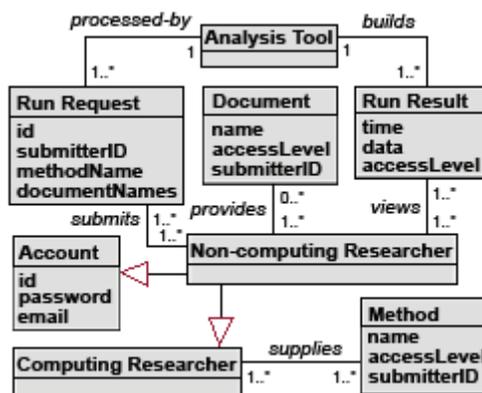


Figure 2. Partial domain model of the ADEAP system

As an example, the domain model in Figure 2 contains a partial view of the conceptual classes and associations of the ADEAP system. During the discovery of candidate conceptual classes demonstrating the behavior of the *Logging* aspect, *Analysis Tool* was identified as one of the conceptual classes crosscut by the aspect. *Analysis Tool* has two associations: *processed-by* and *builds*. From the use cases, whenever the *Analysis Tool* begins building a *Run Result*, the action is

logged. By examination of the use cases, it becomes clear that the *builds* association initiates the *Logging* aspect behavior, and the *processed-by* association is not involved. The *Account* conceptual class provides an example of an aspect behavior existing even though the *Account* associations are not important to the aspect. In the event of an *Account* having its password changed, the action is logged. Through this event, the *Logging* aspect behavior crosscuts *Account* independently of any association that may be related to the conceptual class.

A conceptual class may even require an association to exist in order for it to demonstrate a specific crosscutting behavior. In the case of the *Analysis Tool* conceptual class, the *builds* association it has with the *Run Result* conceptual class exhibits the behavior of the *Logging* aspect. If this association does not exist, *Analysis Tool* no longer involves any action that features the crosscutting behavior of the *Logging* aspect. Removal of the *builds* association from the domain model also omits *Analysis Tool* as a join point for the *Logging* aspect.

Now that the conceptual classes, and their associations, are identified as the join points needed to represent aspects in the domain model, the next step is to diagram the aspects into the domain model in a way that establishes a relationship between the aspects and conceptual classes without complicating or changing the domain model. For this purpose, we introduce the concept of the conceptual aspect, conceptual aspect association and the conceptual aspect pointcut.

*Conceptual Aspect:* A conceptual class stereotyped as an aspect used to represent an aspect in the domain model. It contains a single conceptual aspect association predicate and a single conceptual aspect pointcut predicate.

*Conceptual Aspect Association:* An attribute of a conceptual aspect stereotyped as an association that identifies the associations of the conceptual classes existing within the domain model that exhibit the behavior of the aspect and should be considered as part of the join point. It contains a list of the conceptual classes that have these key associations, each followed by the set containing the names of the key associations. The \* symbol can be used to represent all associations of a conceptual class. Syntax:

```
<<association>> CClass1 {associations list}, ...
```

*Conceptual Aspect Pointcut:* An attribute of a conceptual aspect stereotyped as a pointcut that uses conceptual classes as join points that weave to the aspect. It contains a list of all the conceptual classes that are weaved to the aspect. Syntax:

```
<<pointcut>> CClass1, CClass2, ...
```

Figure 3 shows the *Logging* aspect displayed as a conceptual aspect. To simplify the representation of key associations, \* can be used when all the associations of a conceptual class exhibit the behavior of an aspect. For instance, the {\*} set following *Non-computing Researcher* in the conceptual aspect association is the same as the set {submits, provides, views}. The name of a conceptual class demonstrating the behavior of the aspect should exist in the conceptual aspect pointcut but can be omitted from the conceptual aspect association if it does not have any key associations, such as in the case of *Account*.

Through the example of the *Logging* conceptual aspect, the benefits of our approach, for incorporating aspects into the domain model, are clearly emphasized. The conceptual aspect provides a simple way of relating the domain model to aspects in a way that even works with a preexisting domain model. Developers now have a means to define join points in the domain model without the existence of methods, through the conceptual classes and associations.

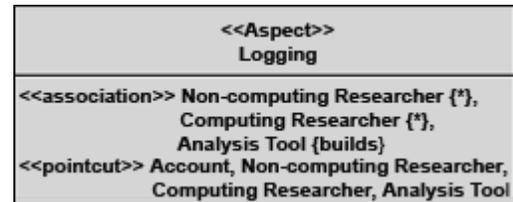


Figure 3. The Logging conceptual aspect

### C. Specifying Aspect Behavior

Use cases and domain modeling are a sufficient way to describe system behaviors but often more detail and description of the behavior is needed before design modeling begins [13]. To provide this additional detail, developers make use of system sequence diagrams and operation contracts. In iterative AOSD, these two development activities are important for the construction of aspects since join points are created in the form of methods. The input and output events captured in the system sequence diagrams serve as the input for the operation contracts and lead to both the creation of design objects and the pointcuts of each aspect.

The process of incorporating aspects into the system sequence diagrams involves two main tasks: 1) identifying which aspects are involved with the current system behavior being modeled, and 2) recognizing at what point during the actions the aspect advice should occur. The first task was already accomplished during the initial discovery of the crosscutting behaviors from the use cases. A system sequence diagram shows the system events for one scenario of a use case; therefore, if the scenario being diagram involves one of the crosscutting behaviors used as an aspect in the domain model, then the aspect behavior is involved in the system event.

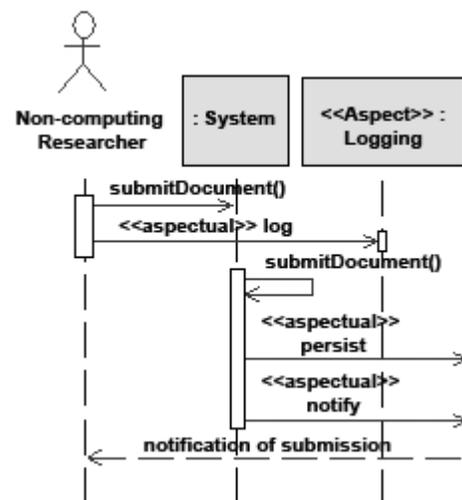


Figure 4. Aspect behavior in a system sequence diagram

Next, determining when the advice of an aspect should occur in the flow of the system events is dependent upon the logical flow of the system events and the preference of the developer. An aspect behavior is dependent on the flow of system events if it must occur at a specific point. For instance, the persist behavior of the *Persistence* aspect is responsible for storing any new or updated information, so the advice of the aspect should occur immediately after any data changes that require storage. Sometimes though, a developer has some freedom in choosing when the advice occurs. The log behavior of the *Logging* aspect records when certain system events are initiated, but the aspect advice can accurately be applied before or after the system event happens.

In Figure 4, some of the aspect behaviors identified for the ADEAP system are modeled into the system sequence diagram that represents the main success scenario of the *Manage Documents* use case. Only the *Logging* aspect is fully shown in the figure, but it should be assumed that the *Persistence* and *Notification* aspects follow the same pattern and the *Notification* aspect returns the “notification of submission” to the *Non-computing Researcher*. To clearly represent the actions handled by the aspects, the “aspectual” stereotype precedes all aspect behaviors. From the system sequence diagram, the advice associated with the *Logging* aspect has been set to occur before the join point that triggered it, but the advice could have been applied after or around the join point. In contrast, the persist behavior of the *Persistence* aspect must wait until the system call to the *submitDocument* method has successfully generated the data for storage.

Recognizing the point at which a piece of aspect advice should occur for a join point is a key step for transforming from system sequence diagrams with aspect behaviors to operation contracts with aspect behaviors. The aspect behaviors presented in the operation contracts provide the majority of the join points needed in the design modeling. Just as the system sequence diagrams help visualize when a piece of aspect advice should occur, the operation contracts help define the pointcuts during the final construction of the aspects.

```

CONTRACT C11: submitDocument
Operation: submitDocument(document: name,
                                accessLevel: boolean)
Cross References:
  Use Cases: Manage Documents
  Aspects: Logging
Preconditions:
- A Computing/Non-computing Researcher is logged in
- The researcher has a method
Postconditions:
- <<Aspectual>> Before: Submission was logged
- An instance dr of DocumentRepository was created
- The parameters document and accessLevel became parameters of dr.submitDocument
  
```

Figure 5. Aspect behavior in an operation contract

Figure 5 shows the introduction of aspect behavior into the *submitDocument* operation contract. The “aspectual” stereotype from the system sequence diagrams remains to identify the aspect behavior, along with a “before” tag to show when the aspect advice should occur. By including the aspect information in the operation contracts, the developer creates an invaluable resource for the upcoming pointcut construction and

also another stepping stone for the traceability and manageability of the aspects.

### III. ASPECT-ORIENTED DESIGN

After the incorporation of aspects into the system sequence diagrams and operation contracts, aspects can now be considered at the design level. In order to model the interaction between aspects and design object, we use sequence diagrams. Since the sequence diagrams requires objects, construction of the sequence diagrams works hand in hand with the construction of the design model [13]. The format for introducing aspects into sequence diagrams follows the same general pattern used for the system sequence diagrams, with only one modification. Now that the design level objects exist, the generalized aspect behavior used in the system sequence diagrams is now replaced by join points that allow the weaving of operations to aspects.

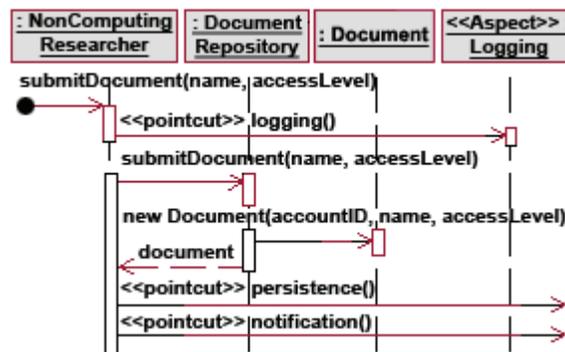


Figure 6. Sequence diagram with aspects

An example of including aspects in a sequence diagram is shown in Figure 6. The “aspectual” stereotype is now replaced by the “pointcut” stereotype along with the name of the pointcut that contains this specific join point. These pointcuts will weave the intended method to the advice of the associated aspect. For example, in the *Logging* aspect, there will be a pointcut named *logging()* that will run some code whenever *submitDocument(name, accessLevel)* is executed.

The creation of classes during the design modeling is a prime example of why iterative development of aspects is important. Figure 7 shows the design classes that are crosscut by the *Logging* aspect behavior. While most of the design classes are directly derived from conceptual classes, for instance *Account*, the *AccountLogin* class is an example of a class that represents a design level requirement described in the use cases but not explicitly covered in the domain model. As a result of handling newly recognized design level requirements, it is possible that some crosscutting behaviors may surface that are not readily identifiable in the earlier stages of development, one such example being caching. Since our approach uses iterative development and has a clear way to handle aspects in the previous stages of development, any new crosscutting concerns can use the techniques available in our approach to adjust the development as necessary for the new aspect.

After the classes, attributes and operations are defined, the classes along with the operation contracts contain the information necessary to create pointcuts for the aspects. To

construct the pointcut for an aspect, a developer must use the design model and the information given in the operation contracts to select the methods and classes that will be join points in the aspects. Referring back to Figure 5, the *submitDocument* method in the *NonComputingResearcher* class is a join point for the *Logging* aspect. Following a similar pattern, the remaining join points existing in the design model can be identified.

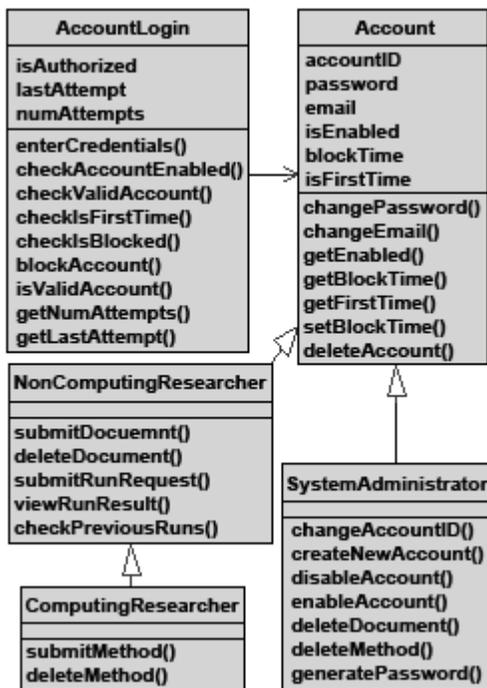


Figure 7. Partial class diagram of the ADEAP system

The value of our approach stems from providing a traceability of aspects across the development phases. From the ADEAP example, the *Logging* aspect can be traced from requirements analysis to design. Initial identification of the aspects involved breaking down the use cases into flows to highlight the logging behavior as a concern that crosscut the system. Use cases that demonstrated the behavior include the *Log In* and *Submit Run Request* examples show in Figure 1. After the log behavior is recognized as an aspect, the aspect is incorporated into the domain model through the inclusion of a conceptual aspect, and then this behavior is modeled in the system sequence diagrams. Transforming from the aspect behavior in the system sequence diagrams to the design aspect in the design model is accomplished through the operation contracts. Lastly, the sequence diagrams show the interaction of the aspects with the design level objects of the design model and along with the operation contracts, help to form the pointcuts for the aspects.

#### IV. RELATED WORK

AOSD aims at the identification, modularization and composition of crosscutting concerns in each software development phase. Existing AOSD approaches can be distinguished as two categories. The first category of the approaches were developed based on object-oriented methods through extending UML notations like the work in [19][6] with

AOSD mechanisms, especially the linguistic constructs from AspectJ [11][12][8]. Although UML based AOSD approaches provide a nice solution for general developers to design crosscutting concerns, the transformation from requirements specifications to aspect-oriented design is not smooth. The composition of aspects with their base modules during the analysis phase is challenging or even impossible in those approaches.

The second category of the approaches were designed to be new approaches like the work in [1][10][17], which create uniform aspect mechanisms for each development phase. However, those new approaches do not work well with legacy software models or experience. Jacobson and NG developed an AOSD with use cases in [10], where concerns are separated as slices of use case models, which in turn are transformed into architectural and design models. Although the approach nicely builds the traceability of aspects in each development phase via slicing use cases, slicing use cases is fairly challenging. Use case slices are not easily transformed into architectural or design modules in many cases.

In the Theme approach for AOSD discussed in [1] and [5], requirements are specified using Theme/Doc, and designs are modeled using Theme/UML. Aspects are identified and separated in the Theme/Doc model, and Theme is used to build the traceability of aspects in different development phases. Although we chose Theme/UML to model the aspect-oriented design, requirements in our approach are specified using use case models. In our approach, the composition of aspects in the domain model is unique so that the weaving of conceptual aspects is possible.

The focus of our approach is to design aspects by reusing existing successful software development methods, specifically object-oriented design and UML modeling. The AOSD approach presented in [17] is an integrated approach for defining the characteristics of aspects at each software development phase, where requirements are specified using Theme/Doc, architecture is defined using CAM and design is modeled using Theme/UML. The approach provides guidelines for refining aspects from requirements to design, and it also provides a way for managing aspect evolution through record aspect decisions. However, the approach in [17] suffers the problems of both of the categories mentioned above.

In addition to the two categories of AOSD approaches, some other AOSD approaches focus on particular development phases like requirements or design like the work in [2][4][9][16]. EA-Miner discussed in [16] is a tool-based approach to automate the identification and modularization of aspects from requirements models. The idea of EA-Miner also can be integrated to our approach. The work called COMPASS discussed in [4] focuses on how to map aspectual requirements to aspect-oriented architecture. Requirements in COMPASS are specified using an aspect-oriented requirements definition language, and the architecture is modeled using an aspect-oriented architecture description language. Although the guideline for mapping requirements to architecture is meaningful, the practical use of COMPASS is limited considering the dominant usage of use case modeling in requirements analysis and design.

## V. SUMMARY AND FUTURE WORK

We have presented an approach for design aspects with use cases through analysis and case study of an archival data extraction, assessment and preservation system. We have devised an iterative approach that establishes the guideline for early identification of aspects from use case models, modeling aspects in domain models, specifying aspect interactions through system sequence diagrams, defining the weaving of aspects to methods in the operation contracts and modularizing the aspects during design modeling. Domain modeling extended with aspects was introduced to AOSD to bridge the gap in aspect development between use case modeling and object-oriented design. In order to resolve the difficulty of composing conceptual aspects with analysis classes, an association mechanism was designed for the conceptual aspects. Compared to existing AOSD approaches, our approach preserves all advantages of AOSD, use case modeling and object-oriented methods. Our approach is more practical and easier to use not only for developing new software systems, but also for maintaining legacy software systems.

Concerning the future work, we plan to build a visualization tool to support the identification of aspects in use case models. In addition, we will apply the aspect testing approach proposed in [20] and [21] to test the ADEAP system.

## REFERENCES

- [1] E. Baniassad, and S. Clarke, "Theme: An Approach for Aspect-Oriented Analysis and Design". In Proceedings of the 26<sup>th</sup> international Conference on Software Engineering (ICSE), pp. 158-167, 2004.
- [2] E. Baniassad, P. Clements, J. Araujo, A. Moreira, A. Rashid, B. Tekinerdogan, "Discovering Early Aspects". IEEE Software 23(1), pp. 61-70, 2006.
- [3] G. Booch, R. A. Maksimchuk, M. W. Engel, and B. J. Young. Object-Oriented Analysis and Design with Applications (3<sup>rd</sup> Ed.). Addison-Wesley, 2007.
- [4] R. Chitchyan, M. Pinto, A. Rashid, and L. Fuentes, "COMPASS: Composition-Centric Mapping of Aspectual Requirements to Architecture", Transactions on Aspect-Oriented Software Development IV, LNCS, vol. 4640, pp.3-53, 2007.
- [5] S. Clarke, and E. Baniassad, "Aspect-Oriented Analysis and Design: The Theme Approach," Addison-Wesley Professional, 2005.
- [6] Z. Cui, L. Wang, X. Li, and D. Xu. "Modeling and Integrating Aspects with UML Activity Diagrams", Proc. of the 24th ACM Symposium on Applied Computing (SAC'09), 2009.
- [7] R. Filman, T. Elrad, S. Clarke, M. Aksit (eds.): "Aspect-Oriented Software Development", Addison-Wesley, 2004.
- [8] Y. Fu, J. Ding, P. Bording, "An Approach for Modeling and Analyzing Crosscutting Concerns," 5<sup>th</sup> IEEE Intl. Conf. on Services Operations, Logistics and Informatics, Chicago, pp. 91-97, 2009.
- [9] M. Harman, F. Islam, T. Xie, and S. Wappler. "Automated Test Data Generation for Aspect-Oriented Programs". In Proceedings of the 8th International Conference on Aspect-Oriented Software Development (AOSD 2009), pp. 185-196, 2009.
- [10] I. Jacobson, and P. Ng, "Aspect-Oriented Software Development with Use Cases." Addison-Wesley Professional, 2004.
- [11] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W.G. Griswold, "An Overview of AspectJ," Proc. of the European Conf. on Object-Oriented Programming (ECOOP'01), pp. 327-353, 2001.
- [12] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C.V. Lopes, J.M. Loingtier, and J. Irwin, "Aspect-Oriented Programming," Proc. of the European Conf. on Object-Oriented Programming (ECOOP'97), pp. 220-242, 1997.
- [13] C. Larman, "Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3<sup>rd</sup> Edition)". Prentice Hall, 2004.
- [14] A.Rashid, A. Moreira, "Domain Models are NOT Aspect Free". MoDELS 2006, LNCS, vol. 4199, pp. 155-169, 2006.
- [15] A. Sampaio, N. Loughran, A. Rashid, P. Rayson, "Mining Aspects in Requirements", In Workshop on Early Aspects, 2005.
- [16] A. Sampaio, A. Rashid, R. Chitchyan, and P. Rayson, "EA-Miner: Towards Automation in Aspect-Oriented Requirements Engineering", Transactions on Aspect-Oriented Software Development III, LNCS, vol 4620, pp.5-39, 2007.
- [17] P. Sánchez, L. Fuentes, A. Jackson, S. Clarke, "Aspects at the Right Time." Transactions on Aspect-Oriented Software Development IV, LNCS, vol 4640, pp.54-113, 2007.
- [18] A. Schauerhuber, W. Schwinger, W. Retschitzegger, and M. Wimmer, "A survey on aspect-oriented modeling approaches," Technical report, Vienna University of Technology, 2006.
- [19] D. Stein, S. Hanenberg, and R. Unland. "A uml-based aspect-oriented design notation for AspectJ." In *AOSD '02: Proc. of the 1st int. conf. on Aspect-oriented software development*, pp. 106-112, 2002.
- [20] D. Xu, and J. Ding, "Prioritizing State-Based Aspect Tests," 3<sup>rd</sup> Intl. Conference on Software Testing, Verification and Validation (ICST'10), Paris, April, 2010.
- [21] D. Xu and W. Xu, "State-Based Incremental Testing of Aspect-Oriented Programs," Proc. of the International Conf. on Aspect-Oriented Software Development (AOSD'06), pp.180-189, 2006.
- [22] D. Xu, and K. E. Nygard. "Threat-Driven Modeling and Verification of Secure Software Using Aspect-Oriented Petri Nets". IEEE Transactions on Software Engineering. Vol. 32, No. 4, pp. 265-278, 2006.

# Reviewers' Index

## A

Alain Abran  
Silvia Teresita Acuna  
Taiseera Albalushi  
Edward Allen

## B

Doo-Hwan Bae  
Ebrahim Bagheri  
Rami Bahsoon  
Xiaoying Bai  
Maria Teresa Baldassarre  
Purushotham Bangalore  
Fevzi Belli  
Nicolas Belloir  
Alessandro Bianchi  
James Bieman  
Jean-Michel Bruel  
Barrett Bryant

## C

Kai-Yuan Cai  
Danilo Caivano  
Gerardo Canfora  
Giovanni Cantone  
Jeffrey Carver  
Jaelson Castro  
Christine Chan  
Keith Chan  
Kuang-Nan Chang  
Ned Chapin  
Shu-Ching Chen  
Zhenyu Chen  
Yung-Pin Cheng  
Yoonsik Cheon  
Peter Clarke  
Nelly Condori-Fernandez  
Panos Constantopoulos  
Daniel Cooke  
Kendra Cooper

Maria Francesca Costabile  
Karl Cox  
Jose Luis Cuadrado  
Juan J. Cuadrado-Gallego  
Alfredo Cuzzocrea

## D

Jose Luis De La Vara  
Deepak Dhungana  
Massimiliano Di Penta  
Scott Dick  
Teresa Diez  
Maria Dominguez  
Jin Song Dong  
Jing Dong  
Dirk Draheim  
Philippe Dugerdil  
Schahram Dustdar

## E

Christof Ebert  
Raimund Ege  
Magdalini Eirinaki  
Faezeh Ensan  
Onyeka Ezenwoye

## F

Davide Falessi  
Behrouz Far  
Robert Feldt  
Eduardo B. Fernandez  
Marian Fernandez De Sevilla  
Renata Fortes

## G

Jerry Gao  
Kehan Gao  
Alessandro Garcia

Felix Garcia  
Ignacio Garcia  
Raul Garcia-Castro  
Holger Giese  
Itana Gimenes  
Swapna Gokhale  
Wolfgang Golubski  
Jeff Gray  
Des Greer  
Eric Gregoire  
Christiane Gresse Von Wangenheim

Bixin Li  
Ming Li  
Tao Li  
Chien-Hung Liu  
Shih-Hsi Liu  
Xiaodong Liu  
Yan Liu  
Yi Liu  
Hakim Lounis  
Joan Lu  
Heiko Ludwig

## **H**

Xudong He  
Miguel Herranz  
Mong Fong Horng  
Shihong Huang

## **I**

Hoh Peter In

## **J**

Clinton Jeffery  
Jason Jung  
Natalia Juristo

## **K**

Taghi Khoshgoftaar  
Sascha Konrad  
Gunes Koru  
Nicholas Kraft  
Dhananjay Kulkarni  
Vinay Kulkarni  
Gihwon Kwon

## **L**

Mark Last  
Konstantin Laufer  
Jeff Lei

## **M**

Jose Carlos Maldonado  
Antonio Mana  
Miriam Martinez  
Hong Mei  
Hsing Mei  
Emilia Mendes  
Ali Mili  
Ana M. Moreno  
Henry Muccini  
Enriqueta Muel

## **N**

Martin Neil  
Kia Ng  
Ngoc Thanh Nguyen  
Allen Nikora

## **O**

Mehmet Orgun

## **P**

Antonio Piccinno  
Alfonso Pierantonio

## **R**

Rick Rabiser  
Damith C. Rajapakse

Rajeev Raje  
Jose Angel Ramos  
Marek Reformat  
Robert Reynolds  
Daniel Rodriguez  
Guenther Ruhe

Hironori Washizaki  
Victor Winter  
Guido Wirtz  
Eric Wong  
Franz Wotawa  
Hui Wu

### **S**

Samira Sadaoui  
Masoud Sadjadi  
Ramon Sagarna  
Salvatore Alessandro Sarcia  
Kamran Sartipi  
Douglas Schmidt  
Andreas Schoenberger  
Naeem (jim) Seliya  
Tony Shan  
Rajan Shankaran  
Yi-Dong Shen  
Michael Shin  
George Spanoudakis  
Xiao Su  
Rajesh Subramanyan

### **T**

Jeff Tian  
Genny Tortora  
Mark Trakhtenbrot  
Peter Troger  
T.h. Tse

### **V**

Michael Vanhilst  
Sira Vegas  
Silvia Vergilio  
Arndt Von Staa

### **W**

Huanjing Wang  
Qianxiang Wang

### **X**

Haiping Xu

### **Y**

Chi-Lu Yang  
Hongji Yang  
Huiqun Yu

### **Z**

Cui Zhang  
Du Zhang  
Jing Zhang  
Min-Ling Zhang  
Zhenyu Zhang  
Zhinan Zhou  
Hong Zhu  
Xingquan Zhu  
Eugenio Zimeo  
Andrea Zisman

# Authors' Index

## A

Alain Abran, 153  
Mohammad Abu-Matar, 468  
Ahmed Raafat Abuzeid, 67  
Tessa Adderley, 599  
Hamidreza Afzali, 748  
Salifu Alhassan, 93  
Khalid T. Al-Sarayreh, 153  
Nelio Alves, 710  
César Andrés, 405  
Patrice Arruda, 559  
Thomas Artner, 198  
Fernando Asteasuain, 430  
Carmen Avila, 393  
George S. Avrunin, 765

## B

Ali Sajedi Badashian, 748  
Xiaoying Bai, 462  
Karim Baïna, 664  
Salah Baïna, 664  
Simon Baker, 319  
Pierre Baldi, 141  
Ajay Bansal, 247  
Ahmad Baraani-Dastjerdi, 79  
Souvik Barat, 577  
Simone D.J. Barbosa, 509  
Flavia de A. Barros, 259  
Márcio O. Barros, 387  
Vitor A. Bastista, 716  
Feten Baccar Ben Amar, 515  
Henda Hajjami Ben Ghezala, 675  
Abdelmajid Ben Hamadou, 515  
Sihem Ben Sassi, 675  
Ayşe Bener, 93  
Mario Bernhart, 198, 269  
Swapan Bhattacharya, 34, 448  
Stefan Biffel, 478, 620  
Eyal Bin, 303

M. Brian Blake, 247  
Vadym Borovski, 298  
Thouraya Bouabana-Tebibel, 771  
Victor Braberman, 430  
Regina Braga, 84  
K. S. Braunsdorf, 331  
Giacomo Bucci, 54  
M. Brian Blake, 247  
Vadym Borovski, 298  
Thouraya Bouabana-Tebibel, 771  
Victor Braberman, 430  
Regina Braga, 84  
K. S. Braunsdorf, 331  
Giacomo Bucci, 54

## C

Bora Çağlayan, 93  
John Campbell, 309  
Fernanda Campos, 84  
Zining Cao, 399  
T. Carvalho, 292  
William Carvalho, 710  
Jaelson Castro, 48  
Yguaratã Cerqueira Cavalcanti, 742  
Mauricio Jacó Cerri, 529  
Nabendu Chaki, 34  
Pierre Chamoun, 559  
Christine W. Chan, 604  
Debasis Chanda, 448  
Lily Chang, 657  
Shi-Kuo Chang, 521  
Soo Ho Chang, 458  
Ana Paula Chaves, 437  
Bin Chen, 765  
Lin Chen, 273  
Yinong Chen, 462  
Zhenyu Chen, 253, 273  
Xiyi Cheng, 315  
Yoonsik Cheon, 393

Sankhayan Choudhury, 34  
Randy Chow, 325  
Tiago Cinto, 361  
Lori A. Clarke, 765  
Francesco Colace, 521  
Emmanuel Coquery, 235  
Hélio R. Costa, 387  
Juan J. Cuadrado-Gallego, 135, 153

## D

Laryssa Machado da Silva, 84  
Lidia Martins da Silva, 608  
Carlos Eduardo Albuquerque da Cunha, 742  
Ana Estela Antunes da Silva, 608  
Deepak Dahiya, 176  
Gargi Dasgupta, 587  
Cícero Roberto Ferreira de Almeida, 499  
Eduardo Santana de Almeida, 742  
Antonio Carlos De Arruda Junior, 499  
Jose Luis de la Vara, 425  
Carlos J.P. de Lucena, 509  
Alba Cristina Magalhaes Alves de Melo, 499  
J.A. Gutiérrez de Mesa, 135  
Lucas Bueno R. de Oliveira, 738  
Massimo De Santo, 521  
Luciano S. de Souza, 259  
Renata M.C.R. de Souza, 265  
Itana Maria de Souza Gimenes, 221  
Maria Beatriz Felgar de Toledo, 221  
Vidroha Debroy, 123  
Morteza Ashurzad Delcheh, 748  
Dwight Deugo, 559  
Junhua Ding, 797  
Fei Dong, 286  
Weichang Du, 192  
Yongwei Duan, 253  
Sheryl Duggins, 599  
Gregor Dürr, 478

## E

Armin Eberlein, 42, 686

Csaba Egyhazy, 672  
Emad Elabd, 235  
Abdel-Halim Hafez Elamy, 495  
Mohamed El-Attar, 571  
Ahmed Elkhodary, 468  
Jay Elston, 462  
Miguel Esteban-Gutiérrez, 129  
Hugo Estrada, 225  
Onyeka Ezenwoye, 587

## F

Roberta A. A. Fagundes, 265  
Marcelo Fantinato, 221  
Behrouz H. Far, 349, 495, 686  
Daniel Feitosa, 738  
Robert Feldt, 374  
Katia R. Felizardo, 738  
Antonio Ferrández, 335  
Régis Fleurquin, 339  
Liana Fong, 587  
Renata Pontin M. Fortes, 355  
Xiang Fu, 535

## G

Matthias Galster, 42  
Kehan Gao, 203, 215  
Raúl García-Castro, 129  
Bilel Gargouri, 515  
Vahid Garousi, 186, 489  
Dragan Gašević, 73  
Reinhard German, 61  
Alaa Ghanayim, 303  
Ankit Goel, 279  
Swapna S. Gokhale, 99  
Wolfgang Golubski, 787  
Hassan Gomaa, 468  
Asunción Gómez-Pérez, 614  
Hemanth Thimme Gowda, 653  
Jim Graham, 640  
Thomas Grechenig, 198, 269  
Thomas Grechenig, 198, 269

Andreas Gregoriades, 24  
Stephan Grimm, 129  
Gilleanes Thorwald Araujo Guedes, 28  
Rym Nesrine Guibadj, 771  
Sonia Gulrajani, 117  
Donghui Guo, 123  
Jin-Gang Guo, 722  
Simon Suigen Guo, 604  
Gopal Gupta, 247  
Priya Gupta, 474

## H

Mohand-Said Hacid, 235  
Haitham S. Hamza, 67  
Taeman Han, 411, 645  
Robert Harrison, 604  
Tobias Haubold, 787  
Xiao-yang He, 722  
Xudong He, 657  
Kai-Steffen Hielscher, 61  
Nishigandha Hirve, 583  
Karen Holtz, 303  
Hao Hu, 452  
Wei-Chung Hu, 345  
Gang Huang, 591  
Ming Huang, 106, 543  
Pei Shu Huang, 417  
Lindsey Hughes, 141  
Elisa H. M. Huzita, 437

## I

Hamdy Ibrahim, 686  
Tacksoo Im, 165

## J

Choulsoo Jang, 653  
Zakwan Jaroucheh, 241  
Mehdi Jazayeri, 73  
He Jiang, 209  
Jia-Rui Jiang, 722

Ning Jiang, 591  
Hewijin Christine Jiau, 345  
Seungwook Jung, 653

## K

Selim Kalayci, 587  
Taeghyun Kang, 653  
Chia Hung Kao, 345  
Jessie Kennedy, 692  
Mick Kerriganz, 129  
Iman Khalkhali, 748  
Taghi M. Khoshgoftaar 203, 215  
Ehsan KhoumSiavash, 79  
Amir A. Khwaja, 649  
Minseong Kim, 468  
Soo Dong Kim, 458  
Sunghoon Kim, 653  
Wolfgang Koch, 298  
Srividya Kona, 247  
Jun Kong, 726  
Aneesh Krishna, 24  
Vinay Kulkarni, 577  
Gihwon Kwon, 411, 645

## L

Edgard Lamounier, 710  
Vincent Le Gloahec, 339  
Oliver Le Goaer, 159  
Sarah B. Lee, 331  
Alessandro Ferreira Leite, 499  
Chengpu Li, 692  
Chung-Chih Li, 535  
Ge Li, 367  
Juan Li, 726  
Weigang Li, 499  
Xiao Li, 325  
Hyoungju Lim, 645  
Erik Linstead, 141  
Xiaodong Liu, 241, 692  
Emadoddin Livani, 634  
William M Lively, 88

Richard Long, 495  
Cristina Lopes, 141  
Alejandra Yopez Lopez, 38  
Jijun Lu, 99  
Yisha Lu, 543  
Zhongxuan Luo, 209

## M

Dawn MacIsaac, 192  
Mostafa Madiesh, 698  
Ana Magazinius, 374  
Mehregan Mahdavi, 748  
Dwijesh Dutta Majumder, 448  
José Carlos Maldonado, 628  
Ricardo M. Marcacini, 553  
Eitan Marcus, 303  
Alicia Martinez, 225  
Borja Martín-Herrera, 135  
Andreas Mauczka, 198, 269  
Frank Maurer, 186  
Jose-Norberto Mazón, 335  
Craig McDonald, 309  
John D. McGregor, 165  
Sara Mehar, 771  
Hong Mei, 591  
Silvio Romero L. Meira, 742  
Emilia Mendes, 319  
Mercedes G. Merayo, 405  
Ronny Morad, 303  
Itzel Morales-Ramirez, 225  
Thomas Moser, 478, 620  
Mohammad Moshirpour, 349  
Abdolmajid Mousavi, 349  
Mahmood Moussavi, 42  
Oscar Muñoz, 614  
Tukaram Muske, 583

## N

Elisa Yumi Nakagawa, 628, 738  
Valeh H. Nasser, 192  
Nikolas Nehmer, 757

Saša Nešić, 73  
Nan Niu, 38  
Zhendong Niu, 315  
Ingrid Nunes, 509  
Manuel Núñez, 405

## O

Anuja Oka, 117  
Mourad Oussalah, 159

## P

Clarindo Isaías P. S. Pádua, 716  
Elham Paikari, 380  
Jingui Pan, 765  
F. Papatella, 292  
Sachoun Park, 411, 645  
David Lorge Parnas, 791  
Oscar Pastor, 225  
Nipul Patel, 565  
Sachin Patel, 474  
Cecilia Sosa Arias Peixoto, 361  
Daniela C. C. Peixoto, 716  
Juan C. Pelaez, 777  
Ofar Peled, 303  
E. Pereira, 292  
Andre Pflueger, 787  
Mario Piattini, 704  
João Pimentel, 48  
Caryna Pinheiro, 186  
Francisco J. Pino, 704  
Ricardo B. C. Prudêncio, 259

## Q

Ju Qian, 253

## R

Łukasz Radliński, 113  
Katrina Reffett, 640  
Zhilei Ren, 209

Rodolfo F. Resende, 716  
Solange O. Rezende, 355, 553  
Marcela Xavier Ribeiro, 529  
Michael M. Richter, 380  
Michal Rimon, 303  
Ana Regina Rocha, 387  
Elder de M. Rodrigues, 483  
Pablo Rodríguez-Soria, 135  
Rafael Rossi, 355  
Arjun G. Roy, 726  
Guenther Ruhe, 380, 634

## S

S. Masoud Sadjadi, 587  
Salah Sadou, 339  
Soraya Sakhraoui, 339  
Juan Sánchez, 425  
Valeriano Sandrucci, 54  
Shadan Saniepour Esfahani, 443  
Emanuel Santos, 48  
Marilde T. P. Santos, 529  
Amritam Sarcar, 393  
Anirban Sarkar, 34  
Ichiro Satoh, 503  
Gregor Scheithauer, 452  
Eduardo Segura, 783  
Snehadeep Sethia, 565  
Mohammad Shoja Shafiei, 748  
Sol M. Shatz, 279, 286  
Yuri Shewchuk, 489  
Chongyang Shi, 315  
Michael E. Shin, 565, 653  
Sajjan G. Shiva, 331  
Ulka Shrotri, 583  
Gil Shurek, 303  
Talal Siddiqui, 443  
Sebastian Siegl, 61  
Jonathan Sillito, 186  
Gabriel Costa Silva, 221  
Paulo Anselmo M. Silveira Neto, 742  
Dick B. Simmons, 88  
Soumaya Slimani, 664

Sally Smith, 241  
Chattrakul Sombattheera, 24  
Byoungyoul Song, 653  
M. Song, 292  
Rodrigo Oliveira Spínola, 732  
Narayanan Srinivasaraghavan, 309  
Kuo-Feng Ssu, 345  
Igor Steinmacher, 437  
Xiao Su, 117, 783  
Mari Carmen Suárez-Figueroa, 614  
Vinitha H. Subburaj, 668  
Wikan Danar Sunindyo, 620  
Prafullakumar Surve, 474

## T

M. N. H. Tabrizi, 797  
Ismail Abdel Hamid Taha, 67  
Americo Talarico Neto, 355  
Dalila Tamzalit, 159  
Christophe Thovex, 548  
Chouki Tibermacine, 339  
Mark Trakhtenbrot, 147  
Guilherme Horta Travassos, 732  
João Vítor Tornisiello Trevisan, 628  
Francky Trichet, 548  
Miguel Morales Trujillo, 704  
Wei-Tek Tsai, 462  
Elena Tsanko, 303  
Frank Tsui, 599

## U

Joseph E. Urban, 649, 668  
Christelle Urtado, 680

## V

Sylvain Vauttier, 680  
R. Venkatesh, 583  
Pawan Kumar Verma, 176  
Rosa Maria Vicari, 28  
Enrico Vicario, 54

Leonardo D. Viccari, 483  
Vaninha Vieira, 437  
Martín Vigo, 614  
Katia Vila, 335

## W

Myles Wallace, 640  
Ching Huey Wang, 417  
Danhua Wang, 765  
Feng Jian Wang, 417  
Huanjing Wang, 215  
Leye Wang, 367  
Lijie Wang, 367  
Lixin Wang, 180  
Qing Wang, 6  
Shaochun Wang, 421  
Yasha Wang, 722, 752  
Yong Wang, 88  
Ziyuan Wang, 273  
Tom Wanyama, 686  
Christopher R. Westbrook, 797  
Rose Williams, 640  
Guido Wirtz, 452, 698  
Denis Wolf, 738  
W. Eric Wong, 123  
Andrew Wyeth, 172

## X

Bing Xie, 367, 752  
Baowen Xu, 273  
Dianxiang Xu, 88  
Haiping Xu, 279, 286  
Xiaofeng Xu, 123  
Yongjun Xu, 752

Jifeng Xuan, 209

## Y

Cristiane A. Yaguinuma, 529  
Jun Yan, 209  
Nacim Yanes, 675  
Feng Pu Yang, 345  
Ye Yang, 6  
Zhongjun Yang, 253  
Peter Yastrebenetsky, 147  
Cesar Yeep, 393  
Yu-Fang Yeh, 18  
Ying Yin, 231

## Z

L. Zarate, 292  
Emilio Zegarra, 521  
Alexander Zeier, 298  
Qingkai Zeng, 106, 543  
Bin Zhang, 231  
Cui Zhang, 172  
Du Zhang, 12  
Huaxi (Yulin) Zhang, 680  
Nuyun Zhang, 591  
Weiyi Zhang, 726  
Wen Zhang, 6  
Xizhe Zhang, 231  
Ying Zhang, 591  
Zhi Zhang, 106  
Junfeng Zhao, 752  
Zhihong Zhao, 253  
Peide Zhong, 462  
Wenhui Zhu, 791  
Avelino F. Zorzo, 483

# SEKE 2011 Call For Papers

## The Twenty-Third International Conference on Software Engineering and Knowledge Engineering

[www.ksi.edu/seke/seke11.html](http://www.ksi.edu/seke/seke11.html)

Eden Roc Renaissance Miami Beach Hotel, USA

July 7 - July 9, 2011

Organized by

Knowledge Systems Institute Graduate School

The Twenty-Third International Conference on Software Engineering and Knowledge Engineering (SEKE 2011) will be held at Eden Roc Renaissance Miami Beach Hotel July 7-9, 2011 ([www.marriott.com/miasr](http://www.marriott.com/miasr))

The conference aims at bringing together experts in software engineering and knowledge engineering to discuss on relevant results in either software engineering or knowledge engineering or both. Special emphasis will be put on the transference of methods between both domains.

### TOPICS

Agent architectures, ontologies, languages and protocols  
Multi-agent systems  
Agent-based learning and knowledge discovery  
Interface agents  
Agent-based auctions and marketplaces  
Artificial life and societies  
Secure mobile and multi-agent systems  
Mobile agents  
Mobile Commerce Technology and Application Systems  
Mobile Systems

Autonomic computing  
Adaptive Systems  
Integrity, Security, and Fault Tolerance  
Reliability  
Enterprise Software, Middleware, and Tools  
Process and Workflow Management  
E-Commerce Solutions and Applications  
Industry System Experience and Report

Service-centric software engineering  
Service oriented requirements engineering  
Service oriented architectures  
Middleware for service based systems  
Service discovery and composition  
Quality of services  
Service level agreements (drafting, negotiation, monitoring and management)  
Runtime service management  
Semantic web

Requirements Engineering  
Agent-based software engineering  
Artificial Intelligence Approaches to Software Engineering  
Component-Based Software Engineering  
Automated Software Specification  
Automated Software Design and Synthesis  
Computer-Supported Cooperative Work  
Embedded and Ubiquitous Software Engineering  
Measurement and Empirical Software Engineering  
Reverse Engineering  
Programming Languages and Software Engineering  
Patterns and Frameworks  
Reflection and Metadata Approaches  
Program Understanding

Knowledge Acquisition  
Knowledge-Based and Expert Systems  
Knowledge Representation and Retrieval  
Knowledge Engineering Tools and Techniques  
Time and Knowledge Management Tools  
Knowledge Visualization  
Data visualization  
Uncertainty Knowledge Management  
Ontologies and Methodologies  
Learning Software Organization

Tutoring, Documentation Systems  
Human-Computer Interaction  
Multimedia Applications, Frameworks, and Systems  
Multimedia and Hypermedia Software Engineering

Smart Spaces  
Pervasive Computing  
Swarm intelligence  
Soft Computing

Software Architecture  
Software Assurance  
Software Domain Modeling and Meta-Modeling  
Software dependability  
Software economics  
Software Engineering Decision Support  
Software Engineering Tools and Environments  
Software Maintenance and Evolution  
Software Process Modeling  
Software product lines  
Software Quality  
Software Reuse  
Software Safety  
Software Security  
Software Engineering Case Study and Experience Reports

Web and text mining  
Web-Based Tools, Applications and Environment  
Web-Based Knowledge Management  
Web-Based Tools, Systems, and Environments  
Web and Data Mining

### CONFERENCE SITE (HOTEL INFORMATION)

The SEKE 2011 Conference will be held at Eden Roc Renaissance Miami Beach Hotel. 4525 Collins Ave. Miami Beach, Florida 33140 USA. The hotel has made available for these limited dates (7/3 - 7/11/2011) to SEKE 2011 attendees a discount rate of USD\$112 per room, per night, single/double occupancy, not including sales tax. [www.marriott.com/miasr](http://www.marriott.com/miasr)

### INFORMATION FOR AUTHORS

Papers must be written in English. An electronic version (Postscript, PDF, or MS Word format) of the full paper should be submitted using the following URL: <http://conf.ksi.edu/seke11/submit/SubmitPaper.php>  
Please use Internet Explorer as the browser. Manuscript must include a 200-word abstract and no more than 6 pages of IEEE double column text (include figures and references).

### INFORMATION FOR REVIEWERS

Papers submitted to SEKE'11 will be reviewed electronically. The users (webmaster, program chair, reviewers...) can login using the following URL: <http://conf.ksi.edu/seke11/review/pass.php>  
If you have any questions or run into problems, please send e-mail to: E-mail: [seke11@ksi.edu](mailto:seke11@ksi.edu)

SEKE 2011 Conference Secretariat  
Knowledge Systems Institute Graduate School  
3420 Main Street  
Skokie, IL 60076 USA  
Tel: 847-679-3135  
Fax: 847-679-3166  
E-mail: [seke11@ksi.edu](mailto:seke11@ksi.edu)

### IMPORTANT DATES

March 07, 2011 *Paper submission due*  
April 20, 2011 *Notification of acceptance*  
May 10, 2011 *Camera-Ready Copy*



**Proceedings of the  
Twenty-Second  
International Conference on  
Software Engineering &  
Knowledge Engineering**

**SEKE**  
2010

Copyright © 2010  
Printed by  
Knowledge Systems Institute  
3420 Main Street  
Skokie, Illinois 60076  
(847) 679-3135  
info@ksi.edu  
www.ksi.edu  
Printed in USA, 2010  
ISBN 1-891706-26-8 (paper)

