

# SEKE 2018

The background of the cover is a photograph of the Golden Gate Bridge in San Francisco. The bridge is a large suspension bridge with two prominent towers and a long span across the water. The water is a deep blue-green color, and the sky is a clear, bright blue. In the foreground, there is a hillside with dry, golden-brown grass. The overall scene is bright and clear, suggesting a sunny day.

**Proceedings of the 30th  
International Conference on  
Software Engineering &  
Knowledge Engineering**

**San Francisco Bay  
July 1 - July 3, 2018**

**PROCEEDINGS**  
**SEKE 2018**

**The 30<sup>th</sup> International Conference on  
Software Engineering &  
Knowledge Engineering**

**Sponsored by**

KSI Research Inc. and Knowledge Systems Institute Graduate School, USA

**Technical Program**

**July 1 – 3, 2018**

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

**Organized by**

KSI Research Inc. and Knowledge Systems Institute Graduate School, USA

Copyright © 2018 by KSI Research Inc. and 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: 1-891706-44-6

ISSN: 2325-9000 (print)

2325-9086 (online)

DOI reference number: 10.18293/SEKE2018

Publisher Information:

KSI Research Inc. and Knowledge Systems Institute Graduate School

156 Park Square

Pittsburgh, PA 15238 USA

Tel: +1-412-606-5022

Fax: +1-847-679-3166

Email: [seke@ksiresearch.org](mailto:seke@ksiresearch.org)

Web: <http://ksiresearchorg.ipage.com/seke/seke18.html>

Proceedings preparation, editing and printing are sponsored by KSI Research Inc. and Knowledge Systems Institute Graduate School, USA.

Printed by KSI Research Inc. and Knowledge Systems Institute Graduate School

# FOREWORD

Welcome to the 30th International Conference on Software Engineering and Knowledge Engineering (SEKE), in Hotel Pullman, Redwood City, USA. In last 30 years, SEKE has established itself as a major international forum to foster, among academia, industry, and government agencies, discussion and exchange of ideas, research results and experience in software engineering and knowledge engineering. The SEKE community has grown to become a very important and influential source of ideas and innovations on the interplays between software engineering and knowledge engineering, and its impact on the knowledge economy has been felt worldwide. On behalf of the Program Committee, it is my great pleasure to invite you to participate, not only in the technical program of SEKE 2018 and its rich assortment of activities, but also in enjoying the beautiful San Francisco Bay Area.

This year, we received 214 submissions from 32 countries. Through a rigorous review process where a majority (90 percent) of the submitted papers received three reviews, and the rest with two reviews, we were able to select 75 full papers for the general conference (35 percent), 72 short papers (34 percent), 35 posters (16 percent) and 32 (15 percent) reject. Out of that, 15 papers have been accepted for the 3 special sessions (SESE 4 papers, DISA 7 papers and LATTICE 4 papers), and 132 papers are scheduled for presentation in thirty five sessions during the conference. In addition, the technical program includes three excellent keynote speeches, as well as the special sessions on Semantic Enabled Software Engineering (SESE), Data Intensive Service-based Applications (DISA) and Conceptual Lattices for Software Systems Engineering (LATTICE).

The high quality of the SEKE 2018 technical program would not have been possible without the tireless effort and hard work of many individuals. First of all, we would like to express our sincere appreciation to all the authors whose technical contributions have made the final technical program possible. We are very grateful to all the Program Committee members whose expertise and dedication made our responsibility that much easier. Our gratitude also goes to the keynote speakers who graciously agreed to share their insight on important research issues, to the conference organizing committee members for their superb work, and to the external reviewers for their contribution.

Personally, we owe a debt of gratitude to a number of people whose help and support with the technical program and the conference organization are unfailing and indispensable. We are deeply indebted to Dr. S. K. Chang, Chair of the Steering Committee, for his constant guidance and support that are essential to pull off SEKE 2018. Our heartfelt appreciation goes to Dr. Xudong He, Florida International University, USA, the Conference Chair, for his help and experience, and to the Program Committee Chair, Dr. Oscar Mortagua Pereira, University of Aveiro, Portugal, and Program Committee Co-Chair, Dr. Angelo Perkusich, Federal University of Campina Grande, Brazil, for their outstanding team work. In addition, we also like to express our appreciation to Prof. Jing Sun, The University of Auckland, New Zealand, to Prof. Honghao Cao, Shanghai University, China, and to Prof. Jaakov Exman, The Jerusalem College of Engineering, Israel, for their excellent job in organizing the special sessions SESE, DISA and LATTICE, respectively.

We would like also to express our great appreciation to all of the conference organization committee members, including the Publicity Chair, Dr. Robert Heinrich, Karlsruhe Institute of Technology, Germany, Poster Session Chair, Dr. Shi-Kuo Chang, University of Pittsburgh, USA. Moreover, we would like to appreciate and recognize our Conference Liaisons in different regions for their important contributions. They are: Asia Liaison – Hironori Washizaki, Waseda University, Japan; Australasia Liaison – Jing Sun, The University of Auckland, New Zealand; Europe Liaison - Raul Garcia Castro, Universidad Politecnica de Madrid, Spain; India Liaison - Swapan Bhattacharya, National Institute of Technology Karnataka, Surathakl; and South America Liaison - Jose Carlos Maldonado, ICMC-USP, Brazil.

We would like to express our great appreciation for the availability expressed by Dr. Bingyang Wei, Midwestern State University, USA, Dr. Yong Wang, New Mexico Highlands University, USA, Dr. Angelo Perkusich, Federal University of Campina Grande, Brazil, Dr. Xudong He, Florida International University, USA, and Dr. Oscar Mortagua Pereira, University of Aveiro, Portugal, to dedicate an outstanding effort on the reviewing process of papers.

Last but certainly not the least, we must acknowledge the important contributions that the KSI staff members have made. Their timely and dependable support and assistance throughout the entire process have been truly remarkable. Finally, we wish you have productive discussion, great networking, effective presentation, and pleasant stay and travel in San Francisco to participate in SEKE 2018.

Finally, to conclude, we are happy to announce that SEKE 2019 will take place in Lisbon, the stunning capital of Portugal and one of the capitals of Western Europe with greater notoriety. Lisbon is a city with more than 800 years where historical heritage, modernism, culture and nightlife combine in a perfect harmony. The excellence of the climate and the sympathy of the Portuguese population associated with the diversity and quality of the opportunities offered leave visitors with a desire to return to Lisbon.

Lisbon and Portugal have recently won several international awards as destinations of excellence, such as those from the World Travel Award.

Oscar Mortagua Pereira, University of Aveiro, Portugal, Program Committee Chair

Angelo Perkusich, Federal University of Campina Grande, Brazil, Program Committee Co-Chair

# **SEKE 2018**

## **The 30<sup>th</sup> International Conference on Software Engineering & Knowledge Engineering**

**July 1– 3, 2018**

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

### **Conference Organization**

#### **CONFERENCE CHAIR**

Xudong He, Florida International University, USA

#### **PROGRAM COMMITTEE CHAIR**

Oscar Mortagua Pereira, University of Aveiro, Portugal

#### **PROGRAM COMMITTEE CO-CHAIR**

Angelo Perkusich, Federal University of Campina Grande, Brazil

#### **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

#### **ADVISORY COMMITTEE**

Jerry Gao, San Jose State University, USA

Swapna Gokhale, University of Connecticut, USA

Natalia Juristo, Universidad Politecnica de Madrid, Spain

Taghi Khoshgoftaar, Florida Atlantic University, USA

Guenther Ruhe, University of Calgary, Canada

Masoud Sadjadi, Florida International University, USA

Du Zhang, California State University, USA

## PROGRAM COMMITTEE

Silvia Teresita Acuna, Universidad Autonoma de Madrid, Spain  
Shadi Alawneh, Oakland University, USA  
Taisira Al-Belushi, Sultan Qaboos University, Oman  
Mark Allison, University of Michigan - Flint, USA  
John Anvik, Univ. of Lethbridge, Canada  
Omar El Ariss, Penn State Univ at Harrisburg, USA  
Doo-Hwan Bae, Korea Advanced Institute of Science and Technology, Korea  
Hamid Bagheri, George Mason University and Massachusetts Institute of Technology, USA  
Xiaoying Bai, Tsinghua University, China  
Fevzi Belli, University of Paderborn, Germany  
Ateet Bhalla, Consultant, India  
Swapna Bhattacharya, NITK, Surathakl, India  
Alessandro Bianchi, University of Bari, Italy  
Raul Garcia Castro, Universidad Politecnica de Madrid, Spain  
Chih-Hung Chang, Hsiuping University of Science and Technology, Taiwan  
Keith Chan, Hong Kong Polytechnic University, Hong Kong  
Lily Chang, University of Wisconsin, Platteville, USA  
Shu-Ching Chen, Florida International University, USA  
Wen-Hui Chen, National Taipei University of Technology, Taiwan  
William Chu, Tunghai University, Taiwan  
Stelvio Cimato, University of Milan, Italy  
Fabio M. Costa, Universidade Federal de Goias, Brazil  
Jose Luis De La Vara, Carlos III University of Madrid, Spain  
Scott Dick, University of Alberta, Canada  
Junhua Ding, East Carolina University, USA  
Zhijian Dong, Middle Tennessee State University, USA  
Weichang Du, University of New Brunswick, Canada  
Philippe Dugerdil, HEG - Univ. of Applied Sciences, Switzerland  
Christof Ebert, Vector Consulting Services, Germany  
Ali Ebneenasir, Michigan Technological University, USA  
Magdalini Eirinaki, San Jose State University, USA  
Ruby ElKharboutly, Quinnipiac University, Canada  
Behrouz Far, University of Calgary, Canada  
Liana Fong, IBM, USA  
Ellen Francine Barbosa, University of Sao Paulo, Brazil  
Jerry Gao, San Jose State University, USA  
Kehan Gao, Eastern Connecticut State University, USA  
Olivier Le Goer, University of Pau, France  
Swapna Gokhale, Univ. of Connecticut, USA  
Wolfgang Golubski, Zwickau University of Applied Sciences, Germany  
Anurag Goswami, North Dakota State Univ., USA  
Desmond Greer, Queen's University Belfast, United Kingdom  
Christiane Gresse Von Wangenheim, UFSC - Federal University of Santa Catarina, Brazil  
Katarina Grolinger, University of Western Ontario, Canada  
Hao Han, National Institute of Informatics, Japan  
Xudong He, Florida International University, USA  
Robert Heinrich, Karlsruhe Institute of Technology, Germany  
Rubing Huang, Jiangsu University, China  
Shihong Huang, Florida Atlantic University, USA  
Basse Isong, North-West University, South Africa  
Clinton Jeffery, University of Idaho, USA  
Jason Jung, Chung-Ang University, South Korea  
Pankaj Kamthan, Concordia University, Canada  
Ananya Kanjilal, B.P. Poddar Institute of Technology and Management, India

Taghi Khoshgoftaar, Florida Atlantic University, USA  
 Aneesh Krishna, Curtin University of Technology, Australia  
 Vinay Kulkarni, Tata Consultancy Services, India  
 Meira Levy, Shenkar College of Engineering and Design, Israel  
 Bixin Li, Southeast University, China  
 Yuan-Fang Li, Monash University, Australia  
 Zhi Li, Guangxi Normal University, China  
 Jianhua Lin, Eastern Connecticut State University, USA  
 Lan Lin, Ball State University, USA  
 Shih-hsi Liu, California State University, Fresno, USA  
 Ting Liu, Xian Jiaotong University, China  
 Xiaodong Liu, Edinburgh Napier University, United Kingdom  
 Luanna Lopes Lobato, Federal University of Goias, Brazil  
 Jian Lu, Nanjing University, China  
 Baojun Ma, Beijing University of Posts and Telecommunications, China  
 Ivan Machado, Federal University of Bahia, Brazil  
 Marcelo de Almeida Maia, Federal University of Uberlândia, Brazil  
 Beatriz Marin, Universidad Diego Portales, Chile  
 Riccardo Martoglia, University of Modena and Reggio Emilia, Italy  
 Santiago Matalonga, University of the West of Scotland, UK  
 Andre Menolli, Universidade Estadual do Norte do Parana (UENP), Brazil  
 Ali Mili, NJIT, USA  
 Alok Mishra, Atilim University, Turkey  
 Hiroyuki Nakagawa, Osaka University, Japan  
 Amjad Nusayr, University of Houston-Victoria, USA  
 Edson A. Oliveira Jr., State University of Maringa, Brazil  
 Oscar Mortagua Pereira, University of Aveiro, Portugal  
 Antonio Piccinno, University of Bari, Italy  
 Alfonso Pierantonio, University of L'Aquila, Italy  
 Rick Rabiser, Johannes Kepler University, Austria  
 Claudia Raibulet, University of Milan, Italy  
 Damith C. Rajapakse, National University of Singapore, Singapore  
 Rajeev Raje, IUPUI, USA  
 Henrique Rebelo, Universidade Federal de Pernambuco, Brazil  
 Marek Reformat, University of Alberta, Canada  
 Hassan Reza, University of North Dakota, USA  
 Ignacio Garcia Rodriguez De Guzman, University of Castilla-La Mancha, Spain  
 Daniel Rodriguez, Universidad de Alcala, Spain  
 Ivan Rodero, The State University of New Jersey, USA  
 Samira Sadaoui, University of Regina, Canada  
 Masoud Sadjadi, Florida International University, USA  
 Claudio Sant'Anna, Universidade Federal da Bahia, Brazil  
 Abdelhak-Djamel Seriai, University of Montpellier 2 for Sciences and Technology, France  
 Michael Shin, Texas Tech University, USA  
 Martin Solari, Universidad ORT Uruguay, Uruguay  
 George Spanoudakis, City University London, United Kingdom  
 Jing Sun, University of Auckland, New Zealand  
 Meng Sun, Peking University, China  
 Yanchun Sun, Peking University, China  
 Gerson Sunye, University of Nantes, France  
 Chuanqi Tao, Nanjing University of Science and Technology, China  
 Mark Trakhtenbrot, Holon Institute of Technology, Israel  
 Burak Turhan, Oulu University, Finland  
 Christelle Urtado, LGI2P Ecole des Mines d'Ales, France  
 Sylvain Vauttier, Ecole des mines d'Ales, France  
 Gennaro Vessio, University of Bari, Italy  
 Sergiy Vilkomir, East Carolina University, USA



Aaron Visaggio, University of Sannio, Italy  
 Arndt Von Staa, Pontifical Catholic University of Rio de Janeiro, Brazil  
 Huanjing Wang, Western Kentucky University, USA  
 Xiaoyin Wang, University of Texas at San Antonio, USA  
 Ye Wang, Zhejiang Gongshang University, China  
 Yong Wang, New Mexico Highlands University, USA  
 Zhongjie Wang, Harbin Institute of Technology, China  
 Ziyuan Wang, Nanjing University of Posts and Telecommunications, China  
 Hironori Washizaki, Waseda University, Japan  
 Bingyang Wei, Midwestern State University, USA  
 Guido Wirtz, Bamberg University, Germany  
 Franz Wotawa, TU Graz, Austria  
 Peng Wu, Institute of Software, Chinese Academy of Sciences, China  
 Dianxiang Xu, Boise State University, USA  
 Frank Weifeng Xu, Bowie State University, USA  
 Haiping Xu, University of Massachusetts Dartmouth, USA  
 Guowei Yang, Texas State University, USA  
 Hongji Yang, Leicester University, United Kingdom  
 Huiqun Yu, East China University of Science and Technology, China  
 Du Zhang, Macau University of Science and Technology, China  
 Pengcheng Zhang, Hohai University, China  
 Yong Zhang, Tsinghua University, China  
 Zhenyu Zhang, Institute of Software, Chinese Academy of Sciences, China  
 Zhigao Zheng, Central China Normal University, USA  
 Hong Zhu, Oxford Brookes University, UK  
 Huibiao Zhu, East China Normal University, China  
 Eugenio Zimeo, University of Sannio, Italy

## **PROGRAM SUB-COMMITTEE ON DISA**

Rong N. Chang IBM T.J. Watson Research Center, USA  
 Abdelrahman Osman Elfaki, University of Tabuk, Saudi Arabia  
 Ajay Kattepur, Tata Consultancy Services, India  
 Alex Norta, Tallinn University of Technology, Estonia  
 Antonella Longo, Univ. of Salento, Italy  
 Guoray Cai, Pennsylvania State University, USA  
 Honghao Gao, Shanghai University, China  
 Joe Tekli, Lebanese American University  
 Klaus-Dieter Schewe, Information Science Research Centre, New Zealand  
 Kumiko Tadano, NEC, Japan  
 Lai Xu, Bournemouth University, UK  
 Li Kuang, Hangzhou Normal University, China  
 Nanjangud Narendra, MS Ramaiah University of Applied Sciences, India  
 Nianjun Zhou, IBM T. J. Watson Res. Center, USA  
 Qing Wu, Hangzhou Dianzi University, China  
 Stephan Reiff-Marganiec, University of Leicester, UK  
 YuYu Yin, Hangzhou Dianzi University, China

## **PUBLICITY CHAIR**

Robert Heinrich, Karlsruhe Institute of Technology, Germany

## **ASIA LIAISON**

Hironori Washizaki, Waseda University, Japan

**AUSTRALASIA LIAISON**

Jing Sun, The University of Auckland, New Zealand

**EUROPE LIAISON**

Raul Garcia Castro, Universidad Politecnica de Madrid, Spain

**INDIA LIAISON**

Swapn Bhattacharya, National Institute of Technology Karnataka, Surathakl, India

**SOUTH AMERICA LIAISON**

Jose Carlos Maldonado, ICMC-USP, Brazil

# Keynote

## Quality Big Data Analytics and AI Modeling for Smart Cities

Professor Jerry Gao  
Department of Computer Engineering  
San Jose State University  
Director, SJSU Silicon Valley Excellence Research Center  
on Smart Technology, Computing, and Complex Systems

### Abstract

In this talk, Dr. Gao will share their different smart city research projects for the city of San Jose to address the top challenges in the city of San Jose, including illegal dumping, city street cleanliness, and graffiti. Dr. Gao will address the related big data analytics, and AI modeling for these projects, share their experience, challenges, and needs. In addition, Dr. Gao will review big data analytics and modeling for smart cities in terms of modeling, methods, and processes. He also shares his vision and discusses the current research issues and directions in big data quality collection, quality deep learning, and AI modeling for smart cities. Moreover, he will cover the research problems, solutions, and tools for quality training and validation for both IOT and unstructured big data. Furthermore, he will explore new challenges and issues in quality assurance for future AI systems in smart cities.

### About the Speaker

Jerry Zeyu Gao is a full professor at the Department of Computer Engineering at San Jose State University. Now, he is the director of SJSU Silicon Valley Excellence Research Center on Smart Technology, Computing, and Complex Systems. He had over 20 years of academic research and teaching experience and over 10 years of industry working and management experience on software engineering and IT development applications. He has published three technical books and over hundreds (200) publications in IEEE/ACM journals, magazines, International conferences and workshops. His current research areas include smart cities, cloud computing, TaaS, software engineering, test automation, and mobile cloud services. Since 2016, Dr. Gao and his team have worked closely with the city of San Jose on different smart city projects to address complex challenges in building future smart cities using different grants from NSF, CEC, and Cities, as well as collaborated local companies.

## Table of Contents

<b>Session Sun-I-1: Data Intensive Service-based Applications I</b>	
A New Satellite Constellation Networking Certification and Reliable Maintenance Protocol (S).....	1
<i>Congyu Huang, Liehuang Zhu, Chunlei Li, Chuan Zhang, Yuxin Chen and Zijian Zhang</i>	
A Novel Hybrid Collaborative Filtering Approach to Recommendation Using Reviews:The Product Attribute Perspective (S) .....	7
<i>Min Cao, Sijing Zhou, Honghao Gao and Youhuizi Li</i>	
Exploiting SDAE Model for Recommendations .....	11
<i>Qing Yang, Xianhe Yao, Jingwei Zhang and Zhongqin Bi</i>	
Towards business identification modeling:A Taobao Case Study (S) .....	17
<i>Rong Zhang, Yuyu Yin, Meng Xi and Hao Jiang</i>	
<b>Session Sun-II-1: Semantic Enabled Software Engineering I</b>	
Ontology-based Software Architectural Pattern Recognition and Reasoning (S) .....	23
<i>Nacha Chondamrongkul, Jing Sun and Ian Warren</i>	
Object-oriented Software Modeling with Ontologies Around - A Survey of Existing Approaches .....	29
<i>Selena Sohaila Baset and Kilian Stoffel</i>	
<b>Session Sun-III-1: Agile Software Development I</b>	
Methods for Estimating Agile Software Projects: A Systematic Review .....	34
<i>Edna Dias Canedo, Dandara Pereira Aranha, Maxwell de Oliveira Cardoso, Ruyther Parente Da Costa and Leticia Lopes Leite</i>	
Investigating gaps on Agile Improvement Solutions and their successful adoption in industry projects - A systematic literature review .....	40
<i>Arthur Freire, André Meireles, Gleyser Guimarães, Mirko Perkusich, Raissa Da Silva, Kyller Gorgônio, Angelo Perkusich and Hyggo Almeida</i>	
<b>Session Sun-I-2: Data Intensive Service-based Applications II</b>	
Towards Cost Effective Privacy Provision for Typed Resources in IoT Environment (S) ...	46
<i>Yucong Duan, Zhengyang Song, Xiaoxian Yang, Quan Zou, Xiaobing Sun and Xinyue Zhang</i>	
Finding Shilling Attack in Recommender System based on Dynamic Feature Selection ....	50
<i>Gaofeng Cao, Huan Zhang, Yuyou Fan and Li Kuang</i>	
Service Language Model: New Ecology for Service Development .....	56
<i>Ying Li, Meng Xi, Hui Chen and Jianwei Yin</i>	
<b>Session Sun-II-2: Semantic Enabled Software Engineering II</b>	

A Knowledge Engineering Approach to UML Modeling (S) .....	60
<i>Bingyang Wei, Jing Sun and Yi Wang</i>	
An Ontology-based Modelling of Vietnamese Traditional Dances (S) .....	64
<i>Truong-Thanh Ma, Salem Benferhat, Zied Bouraoui, Karim Tabia, Thanh-Nghi Do and Huu-Hoa Nguyen</i>	
<hr/> <b>Session Sun-III-2: Software Development I</b> <hr/>	
Influence Factors in Software Productivity - A Tertiary Literature Review .....	68
<i>Edson Oliveira, Tayana Conte, Marco Cristo and Natasha Valentim</i>	
Explanation Templates for Case-based Reasoning in Collaborative Risk Management .....	74
<i>Nielsen Luiz Rechia Machado, Lisandra Manzoni Fontoura, Rafael Heitor Bordini and Luís Alvaro de Lima Silva</i>	
<hr/> <b>Session Sun-I-3: Information Extraction I</b> <hr/>	
Keywords Extraction based on Sentence-Ranking from Chinese Patents .....	80
<i>Zhihong Wang, Yi Guo and Tianmei Qi</i>	
Deep Learning based Information Extraction Framework on Chinese Electronic Health Records .....	86
<i>Bing Tian, Yong Zhang, Kaixin Liu and Chunxiao Xing</i>	
Hot Topic Mining based on the Heat of Micro-blog .....	92
<i>Wang Siyao</i>	
SocialGQ: Towards Semantically Approximated and User-aware Querying of Social-Graph Data .....	98
<i>Riccardo Martoglia</i>	
<hr/> <b>Session Sun-II-3: Conceptual Lattices for Software Systems Engineering</b> <hr/>	
A Model-based Approach for Build Avoidance .....	104
<i>Milena Neumann, Kiana Busch and Robert Heinrich</i>	
Conceptual Software: The Theory Behind Agile-Design-Rules (S) .....	110
<i>Iaakov Exman</i>	
Classutopia: A Serious Game for Conceptual Modeling Design .....	116
<i>Felipe Larenas, Beatriz Marín and Giovanni Giachetti</i>	
Automatic Audience Focusing by Event Interestingness (LATTICE) (P) .....	122
<i>Iaakov Exman, Yakir Winograd and Avihu Harush</i>	
<hr/> <b>Session Sun-III-3: Information Extraction II</b> <hr/>	
Improvement of User Review Classification Using Keyword Expansion (S) .....	125
<i>Kazuyuki Higashi, Hiroyuki Nakagawa and Tatsuhiro Tsuchiya</i>	
A New Scheme for Citation Classification based on Convolutional Neural Networks .....	131
<i>Khadidja Bakhti, Zhendong Niu and Ally Nyamawe</i>	

Learning API Suggestion via Single LSTM Network with Deterministic Negative Sampling	137
<i>Jinpei Yan, Yong Qi, Qifan Rao and Hui He</i>	
Adaptive software search toward users' customized requirements in GitHub	143
<i>Jinze Liu, Zhixing Li, Tao Wang, Yue Yu and Gang Yin</i>	
<hr/> <b>Session Sun-I-4: Requirements Engineering I</b> <hr/>	
A Non-Functional Requirements Recommendation System for Scrum-based Projects	149
<i>Felipe Ramos, Antonio Alexandre Moura Costa, Mirko Perkusich, Hyggo Almeida and Angelo Perkusich</i>	
Analysis of Security-Failure Tolerant Requirements	155
<i>Michael Shin, Don Pathirage and Dongsoo Jang</i>	
An Approach for System of Systems Requirements Management (S)	161
<i>Renata Martinuzzi De Lima and Lisandra Fontoura</i>	
<hr/> <b>Session Sun-II-4: Software Quality I</b> <hr/>	
A Preliminary Investigation of Self-Admitted Refactorings in Open Source Software (S)	165
<i>Zhang Di, Bing Li, Zengyang Li and Peng Liang</i>	
Formalization and Verification of the OpenFlow Bundle Mechanism Using CSP	169
<i>Huiwen Wang, Huibiao Zhu, Yuan Fei and Lili Xiao</i>	
Helpful or Not? An investigation on the feasibility of identifier splitting via CNN-BiLSTM-CRF	175
<i>Jiechu Li, Qingfeng Du, Kun Shi, Yu He, Xin Wang and Jincheng Xu</i>	
<hr/> <b>Session Sun-III-4: Open Source Software</b> <hr/>	
How to Incorporate a Usability Technique in the Open Source Software Development Process	182
<i>Lucrecia Llerena, Nancy Rodríguez, John W. Castro and Silvia T. Acuña</i>	
XPA: An Open Source IDE for XACML Policies (S)	188
<i>Roshan Shrestha, Shuai Peng, Turner Lehmebecker and Dianxiang Xu</i>	
Automatic Detection of Public Development Projects in Large Open Source Ecosystems: An Exploratory Study on GitHub	193
<i>Can Cheng, Bing Li, Zengyang Li and Peng Liang</i>	
Recovering Three-Level Architectures from the Code of Open-Source Java Spring Projects (S)	199
<i>Alexandre Le Borgne, David Delahaye, Marianne Huchard, Christelle Urtado and Sylvain Vauttier</i>	
<hr/> <b>Session Sun-I-5: Agents</b> <hr/>	
An Agent-based Software Framework for Machine Learning Tuning	203
<i>Jefry Sastre, Marx Viana and Carlos Lucena</i>	

Accompanying Observation Modes and Software Architecture for Autonomous Robot Software .....	209
<i>Zhe Liu, Xinjun Mao and Shuo Yang</i>	
An Architecture for the Development of Ambient Intelligence Systems Managed by Embedded Agents.....	215
<i>Carlos Pantoja, Heder Dorneles Soares, José Viterbo and Amal El Fallah Seghrouchni</i>	
Understanding Normative BDI Agents Behavior .....	221
<i>Francisco Cunha, Marx Viana, Tassio Sirqueira, Marcio Rosemberg and Carlos Lucena</i>	

---

### Session Sun-II-5: Software Architecture and Frameworks

---

Towards a Representation of Enterprise Architecture based on Zachman Framework through OMG Standards (S) .....	225
<i>Miguel Ehécatl Morales Trujillo, Boris Escalante Ramírez, Maria Del Pilar Angeles, Hanna Oktaba and Guadalupe Ibarquengoitia González</i>	
STEM: A Simulation-Based Testbed for Electromagnetic Big Data Management .....	230
<i>Mengyuan Lyu, Peiquan Jin, Zhou Zhang, Shouhong Wan and Lihua Yue</i>	
Towards Reference Architecture for a Multi-layer Controlled Self-adaptive Microservice System.....	236
<i>Peini Liu, Xinjun Mao, Shuai Zhang and Fu Hou</i>	
A Heterogeneous Architecture for Integrating Multi-Agent Systems in AmI Systems (S) ..	242
<i>Carlos Pantoja, Vinicius Souza de Jesus, Fabian Manoel and José Viterbo</i>	

---

### Session Mon-I-1: Vehicular and Transportation

---

BackPocketDriver - A Mobile App to Enhance Safe Driving for Youth (S) .....	246
<i>Catherine Shanly, Michael Ieti, Ian Warren and Jing Sun</i>	
A Real-Time Ride-Sharing Matching Framework Using Simulated Annealing Genetic Algorithm.....	250
<i>Jie Xu, Yong Zhang, Chunxiao Xing and Guigang Zhang</i>	
A Multiple-Level Assessment System for Smart City Street Cleanliness .....	256
<i>Wenrui Li, Bharat Bhushan and Jerry Gao</i>	

---

### Session Mon-II-1: Security and Privacy I

---

Method and System for Detecting Anomalous User Behaviors: An Ensemble Approach ...	263
<i>Xi Xiangyu, Tong Zhang, Dongdong Du, Guoliang Zhao, Qing Gao, Wen Zhao and Shikun Zhang</i>	
Modeling and Verification of IEEE 802.11i Security Protocol for Internet of Things .....	270
<i>Yuteng Lu and Meng Sun</i>	
SeqBAC: A Sequence-Based Access Control Model (S).....	276
<i>Diogo Regateiro, Óscar Mortágua Pereira and Rui Aguiar</i>	

---

### Session Mon-III-1: Software Evolution

---

A Self-Adaptation Framework of Microservice Systems (S).....	282
<i>Shuai Zhang, Xinjun Mao, Peini Liu and Fu Hou</i>	
A Framework to Support the Development of Self-adaptive Service-oriented Mobile Applications .....	286
<i>William Passini and Frank Affonso</i>	
<hr/> <b>Session Mon-I-2: Software Modelling I</b> <hr/>	
Modeling of Interlocking Systems based on Patterns .....	292
<i>Yan Wang, Wen Zhong, Xiao Hong Chen and De Hui Du</i>	
ComD2: Family of Techniques for Inspecting Defects in Models that Affect Team Communication .....	298
<i>Adriana Lopes, Ursula Campos, Tayana Conte and Clarisse de Souza</i>	
Effects of Model Composition Techniques on Effort and Affective States: A Controlled Experiment (S) .....	304
<i>Mateus Manica, Kleinner Farias, Lucian Gonçalves, Vincius Bischoff, Bruno Carreiro Da Silva and Everton Guimarães</i>	
<hr/> <b>Session Mon-II-2: Software Testing I</b> <hr/>	
Testing Android Applications Using Multi-Objective Evolutionary Algorithms with a Stopping Criteria .....	308
<i>Anshuman Rohella and Shingo Takada</i>	
An Empirical Study on the Impact of Android Code Smells on Resource Usage .....	314
<i>Johnatan Oliveira, Markos Viggiato, Mateus Santos, Eduardo Figueiredo and Humberto Marques-Neto</i>	
Mining Intentions to Improve Bug Report Summarization .....	320
<i>Beibei Huai, Wenbo Li, Qiansheng Wu and Meiling Wang</i>	
<hr/> <b>Session Mon-III-2: Software Modelling II</b> <hr/>	
Evaluating the Effort of Integrating Feature Models: A Controlled Experiment (S).....	326
<i>Vinicius Bischoff, Kleinner Farias and Lucian Gonçalves</i>	
Modeling of software process families with automated generation of variants (S) .....	330
<i>Andrea Delgado, Daniel Calegari and Félix García</i>	
Tailored Quality Modeling and Analysis of Software-intensive Systems (S) .....	336
<i>Robert Heinrich</i>	
<hr/> <b>Session Mon-I-3: Distributed Systems</b> <hr/>	
Modeling and Verifying Leader Election Algorithm in CSP (S) .....	342
<i>Yucheng Fang, Huibiao Zhu and Huiwen Wang</i>	
A Formal Approach for Distributed Computing of Maximal Cliques in Dynamic Networks	348
<i>Faten Fakhfakh, Mohamed Tounsi, Mohamed Mosbah and Ahmed Hadj Kacem</i>	



DCCD: An Efficient and Scalable Distributed Code Clone Detection Technique for Big Code.....	354
<i>Junaid Akram, Zhendong Shi, Majid Mumtaz and Ping Luo</i>	
A Hybrid System for Detection of Implied Scenarios in Distributed Software Systems (S) .	360
<i>Anja Slama, Fatemeh Hendijani Fard and Behrouz Far</i>	
<hr/> <b>Session Mon-II-3: Software Development II</b> <hr/>	
A structured stochastic model for software project estimation in Waterfall models (S) ....	364
<i>Ildo Massitela, Joaquim Assunção, Alan Santos and Paulo Fernandes</i>	
Revisiting the Conclusion Instability Issue in Software Effort Estimation (S).....	368
<i>Michael Bosu, Solomon Mensah, Kwabena Bennin and Diab Abuaiadah</i>	
On the UML use in the Brazilian industry: A state of the practice survey (S).....	372
<i>Kleinner Farias, Lucian Gonçales, Vinicius Bischoff, Bruno Carreiro Da Silva, Everton Guimarães and Jacob Nogle</i>	
Using IFML for user interface modeling: an empirical study (S) .....	376
<i>Randerson Queiroz, Anna Beatriz Marques and Tayana Conte</i>	
<hr/> <b>Session Mon-III-3: Formal Methods</b> <hr/>	
Modeling mobility and communication in a unified way (S).....	381
<i>Jianmin Jiang, Xiaofei Yu and Zhong Hong</i>	
Towards Formal Modeling and Verification of Probabilistic Connectors in Coq (S) .....	385
<i>Xiyue Zhang and Meng Sun</i>	
Reo2PVS: Formal Specification and Verification of Component Connectors .....	391
<i>M. Saqib Nawaz and Meng Sun</i>	
Modeling and Analyzing Hybrid Systems Using Hybrid Predicate Transition Nets (S).....	397
<i>Dewan Mohammad Moksedul Alam, Xudong He and William Chu</i>	
<hr/> <b>Session Mon-I-4: Visualization</b> <hr/>	
Visualizing Interactions in AngularJS-based Single Page Web Applications .....	403
<i>Gefei Zhang and Jianjun Zhao</i>	
Software Visualization Using Topic Models.....	409
<i>Sandeep Reddivari and William Hackney</i>	
<hr/> <b>Session Mon-II-4: Software Defect Management I</b> <hr/>	
How Many Versions does a Bug Live in? An Empirical Study on Text Features for Bug Lifecycle Prediction .....	415
<i>Chuanqi Wang, Yanhui Li and Baowen Xu</i>	
Bayesian Logistic Regression for software defect prediction (S).....	421
<i>Jinu M Sunil, Lov Kumar and Lalita Bhanu Murthy Neti</i>	

Revisiting the Impact of Regression Models for Predicting the Number of Defects . . . . .	427
<i>Wu Man, Ye Sizhe, Li Chunhua, Ma Ziyi and Fu Zhongwang</i>	

---

#### **Session Mon-III-4: Sentimental Analysis and User Experience**

---

A Gated Hierarchical LSTMs for Target-based Sentiment Analysis . . . . .	433
<i>Xiaofang Zhang, Bin Liang, Qian Zhou, Hao Wang and Baowen Xu</i>	
Does Ad-Context Matter on the Effectiveness of Online Advertising? . . . . .	439
<i>Caihong Sun, Meina Zhang and Meiyun Zuo</i>	
Analyzing The Impact Of Feedback In GitHub On The Software Developer's Mood . . . . .	445
<i>Mateus Santos, Josemar Caetano, Johnatan Oliveira and Humberto T. Marques-Neto</i>	
Do Scale Type Techniques Identify Problems that Affect User eXperience? User Experience Evaluation of a Mobile Application (S) . . . . .	451
<i>Leonardo Marques, Walter Nakamura, Natasha Valentim, Luis Rivero and Tayana Conte</i>	

---

#### **Session Mon-I-5: Software Development III**

---

A Systematic Approach for Developing Cyber Physical Systems . . . . .	456
<i>Xudong He, Zhijiang Dong and Yujian Fu</i>	
Investigating Technical Debt Folklore: A Replicated Survey . . . . .	462
<i>Nicolli Rios, José Amâncio Santos, Manoel Mendonça and Rodrigo Spinola</i>	
Knowledge Management Governance in Software Development Process with GI-Tropos . . . . .	468
<i>Vu Nguyen Huynh Anh, Manuel Kolp and Yves Wautelet</i>	
A Search-based Software Engineering Approach to Support Multiple Team Formation for Scrum Projects . . . . .	474
<i>Alexandre Costa, Felipe Ramos, Mirko Perkusich, Arthur Freire, Hyggo Almeida and Angelo Perkusich</i>	

---

#### **Session Mon-II-5: Software Defect Management II**

---

Bug or Not Bug? Labeling Issue Reports via User Reviews for Mobile Apps (S) . . . . .	480
<i>Haoming Li, Tao Zhang and Ziyuan Wang</i>	
A Topic Modeling Approach for Code Clone Detection . . . . .	486
<i>Sandeep Reddivari and Mohammed Salman Khan</i>	
XMILE - An Expert System for Maintenance Learning from Textual Reports (S) . . . . .	492
<i>Eduardo Máximo and Vladia Pinheiro</i>	

---

#### **Session Tue-I-1: Agile Software Development II**

---

Effort Estimation in Agile Software Development: an Updated Review . . . . .	496
<i>Emanuel Dantas Filho, Mirko Perkusich, Ednaldo Dilorenzo, Danilo Santos, Hyggo Almeida and Angelo Perkusich</i>	

SIDD – SCRUM Iteration Driven Development: An Agile Software Development and Management Process Based on SCRUM (S) .....	502
<i>Tayse Virgulino Ribeiro, Cristina D’Ornelas Filipakis Souza and Heloise Acco Tives Leão</i>	
Investigating the Effects of Agile Practices and Processes on Technical Debt - The Viewpoint of the Brazilian Software Industry .....	506
<i>Viviane Caires, Nicolli Rios, Johannes Holvitie, Ville Leppänen, Manoel Mendonça and Rodrigo Spinola</i>	
<hr/> <b>Session Tue-II-1: Real-time Systems</b> <hr/>	
Timing Analysis for Microkernel-based Real-Time Embedded System .....	512
<i>Rongfei Xu, Li Zhang, Ning Ge and Jing Jiang</i>	
Schedulability Analysis of Real-time Tasks with Precedence Constraints .....	518
<i>Rongfei Xu, Li Zhang, Ning Ge and Xavier Blanc</i>	
<hr/> <b>Session Tue-III-1: Requirements Engineering II</b> <hr/>	
Conflict Management in the Collaborative Description of a Domain Language (S) .....	524
<i>Claudia Litvak, Gustavo Rossi and Leandro Antonelli</i>	
Belief Function Theory in Constraint Satisfaction Problems: a Unifying Approach .....	530
<i>Aouatef Rouahi, Ben Salah Kais and Ghedira Khaled</i>	
<hr/> <b>Session Tue-I-2: Software Testing II</b> <hr/>	
DevOps Enhancement with Continuous Test Optimization .....	536
<i>Dusica Marijan and Sagar Sen</i>	
Reducing the Cost of Android Mutation Testing .....	542
<i>Lin Deng and Jeff Offutt</i>	
A Test Case Generation Method Based on State Importance of EFSM for Web Application	548
<i>Junxia Guo, Weiwei Wang, Linjie Sun, Zheng Li and Ruilian Zhao</i>	
<hr/> <b>Session Tue-II-2: Software Quality II</b> <hr/>	
Parallel Property Checking with Symbolic Execution .....	554
<i>Junye Wen and Guowei Yang</i>	
Software Process Improvement Programs: What happens after official appraisal? .....	560
<i>Regina Albuquerque, Andreia Malucelli and Sheila Reinehr</i>	
Improving code summarization by combining deep learning and empirical knowledge (S) ..	566
<i>Lingbin Zeng, Xunhui Zhang, Tao Wang, Xiao Li, Jie Yu and Huaimin Wang</i>	
<hr/> <b>Session Tue-III-2: Components and Memory Management</b> <hr/>	
Reverse Engineering Encapsulated Components from Object-Oriented Legacy Code .....	572
<i>Rehman Arshad and Kung-Kiu Lau</i>	

Leveraging the Power of Component-based Development for Front-End Components: Insights from a Study of React Applications (S) .....	578
<i>Chen Yang, Yan Liu, Jia Yu and Yiwei Lin</i>	
A Lightweight Approach to Detect Memory Leaks in JavaScript (S) .....	582
<i>Ju Qian, Long Wang and Xiaoyu Zhou</i>	
<hr/> <b>Session Tue-I-3: Software Quality III</b> <hr/>	
Pseudo-Exhaustive Verification of Rule Based Systems .....	586
<i>Rick Kuhn, Dylan Yaga, Raghu Kacker, Jeff Lei and Vincent Hu</i>	
Metrics for Data Uniformity of User Scenarios through User Interaction Diagrams (S) ....	592
<i>Douglas Hiura Longo and Patrícia Vilain</i>	
Feedback Topics in Modern Code Review: Automatic Identification and Impact on Changes .....	598
<i>Janani Raghunathan, Lifei Liu and Huzeefa Kagdi</i>	
Expediting Binary Fuzzing with Symbolic Analysis .....	604
<i>Luhang Xu, Wei Dong, Liangze Yin, Weixi Jia and Shenzhi Li</i>	
<hr/> <b>Session Tue-II-3: Computational Intelligence, Models and Algorithms</b> <hr/>	
Topic Modeling for Noisy Short Texts with Multiple Relations .....	610
<i>Chiyu Liu, Zheng Liu, Tao Li and Bin Xia</i>	
Svega: Answering Natural Language Questions over Knowledge Base with Semantic Matching .....	616
<i>Gaofeng Li, Pingpeng Yuan and Hai Jin</i>	
Software Process Selection based upon Abstract Machines for Slow Intelligence Systems ..	622
<i>Shikuo Chang, JinPeng Zhou, Akhil Yendhuri and Kadie Clancy</i>	
Evolutionary propositionalization of multi-relational data .....	629
<i>Valentin Kassarnig and Franz Wotawa</i>	
<hr/> <b>Session Tue-III-3: Software Testing III</b> <hr/>	
On A Simpler and Faster Derivation of Single Use Reliability Mean and Variance for Model-Based Statistical Testing (S) .....	635
<i>Yufeng Xue, Lan Lin, Xin Sun and Fengguang Song</i>	
A Document-based Parameter Correlation Metric for Test Design (S) .....	641
<i>Hiroyuki Nakagawa, Nobukazu Ishii and Tatsuhiro Tsuchiya</i>	
Improving Integration Testing of Web Service by Propagating Symbolic Constraint Test Artifacts Spanning Multiple Software Projects (S) .....	647
<i>Andreas Fuchs and Vincent von Hof</i>	
Prioritizing Unit Testing Effort Using Software Metrics and Machine Learning Classifiers (S) .....	653
<i>Fadel Toure and Mourad Badri</i>	
<hr/> <b>Session Tue-I-4: Security and Privacy II</b> <hr/>	

Security Analysis of the Access Control Solution of NDN Using BAN Logic (S) .....	659
<i>Yuan Fei, Huibiao Zhu and Huiwen Wang</i>	
Re-checking App Behavior against App Description in the Context of Third-party Libraries .....	665
<i>Chengpeng Zhang, Haoyu Wang, Ran Wang, Yao Guo and Guoai Xu</i>	
Whether Android Applications Broadcast Your Private information: A Naive Bayesian-based Analysis Approach (S) .....	671
<i>Li Lin, Jian Ni, Xinya Mao and Jianbiao Zhang</i>	
<hr/> <b>Session Tue-II-4: Software Quality IV</b> <hr/>	
Model Checking Method for SPA Page Transition Based on Component-based Framework	675
<i>Naito Oshima and Tomoji Kishi</i>	
A Systematic Mapping Study on Software Comments Analysis .....	681
<i>Amanda Passos, Mário Farias, Crescencio Lima, Manoel Mendonça and Rodrigo Spinola</i>	
Process metrics for system quality with specifications' shifts from a bid phase to an operation phase (S) .....	687
<i>Noriko Hanakawa and Masaki Obana</i>	
<hr/> <b>Session Tue-III-4: Recommendation</b> <hr/>	
Exploratory Recommender Systems Based on Reinforcement Learning for Finding Research Topic .....	691
<i>Li Yu and Zhuangzhuang Wang</i>	
Evaluating Multiple User Interactions for Ranking Personalization Using Ensemble Methods .....	697
<i>Frederico Durao, Bruno Souza Cabral, Marcelo Manzato and Arthur Fortes Da Costa</i>	
<hr/> <b>Session Sun-III-5: Poster and Demo</b> <hr/>	
Weighted Data Set Reduction for Automatic Bug Triaging (P) .....	703
<i>Miaomiao Wei, Shikai Guo and Rong Chen</i>	
Integrating Challenge Based Learning Into a Smart Learning Environment: Findings From a Mobile Application Development Course (P) .....	704
<i>Rafael Chanin, Alan Santos, Nicolas Nascimento, Afonso Sales, Leandro Pompermaier and Rafael Prikladnicki</i>	
A Personalized Metasearch Engine Based on Multi-agent System (P) .....	707
<i>Meijia Wang, Qingshan Li and Yishuai Lin</i>	
Interval-valued Data Clustering Based on Range Metrics (P) .....	710
<i>Sérgio Galdino, Wellington Santos and Ricardo Paranhos</i>	
A Revisit of Fault-Detecting Probability of Combinatorial Testing for Boolean-Specifications (P) .....	711
<i>Min Yu, Ziyuan Wang, Feiyan She and Yuanchao Qi</i>	

Big Data ETL Implementation Approaches: A Systematic Literature Review (P) .....	714
<i>Joshua Nwokeji, Faisal Aqlan, Apoorva Anugu and Ayodele Olagunju</i>	
Research on Crowd-based mobile application testing platforms (P) .....	716
<i>Wenguang Xie and Kenian Wang</i>	
Mobile App Development Using Software Design Patterns (P) .....	718
<i>Nicole Barakat and Doan Nguyen</i>	
BoolMuTest: A Prototype Tool for Fault-Based Boolean-Specification Testing (P) .....	720
<i>Ziyuan Wang</i>	

Note: (S) indicates a short paper

(P) indicates a poster or demo description



# A New Satellite Constellation Networking Certification and Reliable Maintenance Protocol(DISA)

1<sup>st</sup> Congyu Huang  
School of Computer Science  
Beijing Institute of Technology  
Beijing, China  
Email:2120171018@bit.edu.cn

2<sup>nd</sup> Liehuang Zhu  
School of Computer Science  
Beijing Institute of Technology  
Beijing, China  
Email:liehuangz@bit.edu.cn

3<sup>rd</sup> Chunlei Li  
China TravelSky Holding Company  
Beijing, China  
Email: lcl@travelsky.com

4<sup>th</sup> Chuan Zhang  
School of Computer Science  
Beijing Institute of Technology  
Beijing, China  
Email:chuanz@bit.edu.cn

5<sup>th</sup> Yuxin Chen  
School of Computer Science  
Beijing Institute of Technology  
Beijing, China  
Email:realcyx@126.com

6<sup>th</sup>Zijian Zhang  
School of Computer Science  
Beijing Institute of Technology  
Beijing, China  
Email:zhangzijian@bit.edu.cn

**Abstract**—With the rapid development of satellite technology, the deployment of intensive service applications through satellite has become a trend. In the process of establishing a satellite communication system, there will be some security threats such as counterfeiting, forgery, tampering. This must establish a secure satellite communication system. In this paper, according to the characteristics of satellite communication system, a protocol of satellite network authentication and trusted maintenance is designed. The protocol can accomplish two-way authentication between entities in the satellite network and the credible maintenance of the communication link. The protocol is based on the symmetric encryption system and can adapt to the current satellite load is small, the computing power is limited. This paper also analyses the security of the protocol and can resist replay attacks and man-in-the-middle attacks. Experiments show that the proposed network authentication protocol is 28% faster than the symmetric encryption system. The average time to keep the agreement credible is 254.64 ms.

**Index Terms**—Protocols, Security authentication, Reliable maintenance, GEO/LEO satellite networks

## I. Introduction

With the continuous progress of science and technology, satellite communication system such as the Iridium system and the Globalstar system are becoming more and more popular. In those systems, satellite provide a wide range of significant services, including Weather forecasting, TV signal transmitting, global positioning, communicating and Internet accessing, etc [1]–[4]. Nowadays, the satellite can provide more and more services and the satellite network is more and more perfect. It is an inevitable trend to deploy data intensive services

based application in satellite, so we need to build a robust network.

Meanwhile, in satellite networks, one of the biggest challenges is how to work more securely over the network. The satellite network has the characteristics of open channel, large transmission delay, intermittent link connection and so on [5]–[7]. These characteristics determine that satellite networks are more vulnerable to counterfeiting, tampering and other security threats than traditional networks. How to resist these security threats has become an important research direction. A simplest way is to authenticate network entities and users.

In order to build a robust and security network, many scholars have proposed many solutions to improve the security and robustness of satellite networks. There are some protocols guaranteed the security and some strategies provide robustness in present satellite network. Reference [8] propose a public key cryptosystem-based authentication technology. However, the authentication technology is unidirectional and can not meet the current need for bidirectional authentication. Reference [9] design and implement a two-way authentication protocol between the client and the satellite, but the authentication protocol has high maintenance cost and high failure risk. Reference [10] put forward a double-layered inclined orbit constellation to improve the robustness of satellite communication network. But they do not consider about security in the network.

We believe that this paper makes the following contributions:

(1) This paper proposes a satellite constellation network authentication protocol for double-layered satellite constellation. The network authentication protocol takes into account the characteristics of the satellites, and the

Foundation Items: The National Key Research and Development Program of China(2016YFB0800301)

DOI reference number: 10.18293/SEKE2018-041



satellite will be carried out one by one, and the satellites will be gradually authenticated by the network.

(2) This paper presents a reliable maintenance protocol, which uses geostationary-earth-orbit (GEO) satellites to control low-earth-orbit (LEO) satellite clusters, and realizes operations such as key renewal and revocation for LEO satellites under link connectivity. It ensures reliable communication and reliability of low orbit satellites.

(3) In this paper, we have analysed the security of the network authentication protocol and the security of the reliable maintenance protocol. Afterwards, we realize the satellite network protocol and the trusted maintenance protocol by experimental simulation. In the network authentication protocol, we compare with the traditional public key system and symmetric system, the efficiency of the network authentication protocol in this paper accounts for 28% which is faster than the original symmetric system.

## II. Related Work

### A. Security Authentication Protocol for Satellite Network

Zhibo, X et al. [11] put forward an end-to-end authentication protocol for satellite networks based on the Internet key exchange (IKE) protocol. The protocol is based on the IKE protocol and IKE is based on the public key encryption system. The calculation cost is high and the number of key negotiation interactions is more frequent.

Chang et al. [12] propose an authentication and key agreement protocol for satellite communications, this protocol is mainly about the authentication security between users and satellite. The protocol is not suitable for the direct use of authentication between satellites.

### B. Satellite Network Reliable maintenance

X Jin et al. [13] propose a communication framework between satellite and ground station in order to improve the robustness of satellite network. Meanwhile, propose a communication architecture of space ground integrated information network which adopts simplified IP protocol, and analyse the feasibility of the implementation is analysed from the view of business process. But this framework does not consider about the communication security.

Kimura et al. [14] propose a double-layered inclined orbit constellation for satellite communication network connected by optical inter-satellite links. But this architecture is not secure because it does not take into account that will be attacked during the link transfer process. The security of information transmission in the link will be threatened greatly.

## III. Satellite Constellation Network Authentication and Reliable Maintenance Protocol

Satellite network models include user terminals (UT), ground control center (GCC) and GEO satellite constellation networks, as well as LEO satellite constellation networks. Due to the need for network authentication between

GEO satellites, we need to design onboard network authentication protocol to adapt the actual situation of GEO satellites. Since the existence of LEO satellite networks in the system, many LEO satellites can not be directly connected with GCC because of the characteristics of LEO satellite networks. Therefore, it is necessary to study the adaptive and trustworthy link of links in resource-limited environments. In order to formally describe the model of the program, the program first establishes the model as follows.

The model of the satellite network is shown in Fig. 1.

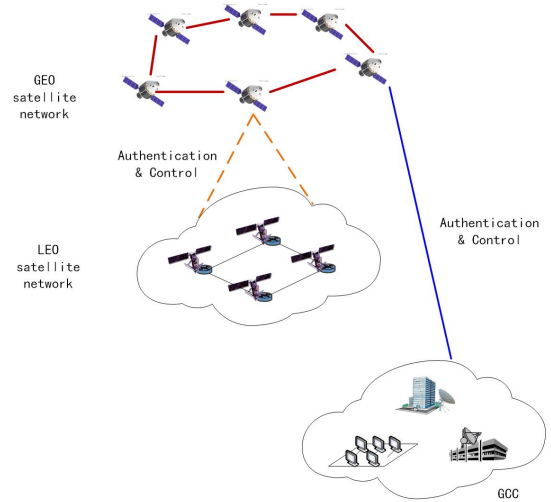


Fig. 1. Satellite network models

- GEO satellite network.  $GSN$  indicates the GEO satellite network, which consists of GEO satellites and GEO satellite links. The GEO satellite network can be represented by an undirected graph  $GSN = (GV, GE)$ , where  $GV$  represents a GEO satellite node like vertex in graph and  $GE$  represents a GEO satellite link like edge in graph.
- GEO satellite node.  $GV$  indicates the GEO satellite in a GEO satellite network and it is denoted as  $\langle n^{GV}, s^{GV}, c^{GV} \rangle$ , where  $n^{GV}$  Satellites node number, uniquely identifies a GEO satellite;  $s^{GV}$  is a security attribute that indicates satellite-mounted authentication information and protocols.  $c^{GV}$  indicates control information used to control LEO satellites.
- LEO satellite node.  $LV$  is a LEO satellite, and it is also possible to have a number, security attribute, and control attribute, denoted as  $\langle n^{LV}, s^{LV}, c^{LV} \rangle$ .

The model is mainly used to achieve GEO satellite network authentication and credible maintenance. In this model, a high-orbit satellite,  $GV$ , is launched in turn. Based on the satellite security attribute  $s^{GV}$  on the  $GV$ , mutual authentication of GEO satellites and GCC can be achieved by selecting the satellite and ground authentication mode. If the certification is successful, the GEO satellite can access GCC and GCC can control the

GEO satellite and distribute the key. If the authentication fails, GCC denies access and the GEO satellite refuses the control.

After launching  $GV$ , the satellite security attribute module  $s^{GV}$  is used to select inter-satellite authentication. If the satellite authentication is successful, then a secure communication link is established between  $GV$ , whereas the two satellites can not communicate with each other. All  $GV$  satellite between the realization of the certification, all high-orbiting satellites in mutual authentication, the realization of  $GV$  networking completed  $GSN$  build.

In addition, for the LEO satellite network, due to the relatively high-speed trajectory of  $LV$ , it is difficult for GCC to control it directly. To achieve the credible maintenance of the system,  $GV$  needs to support the control of  $LV$  achieve credible maintenance. In order to achieve the security of control, it should be authenticated between  $GV$  and  $LV$ , and use the  $GV$  security attribute  $s^{GV}$ , and the type options are  $GV$  and  $LV$ . If the authentication succeeds, the control channel will establish, and  $GV$  can control  $LV$  through the secure channel, otherwise, it can not be controlled.

#### A. Satellite Constellation Network Authentication

##### 1) Satellite and GCC networking Authentication:

Based on the definition in the abstract model, we know that the high-orbit satellite  $GV$  is  $\langle n^{GV}, g^{GV}, s^{GV}, c^{GV} \rangle$ . Before launching of the satellite numbering, in accordance with the launch of the satellite numbering, may wish to set  $G1, G2, \dots, GM$ . Need to set security attributes satellites, security attributes need to be defined cryptographic algorithms, keys and authentication protocols.

GEO satellites carry all the symmetric keys with LEO satellites, and the satellites use pairs of keys for authentication. Due to the limited computing power of satellites, the authentication protocol is based on a symmetric key design. The former satellite carries the symmetry  $K_{G_i}$  for itself and the control center, presets the symmetric key  $K_{G_{ij}}$  used for authentication between the satellites. The original satellite sends the symmetric key to the satellite already in orbit, afterwards the satellite uses the key  $K_{G_{ij}}$  for satellite authentication.

2) Authentication between GEO Satellites: Because the satellites need to be launched one by one in the process of satellite, that is, in the process of satellite networking, the satellite needs to be gradually accessed to the network, so in this process, the certification of satellites is different. When launching the first satellite, space satellites have not been networked yet. At this moment, the satellite authentication is authenticated based on the key set in advance. After the first one is authenticated, the satellite of the second backbone network is deployed and controlled  $K_{G_1}$  for center certification, and  $K_{G_{12}}$  and SQN sequences for the first and second satellite certifications.

First of all, it is determined whether the inter-satellite certification link can be constructed with the adjacent satellites. If an inter-satellite certification link can be constructed, the first and second inter-satellite authentication can be established by using the secure communication channel between the network service center and the previous satellite. The symmetric key  $K_{G_{12}}$  and the SQN sequence are sent to the first satellite so that both the first satellite and the second satellite have the authentication key  $K_{G_{12}}$  and the SQN sequence. The specific process of certification is shown in Fig. 2. The process behind the satellite is similar. After all the satellites are launched, a high-orbit satellite network is established between the satellites, thus completing the satellite networking Authentication process.

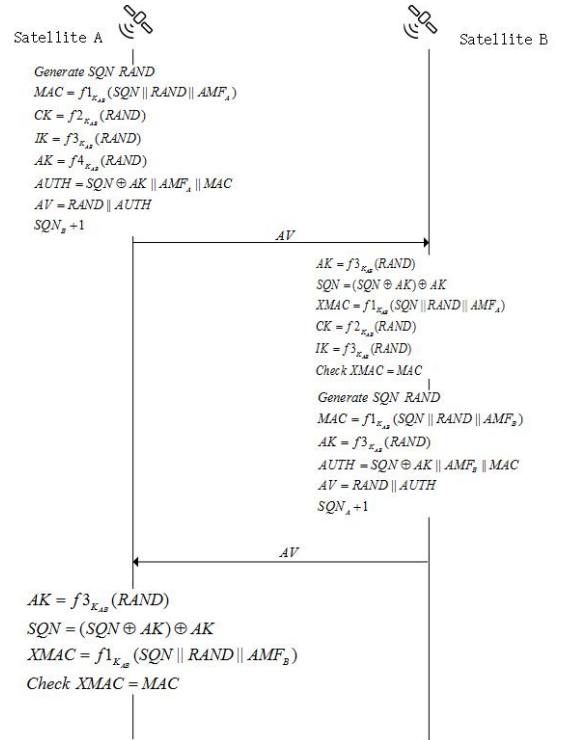


Fig. 2. Authentication between Satellites

(1) First, satellite A sends an authentication request message to satellite B. Before the satellite A initiates the authentication request, it needs to calculate the authentication vector AV based on its own key  $K_{AB}$ , where the authentication vector consists of the following three elements, namely the random number RAND, the session key for encryption CK, authentication token AUTH. Each vector AV generation process is as follows. Generate sequence number SQN and random number RAND, and calculate  $MAC = f1_{K_{AB}}(SQN || RAND)$   $CK = f2_{K_{AB}}(RAND)$   $AK = f3_{K_{AB}}(RAND)$   $AUTH = SQN \oplus AK || MAC$   $AV = RAND || AUTH$ . Satellite A then sends the vector AV to satellite B.

(2)After satellite B receives the AV from satellite A, satellite B calculates AK using  $K_{AB}$  through RAND, decrypts it with AK to obtain SQN and verifies whether satellite A is valid by computing  $f1_{K_{AB}}(SQN||RAND)$ . Has a symmetric key  $K_{AB}$ .After the verification is passed, a new random number  $RAND$  is generated and calculates  $MAC = f1_{K_{AB}}(SQN||RAND)$   $CK = f2_{K_{AB}}(RAND)$   $AK = f3_{K_{AB}}(RAND)$   $AUTH = SQN \oplus AK||MAC$   $AV = RAND||AUTH$ .Then sent the vector AV to satellite A.

(3)After satellite B's AV is received by satellite A, AK is computed by RAND using  $K_{AB}$  and decrypted by AK to obtain SQN, verifying whether satellite B is valid by computing  $f1_{K_{AB}}(SQN||RAND)$ . Have a symmetric key  $K_{AB}$ , verify the success, then the certification process is completed.

In the algorithm, the function  $f1$  used as a message verification code generation function, and  $f2$  and  $f3$  are key derivation functions [15].

## B. Reliable Maintenance

The main needs of GEO satellites for reliable maintenance of LEO satellites are divided into two steps. The first step is to authenticate connection between the GEO satellites and the LEO satellites. The second step is to control the LEO satellites by using the control attributes of the GEO satellites.

1) Authentication between GEO satellite and LEO satellite: After GEO satellite network and LEO satellite network set up to achieve double-layer satellite networking, access authentication is achieved through LEO satellite network, and backup and trusted maintenance is achieved through GEO satellite network.

In the current international environment, basically all countries can not be deployed on a global scale. When satellite authentication systems need to be updated, there is no guarantee that all satellite satellites will be able to travel through one country, which may result in a failure to update all of them during the update. In this case, GEO satellites are required to cover the LEO satellites.

After the completion of the construction of the double-layer network, it is also necessary to consider the process of completing the certification of the LEO satellites and the LEO satellites during use. The certification process between LEO and LEO satellites is similar to the process of certification between GEO and GEO satellites, as detailed in the earlier section on certification between GEO satellites.

The reliable maintenance of the GEO satellite network for the LEO satellite network is to update the authentication module, such as the authentication key in the system of the LEO satellite network. When a satellite is out of work, the packets that were originally forwarded through the failed satellite will no longer pass the invalid satellite. At this time, we need to broadcast the entire network to the failed satellites in the LEO

satellite network by the high orbit satellite, which will make the related topological paths of the failed satellites change to infinity, indicating that the other satellites which satellite is failed, and will invalidate the failed satellite's authentication key. The process is shown in Fig. 3.

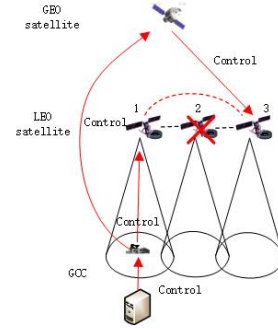


Fig. 3. Control the invalid LEO satellite

The following example illustrates the failure of a high-orbiting satellite to describe the process of updating keys and reestablishing a secure communication channel between adjacent high-satellites after symmetric key deletion. The process is shown in Fig. 4.

(1)For authentication between satellite and GCC, we use 3GPP's authentication protocol [15]. The satellite A and GCC are authenticated and the session key  $CK_1$  and the integrity key  $IK_1$  are obtained. The satellite B and GCC are authenticated and the session key  $CK_2$  and the integrity key  $IK_2$  are obtained. After completing the above process, two satellites and GCC secure communication channels are established.

(2)The GCC assigns the symmetric keys  $K_{AB}$  and  $ID_B$  to the satellite A through the secure channel. At the same time, the GCC secure channel allocates the symmetric keys  $K_{AB}$  and  $ID_A$  to the satellite B.

(3)Satellite A and satellite B complete the certification. For details on the authentication method, see Satellite Constellation Network Authentication Section.

The completion of the above steps will enable the implementation of satellite key update and reliable maintenance.

## IV. Security Analysis and Performance Analysis

### A. Security Analysis

In the satellite network authentication protocol, the protocol can accomplish two or two-way satellite authentication. The protocol used is based on the 3GPP protocol and uses symmetric keys to ensure the privacy of the protocol. The protocol uses message authentication codes to ensure the integrity of information during transmission, thus being able to resist attacks such as counterfeiting and forgery. Due to the large transmission delay of the satellite, the use of timestamps to resist replay attacks is less controllable. We use the serial number SQN instead of timestamps to protect against replay attacks. In the

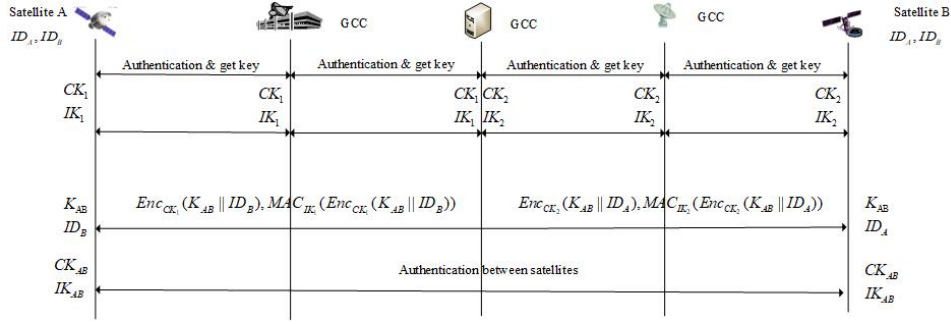


Fig. 4. Control the invalid LEO satellite protocol

process of satellite communication, key transmission is established in the secure channel, which can effectively resist man-in-the-middle attacks.

In the reliable maintenance protocol, the control link first passes two-way authentication and generates the session key and the integrity key. This ensures the privacy and completeness of the process of transmitting the update key in the control link. During the process of re-establishing the link, the protocol derives the symmetric authentication key and integrity key using the symmetric key derivation function to protect the underlying symmetric key to ensure the privacy and integrity of the protocol. Through the above two methods, it can resist attacks such as counterfeiting and counterfeiting in the satellite environment. The agreement also uses the serial number SQN to resist replay attacks.

### B. Performance Analysis

In order to test the performance of the protocol, the protocol was simulated to analyze the performance under an Intel (R) Core i7-7700HQ CPU@2.80GHz processor. Respectively, the network authentication protocol and trusted to keep the agreement has been tested. This experiment uses openssl open source library security algorithm for experiments.

In this paper, 100 tests of network authentication were conducted and compared with the traditional symmetry scheme, the test results shown in Fig.5. Due to the large satellite delay, we removed the satellite communication delay and compared the calculation times of the two authentication schemes. The protocol calculation time is shown in Fig. 6.

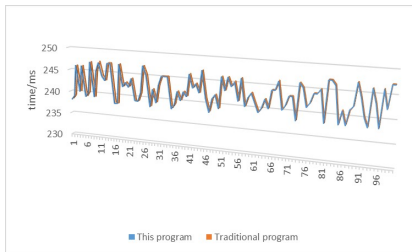


Fig. 5. Network authentication performance test results

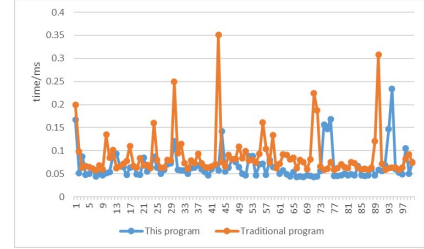


Fig. 6. Protocol calculation time

In the experimental results, the maximum calculation time of this scheme is 0.339 ms, the minimum time is 0.048 ms and the average calculation time is 0.074 ms. The maximum calculation time of the the traditional symmetry authentication scheme is 0.289 ms, the minimum calculation time is 0.061 ms and the average calculation time is 0.093 ms. The average efficiency of this scheme is 28% higher than that of public key encryption schemes.

This article also tests the trusted maintenance for 10 times, and the test results are shown in figure 7. As the satellite failure increases, the test results are shown in figure 8.

The experimental results, the maximum time of this program for reliable maintenance is 272.14 ms, the minimum time is 238.20 ms, the average time is 254.64 ms.

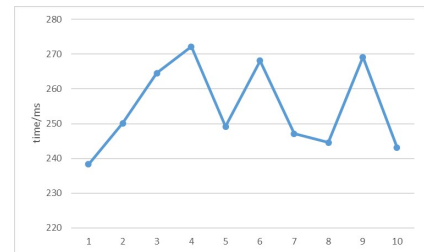


Fig. 7. Reliable maintenance of test results

## V. Conclusion

This paper designs a new GEO satellite network authentication and credible maintenance protocol, verifies the feasibility and efficiency of the protocol experimentally,

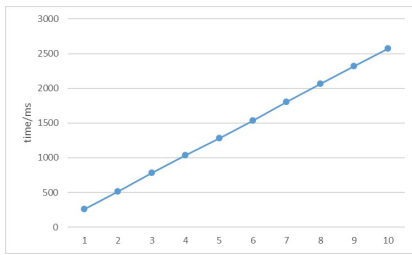


Fig. 8. Failure satellite increasing test results

[15] 3GPP, TS 33.102 v9.1.0. 3G Security; Security architecture (Release 9), 2009.12

and proves the security of the protocol through theoretical analysis.

Future satellite computing power will be more and more strong, the use of public key system on the satellite will be the new direction. For credible maintenance of LEO satellites, other safer and more efficient options may emerge in the future.

### References

- [1] Amirshahi, Pouyan, and Steven Grippando. "Radio frequency interference monitoring system for weather satellite ground stations: Challenges and opportunities." *Dynamic Spectrum Access Networks (DySPAN)*, 2017 IEEE International Symposium on. IEEE, 2017.
- [2] Berman, Elliot. "Movable window support device for a satellite TV dish." U.S. Patent No. 6,731,250. 4 May 2004.
- [3] De Sanctis, Mauro, et al. "Satellite communications supporting internet of remote things." *IEEE Internet of Things Journal* 3.1 (2016): 113-123.
- [4] Ashjaee, Javad, et al. "Satellite differential positioning receiver using multiple base-rover antennas." U.S. Patent No. 9,035,826. 19 May 2015.
- [5] LI F H , YIN L H , WU W ,et al. Research status and development trends of security assurance for space-ground integration information network[J]. *Journal on Communications*, 2016,37(11): 156-168.
- [6] Jiang, Chunxiao, et al. "Security in space information networks." *IEEE communications magazine* 53.8 (2015): 82-88. Zheng, Gan, Pantelis-Daniel Arapoglou, and Bjorn Ottersten.
- [7] "Physical layer security in multibeam satellite systems." *IEEE Transactions on wireless communications* 11.2 (2012): 852-863.
- [8] Willems C, Pozzobon O, Kubik K. Signal Authentication and Integrity Schemes for Next Generation Global Navigation Satellite Systems[C]// *European Navigation Conference Gnss*. 2005:1.
- [9] Cruickshank H S. A security system for satellite networks. *Proceedings of the Fifth International Conference on Satellite Systems for Mobile Communications and Navigation*, London, UK, 1996.
- [10] Sasaki, Takao, and Tadayoshi Katoh. "Dual layer satellite communications system and geostationary satellite therefor." U.S. Patent No. 6,023,605. 8 Feb. 2000.
- [11] Zhibo, X., Ma, H.: Design and simulation of security authentication protocol for satellite network. *Comput. Eng. Appl.* 43(17), 130–132 (2007)
- [12] Chang, Chin-Chen, Ting-Fang Cheng, and Hsiao-Ling Wu. "An authentication and key agreement protocol for satellite communications." *International Journal of Communication Systems* 27.10 (2014): 1994-2006.
- [13] Jin, Xiaoning, Peiyang Zhang, and Haipeng Yao. "A communication framework between backbone satellites and ground stations." *Communications and Information Technologies (ISCIT)*, 2016 16th International Symposium on. IEEE, 2016.
- [14] Kimura, Kazuhiro, Keizo Inagaki, and Yoshio Karasawa. "Double-layered inclined orbit constellation for advanced satellite communications network." *IEICE Transactions on Communications* 80.1 (1997): 93-102.

# A Novel Hybrid Collaborative Filtering Approach to Recommendation Using Reviews: The Product Attributes Perspective

Min Cao<sup>1</sup>, Sijing Zhou<sup>1</sup>, Honghao Gao<sup>1,2,3,\*</sup>, Youhuizi Li<sup>4</sup>

<sup>1</sup>, School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China

<sup>2</sup>, Computing Center, Shanghai University, Shanghai 200444, China

<sup>3</sup>, Shanghai Key Laboratory of Computer Software Testing & Evaluating, Shanghai 201112, China

<sup>4</sup>, College of Computer, Hangzhou Dianzi University, Hangzhou 310018, China

mcao@staff.shu.edu.cn, zhousijing@shu.edu.cn, gaohonghao@shu.edu.cn, huizi@hdu.edu.cn

**Abstract**—The product recommendation research has been focusing on modelling users' reviews to construct the relation of users and products. Thus, the recommended performance can be improved by obtaining virtual ratings from corresponding reviews. However, these perspectives on reviews do not take into account the product field characteristic, which may impact the recommendation performance. To this point, this paper proposes a hybrid collaborative filtering approach to compute the correlation value considering product attributes. First, *Product Attribute Weight* and *Product Attribute Score* are introduced to formalize the product attributes for user and product respectively in a quantitative way. After that, the recommended ranking formula for the new model is presented. Finally, we carry out experimental analysis to show our method can effectively improve the performance of recommendation under a sparseness dataset.

**Keywords** — *Product Recommendation, Reviews, Hybrid Collaborative Filtering, Product Attributes*

## I. INTRODUCTION

The recommender system originated from information retrieval has been served to provide users with personalized online product recommendations to improve user experience [1][2]. With the increasing information on line, the performance of the recommendation using reviews becomes a crucial problem to modern service industry. However, there are some deficiencies in the existing product recommendation approaches using product reviews.

First, the mainstream of recommendation methods using reviews is usually using aspect preference, such as *aspect need* and *aspect importance* [2-4]. The implicit condition in aspect preference methods is reviews' characteristic of centralized features. But the features of product reviews are scattered and not uniform because the reviews have multiple categories and are numerous. Therefore, aspect preference methods bring disunity problem of features and are not suitable for product field.

Second, product attributes is proved that affects consumers' desire for consumption [5-8]. The expression of product

attributes was once focused on the calculation of weight values [2]. Until the introduction of matrix factorization theory, the modeling methods based on the multi-irrelevant-models form began to emerge [3][9-10]. However, the models were generally only based on user. Lack of product perspective, user preference simulation is not comprehensive, which affects recommended performance negatively.

In response to above issues, this paper presents a hybrid collaborative filtering approach based on product attributes – PACF (Product Attributes Collaborative Filtering). To model from two angles of user and product, PAM (Product Attributes Model) based on the matrix factorization's vectors multiplication idea is discussed. After that, important elements of *Product Attribute Weight* and *Product Attribute Score* for the PAM are defined for users and products respectively. It needs applicable formula to construct new model to integrate these factors. Then, a new hybrid collaborative filtering formula  $\Gamma_{PAM}$  is proposed to generate the recommended results for the PAM.

The rest of this paper is organized as follows. Section II reviews related work. Section III shows the formal definition. Section IV introduces the model PAM and the formula  $\Gamma_{PAM}$  for PACF. Section V discusses the experimental analysis, and Section VI presents conclusions and provides future research directions.

## II. FORMAL DEFINITION

Integrating the valuable information embedded in reviews not only promotes user experience in the recommender system but also improves the recommended performance [2]. A virtual rating can be generated through users' implicit preference information from reviews.

Aimed at characteristics of product reviews, product attributes are fixed to facilitate feature consistency first. Then, sentiment polarity is introduced to achieve accurate user preferences. Product attributes can be defined including quality, performance, appearance and other aspects. Positive polarity and negative polarity are embodied in sentiment polarity. The

specific symbols are clearly defined in Table I. The product attributes parameter is fixed through building the dictionary, shown in Table II.

TABLE I. SYMBOL AND ITS MEANING

Symbol	Meaning
$R=\{r_1, r_2, \dots, r_{ R }\}$	Reviews set
$U=\{u_1, u_2, \dots, u_{ U }\}$	Users set
$P=\{p_1, p_2, \dots, p_{ P }\}$	Products set
$PA=\{pa_1, pa_2, \dots, p_{ PA }\}$	Product attributes set. The specific definition is shown in Table II.
$F_k = \{f_1, f_2, \dots, f_{ F_k }\}$	$F_k$ is a set of feature words $f_i$ of the product attribute $pa_k$ . The specific definition shown in Table II.
$\delta_i \in \{1, -1\}$	$\delta_i$ is the sentiment polarity corresponding to the product attribute feature word $f_i$ . Among the set, -1 is negative sentiment, and 1 is non-negative sentiment (including positive and neutral).

TABLE II. PRODUCT ATTRIBUTES AND FEATURE WORDS

Symbol	Meaning
$PA$	$PA=\{Quality, Service, Performance, Package\}$
$F_{Quality}$	$F_{Quality}=\{nature, product, greener, brand .etc\}$
$F_{Service}$	$F_{Service}=\{communication, efficient, responsive .etc\}$
$F_{Performance}$	$F_{Performance}=\{fresh, flavor, awful, tasted .etc\}$
$F_{Package}$	$F_{Package}=\{delivery, ship, on-time, speed .etc\}$

Accordingly, the research problem is to tackle the following challenges. First, how to reliably model inference user preferences from two-tuples  $(pa_k, \delta_i)$ ? Second, how to effectively incorporate product attributes information to generate recommendation results? Another problem is to establish the relevance of users' model and products' model.

### III. PRODUCT ATTRIBUTES COLLABORATIVE FILTERING

This section presents the recommended framework of PACF shown in Fig 1. After data preprocessing, a collector of two-tuples  $(pa_k, \delta)$  in reviews is obtained.

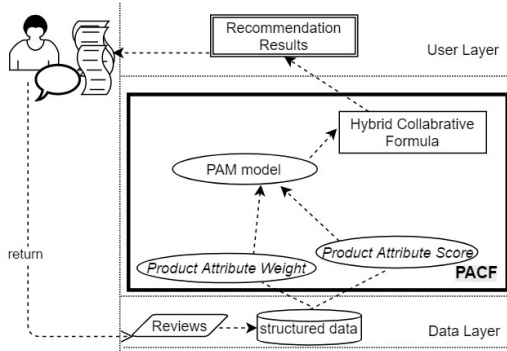


Fig. 1. The recommendation framework of PACF

Then, a novel recommendation method is used to generate recommendations. We formalize PACF with the PAM model based on reviews and a hybrid collaborative filtering formula  $\Gamma_{PAM}$ . *Product Attribute Weight* proposed in PAM characterizes the user's weight while *Product Attribute Score* describes the product's score. The specific modeling is detailed in Section A. Learning the idea of a hybrid collaborative filtering approach, Section B proposes a new formula  $\Gamma_{PAM}$  for the PAM. The premise of the formula is to make *Product Attribute Weight* relate to *Product Attribute Score*.

Finally, the recommendation results are generated through the formula.

#### A. Product Attribute Model based on Reviews

Users and products are modeled separately through the matrix factorization idea [9]. From the perspective of the user's weight and the product's score, PAM is subdivided into *Product Attribute Weight* and *Product Attribute Score*.

##### 1) Product Attribute Weight Analysis

The first measure, *Product Attribute Weight* recorded as  $W$ , is the degree of attention given to the attributes of the product. Given a user  $u_i$  and a product attribute  $pa_k$ , the formula *Product Attribute Weight* is defined as follows.

$$W(u_i, pa_k) = \frac{|F_{ik}|}{|F_i|} = \frac{|\delta_{ik}|}{|\delta_i|} = \frac{\sum_{p_j \in R_i} |\delta_{ijk}|}{\sum_{p_j \in R_i} |\delta_{ij}|} \quad (1)$$

In formula (1),  $R_i$  is the set of  $u_i$ 's reviews;  $p_j \in R_i$  represents the products which the user reviewed;  $|F_{ik}|$  represents the frequency that the feature word is mentioned by the user with respect to product attribute  $pa_k$ ;  $|F_i|$  represents the number of times that all product attributes' feature word is mentioned by the user;  $\delta$  indicates the sentiment polarity value.  $|\delta_{ijk}|$  is the user's sentiment polarity set of product attribute  $pa_k$  for product  $p_j \in R_i$ . The number of user comments on product attribute  $pa_k$  is  $|\delta_{ik}|$ .  $|\delta_{ij}|$  represents the user's sentiment polarity set for  $p_j \in R_i$  on all product attributes  $PA$ .  $|\delta_i|$  is the frequency of all product attributes  $PA$  in the comments. When the value of  $W$  is zero or unknown, *Product Attribute Weight* is 0.1.

##### 2) Product Attribute Score Analysis

The second measure, *Product Attribute Score*, is similar to the user rating the project. The user's views of product attributes are scored and recorded as  $S$ . Given a product  $p_j$  and a product attribute  $pa_k$ , the measure *Product Attribute Score* can be defined as follows.

$$S(p_j, pa_k) = \frac{\sum_{u_i \in R_j, \delta_{ijk}=1} |\delta_{ijk}|}{\sum_{u_i \in R_j} |\delta_{ijk}|} = \frac{\sum_{u_i \in R_j, \delta_{ijk}=1} |\delta_{ijk}|}{|\delta_{jk}|} \quad (2)$$

Among the formula (2),  $R_j$  is the reviews set for the product  $p_j$ ;  $u_i$  is the user who has commented on it; and  $\delta$  indicates the sentiment polarity value.  $\delta_{ijk}$  is the user's sentiment polarity set of product attribute  $pa_k$ .  $\sum_{u_i \in R_j, \delta_{ijk}=1} |\delta_{ijk}|$  expresses the size of

the sentiment polarity set when  $\delta_{jk}=1$ . When the value of  $S$  is zero or unknown, *Product Attribute Score* is 0.1.

### 3) Product Attributes Model

Draw on the experience of the multiplication form of matrix factorization, the PAM model is divided into different models corresponding to users and products. With the two formulas proposed above, PAM can be defined:

$$PAM(u_i, p_j, PA) = \begin{cases} (W_1, W_2, \dots, W_{|PA|}) & W_k = W(u_i, pa_k) \text{ for users} \\ (S_1, S_2, \dots, S_{|PA|}) & S_k = S(p_j, pa_k) \text{ for products} \end{cases} \quad (3)$$

### B. Hybrid Collaborative Filtering Formula for PAM: $\Gamma_{PAM}$

The vectors corresponding to users and products separately in the PAM are unrelated. In order to solve the problem, the average *Product Attribute Score* of users recorded as  $\bar{S}$  is introduced based on the shopping history in reviews. The average *Product Attribute Weight* of the product is calculated similarly and denoted as  $\bar{W}$ . The formula is shown as follows.

$$PAM'(u_i, p_j, PA) = \begin{cases} (W_1, W_2, \dots, W_{|PA|}) \xrightarrow{\text{reviews}} (\bar{S}_1, \bar{S}_2, \dots, \bar{S}_{|PA|}) & \text{for users} \\ (S_1, S_2, \dots, S_{|PA|}) \xrightarrow{\text{reviews}} (\bar{W}_1, \bar{W}_2, \dots, \bar{W}_{|PA|}) & \text{for products} \end{cases} \quad (4)$$

Next, the generalization formula (5) we proposed is extracted for calculating the hybrid collaborative filtering [1][2][11-13]. A new hybrid collaborative filtering formula (6) for the PAM is obtained by derivation the formula (4) through combining the formula (5) and the cosine formula.

$$\Gamma_{HCF} = UBCF \times IBCF \quad (5)$$

$$\Gamma_{PAM}(u_i, p_j) = \frac{\sum_1^{|PA|} (W_i \times \bar{W}_j)}{\sqrt{\sum_1^{|PA|} W_i} \times \sqrt{\sum_1^{|PA|} \bar{W}_j}} \times \frac{\sum_1^{|PA|} (\bar{S}_i \times S_j)}{\sqrt{\sum_1^{|PA|} \bar{S}_i} \times \sqrt{\sum_1^{|PA|} S_j}} \quad (6)$$

For the current user, the cosine  $\cos_{UBCF}$  is obtained by the vector  $(W_1, W_2, \dots, W_{|PA|})$  and the average user weight value  $\bar{W} = (\bar{W}_1, \bar{W}_2, \dots, \bar{W}_{|PA|})$ . It is the same way to figure out the value for  $\cos_{IBCF}$ .

## IV. EXPERIMENTS

To evaluate the performance of PACF, experiments are carried out and compared by analyzing our method against other algorithms through offline dataset.

### A. Dataset

We used the Amazon fine-food reviews dataset from SNAP. The dataset collected 568,454 reviews posted by 256,059 users for 74,258 items of food [14]. The format contains the *UserId*, the *ProductId*, the *Text*, and the *Score* attributes, which are

required for the experiment. The *Score* is an integer from 1 to 5. In large shopping sites, the number of users and products increases daily. Meanwhile, the dataset of actual purchases is sparse, usually below 0.1%. The datasets are described in Table III.

TABLE III. DESCRIPTIVE STATISTICS OF DATASETS

Dataset	Descriptive Statistics			
	Num. of users	Num. of products	Num. of ratings and reviews	Sparsity
Data1	1232	754	1250	0.1346%
Data2	2420	1193	2500	0.0866%
Data3	4719	1791	5000	0.0592%
Data4	9051	1765	10000	0.0626%
Data5	17139	3148	20000	0.0371%

### B. Experimental Result and Evaluation

The rating data have a sparseness of less than 0.1% on shopping sites. Furthermore, the product category is in billions of units. Thus, using TOP-N sorting to evaluate accuracy is not an appropriate method. Our experiment is to simulate the data from actual shopping sites. The extremely low accuracy of this site does not have a value.

In this case, *coverage* is more appropriate. The *coverage* is used to measure the ability of the methods to discover products. The PACF method contains the idea of collaborative filtering and matrix factorization. Therefore, the following comparison is considered: UBCF, IBCF and SVD [15]. UBCF and IBCF are classical algorithms for collaborative filtering; SVD is a representative algorithm for matrix factorization. Firstly, we take the datasets in Table III as the experimental dataset. Then, the evaluation metric *coverage* is calculated under  $N = 1, 5, 10$  and 20. The experimental results are shown in Table IV. The *coverage* unit is %.

TABLE IV. EXPERIMENTAL RESULTS OF COVERAGE

Dataset	Methods	Coverage			
		N=1	N=5	N=10	N=20
Data1	UBCF	0.1326	0.0119	0.0146	0.0291
	IBCF	0.2652	0.0093	0.0172	0.0305
	SVD	0.6631	0.0186	0.0358	0.0650
	PACF	1.3263	0.0597	0.0941	0.1552
Data2	UBCF	0.0008	0.0042	0.0092	0.0176
	IBCF	0.0017	0.0050	0.0101	0.0192
	SVD	0.0117	0.0268	0.0360	0.0762
	PACF	0.0142	0.0386	0.0695	0.1215
Data3	UBCF	0.0006	0.0036	0.0430	0.0122
	IBCF	0.0017	0.0061	0.0089	0.0151
	SVD	0.0168	0.0329	0.0061	0.0642
	PACF	0.0101	0.0274	0.0519	0.0966
Data4	UBCF	0.0017	0.0045	0.0073	0.0130
	IBCF	0.0017	0.0062	0.0102	0.0164
	SVD	0.0221	0.0567	0.0771	0.1082



Dataset	Coverage				
	Methods	N=1	N=5	N=10	N=20
Data5	PACF	0.0096	0.0306	0.0515	0.0816
	UBCF	0.0004	0.0016	0.0030	0.0048
	IBCF	0.0006	0.0022	0.0042	0.0076
	SVD	0.0072	0.0198	0.0358	0.0550
	PACF	0.0056	0.0150	0.0260	0.0398

### C. Discussion

The overall *coverage* of PACF is on the rise in Table IV. PACF depends on the number of reviews. The larger the reviews, the better the performance. PACF also applies to scenarios where SVD is suitable. In Data1, Data2 and Data3 of Table IV, PACF performs better than SVD. In Data4 and Data5 of Table IV, PACF's *coverage* is lower than SVD's. In further analysis, PACF's formula is similarly affected by the purchase record. With a sparsity of 0.05%, PACF sacrifices coverage. Moreover, Table IV illustrates that when sparseness is higher than 0.05%, the *coverage* performance of our proposed PACF is good.

### V. CONCLUSION AND FUTURE WORK

Considering the characteristics of product reviews, this paper uses constant product attributes and establish irrelevant and multi-perspective models. The objective is to solve the problem of complicated product reviews but also to refine information on user preferences. Based on the above conditions, a hybrid collaborative filtering method PACF is proposed. The PAM model and the  $\Gamma_{PAM}$  formula are constituted to make PACF. PAM consists of *Product Attribute Weight* and *Product Attribute Score*. The perspectives of users and products can effectively simulate user preferences. The experiments have showed that PACF achieved better recommended performance in dealing with large and sparse reviews and predicted user behavior well. The coverage is superior to other methods at a sparsity higher than 0.05%.

In the future, we will study the current issue, PACF sacrifices coverage when sparsity is less than 0.05%. Using user relation and social information from other platforms can further improve the performance of the recommendation on sparse data. Moreover, the implementation and effective response of large-scale electronic website platform is worthy of further work. To this point, cloud computing and cluster will be considered to accelerate the reaction speed.

### ACKNOWLEDGMENT

The authors thank the anonymous reviewers for their valuable and constructive comments. This work is supported by The National Key Research and Development Plan of China under Grant No. 2017YFD0400101, and The National Natural Science Foundation of China under Grant No. 61502294, 61572306.

### REFERENCES

- [1] Lu J., Wu D. S., Mao M. S., Wang W. and Zhang G. Q., "Recommender system application developments: A survey", *Decision Support Systems*, 2015, 74, pp.12-32.
- [2] Chen L., Chen G. L. and Wang F., "Recommender systems based on user reviews: the state of the art", *User Modeling And User-adapted Interaction*, 2015, 25(2), pp. 99-154.
- [3] Ma Y., Chen G. Q. and Wei Q., "Finding users preferences from large-scale online reviews for personalized recommendation", *Electronic Commerce Research*, 2017, 17(1), pp. 3-29.
- [4] T. Sangeetha, N. Balaganesh and K. Muneeswaran, "Aspects based opinion mining from online reviews for product recommendation", *ICCIDS*, 2017, DOI: 10.1109/ICCIDS.2017.8272657.
- [5] M. Vamsee Krishna Kiran, RE Vinodhini, R. Archanaa and K. Vimalkumar, "User specific product recommendation and rating system by performing sentiment analysis on product reviews", *ICACCS*, 2017, DOI: 10.1109/ICACCS.2017.8014640.
- [6] ZKA Baizal, A. Iskandar and E. Nasution, "Ontology-based recommendation involving consumer product reviews", *ICoICT*, 2016, DOI: 10.1109/ICoICT.2016.7571890.
- [7] Alton Y.K. and C.S. Banerjee, "Helpfulness of user-generated reviews as a function of review sentiment, product type and information quality", *Computers in Human Behavior*, 2016, 54, pp.547-554.
- [8] Su J.K., E. Maslowska and E. C. Malthouse, "Understanding the effects of different review features on purchase probability", *International Journal of Advertising*, 2018, 37, Issue 1: Electronic Word-of-Mouth
- [9] Zhao W. X., Wang J. P., He Y. L. and et al, "Mining Product Adopter Information from Online Reviews for Improving Product Recommendation", *ACM Transactions on Knowledge Discovery from Data*, 2016, 10(3), article num.29.
- [10] Z.W. Yu, H. Xu, Z. Yang and B. Guo, "Personalized Travel Package with Multi-Point-of-Interest Recommendation based on Crowdsourced User Footprints", *IEEE Transactions on Human-Machine Systems*, 2016, 46(1), pp.151-158.
- [11] Koren Y., Bell R. and Volinsky C., "Matrix Factorization Techniques for Recommender systems", *Computer*, 2009, 42(8), pp. 30-37.
- [12] Hammou B. A. and Lahcen A. A., "FRAIPA: A fast recommendation approach with improved prediction accuracy", *Expert Systems with Applications*, 2017, 87, pp. 90-97.
- [13] Kassak O., Kompan M. and Bielikova M., "Personalized hybrid recommendation for group of users: Top-N multimedia recommender", *Information Processing & Management*, 2016, 52(3), pp. 459-477.
- [14] [Online]. Available: <http://online.cambridgecoding.com/notebooks/eWReNYcAfB/implementing-your-own-recommender-systems-in-python-2>
- [15] [Online]. Available: <http://snap.stanford.edu/data/web-Amazon.htm>

# Exploiting SDAE Model for Recommendations

Qing Yang, Xianhe Yao

*Guangxi Key Laboratory of Automatic Detecting Technology and Instruments  
Guilin University of Electronic Technology, Guilin, China  
gtyqing@hotmail.com, 407953822@qq.com*

Jingwei Zhang

*Guangxi Key Laboratory of Trusted Software  
Guilin University of Electronic Technology  
Guilin, China  
gtzjw@hotmail.com*

Zhongqin Bi

*College of Computer Science and Technology  
Shanghai University of Electronic Power  
Shanghai, China  
zqbi@shiep.edu.cn*

**Abstract**—The data for recommendations, usually a matrix composed of users and items, include a large number of missing data, noise data, etc, which have a negative effect on the accuracy of recommendations. In order to improve recommendation performance, this paper put forwards an improved model named Stacked Denoising AutoEncoder (SDAE), which improves the autoencoder by both indicators and denoising parts to construct an effective stacked autoencoding network. The first layer of the encoding network is responsible for dealing with missing data with the help of indicators and to get a new encoding for features, and then stacked denoising is applied to process noise data for a further optimization. SDAE's output can be accepted by collaborative filtering methods to provide a more accurate recommendation. Three data sets are used to verify the proposed model, the experimental results show that the proposed model presents an active ability on improving recommendation performance and mitigates the negative influence caused by missing data, noise data, etc.

**Index Terms**—recommendation, data preprocessing, s-tacked encoding

## I. INTRODUCTION

Data play a key role for recommendations to match users' interests, such as smart tourism, online shopping. But missing data, noise data etc are often included when collecting data from applications. These inaccurate data bring negative impact on the effectiveness of recommendations. For example, as a popular strategy for recommendation, collaborative filtering will be weakened on computing the similarity between users or items when using data containing both missing values and noise data.

In addition, it is also difficult to capture users' characteristics accurately when facing large amount of missing values and noise data. For example, 0 is usually a default value for those unobserved behaviors, but may also be a score from a real user, the same value meaning different

actions for users will yield unsatisfied recommendations. Obviously, it is also possible to output unexpected decision on users' similarity caused by noise data, which can even result in a series of failures on recommendation for an initializer. Aiming at the above challenges, this paper focuses on data preprocessing models to improve data quality for recommendations. Depending on deep learning and encoding technologies, this paper put forwards a stacked denoising autoencoder to preprocess missing data and noise data in recommendation data, which can then cooperate with collaborative filtering to improve recommendation accuracy. The major contributions are as followings,

- introducing indicators to deal with missing data and to compute the hidden features for recommendations;
- constructing a stacked denoising autoencoder to process noise data and to cooperate with collaborative filtering for improving recommendation performance;
- conducting comprehensive experiments to verify the effectiveness of the proposed model.

This paper is organized as following. Section II summarized the related work on recommendation and data preprocessing technologies. Section III presented the primary objective of this paper and the problem statement. Section IV detailed the proposed model, including the network structures for encoding and parameter training. Section V designed experiments to provide proof for verifying and analyzing our model. Section VI concluded the whole paper and discussed the future work.

## II. RELATED WORK

Data preprocessing is a key part to ensure the effectiveness of recommendations. Considering the negative effect on recommendations brought by missing values, these current popular strategies are to fill missing values, dimensionality reduction or clustering. [1] proposed a model-based collaborative filtering method that used a specific

Jingwei Zhang and Zhongqin Bi are corresponding authors.  
DOI reference number: 10.18293/SEKE2018-094

value to substitute those missing values and to compute the similarity between items. [2] put forwards a two-stage clustering method on sparse data, which combined graph summarization with content-based similarity to provide topic recommendation based on users' interests.

Aiming at suppressing noise data, principal component analysis(PCA) is a classical strategy on both denoising and dimensionality reduction. On denoising applications, PCA belongs to fixed effect model and uses a fixed structure with low noise to generate data. [3] analyzed the difference between PCA and regularized PCA and proved that the regularized PCA performs better even facing high-noise data. In addition, neural networks are becoming active for processing noise data. [4] applied radial basis function neural networks on denoising of ECG signal and improved analysis performance on ECG data.

In 2006, both deep learning and the improvement on model training broke the bottleneck of traditional BP neural networks, Geoffrey Hinton elaborated systematically the strong learning ability on features contributed by multi-layer neural networks and proved that those features learned by deep learning model can express those initial data better [5]. [6] applied local denoising to learn the effective features in deep networks. Aiming at the local minimum of gradient descent caused by stochastic weight initialization of neural networks, [7] put forwards stacked autoencoder on stacked RBM and trained neural networks layer by layer in greedy mode to capture the deep features, which can learn deep nonlinear network structures from large-scale data. This paper made a comprehensive consideration on both current data preprocessing for recommendation applications and the popularity of deep learning, and constructed an improved stacked denoising autoencoder model to provide more effective and accurate input data for further recommendations.

### III. PROBLEM STATEMENT

This paper focuses on more effective data preprocessing and feature capturing method, which can cooperate with those popular recommendation methods, such as collaborative filtering, top- $N$  recommendations, etc, for better recommendations. The specific problem can be stated as,

Given an initial dataset  $D$ , we need to design a function  $df$  for improving their effectiveness on recommendations,  $D' = df(D)$ , which should satisfy  $ACC(rf(D')) > ACC(rf(D))$  when given a specific recommendation function  $rf$  and a concrete evaluation metric  $ACC$ .

Our basic strategy for the function  $df$  is to introduce autoencoders and neural networks for dealing with two kinds of abnormal data in the initial dataset, namely missing data and noise data. The following parts will detail our solutions.

### IV. SDAE MODEL

A Stacked Denoising AutoEncoder(SDAE) model is proposed to optimize data for recommendations. SDAE model introduces an autoencoder with indicators to deal with the data sparsity and to construct their initial encoding. Then, a Gaussian noise is applied on the output encoding and the stacked denoising technology is combined with autoencoding to suppress the noise. The proposed model reduces the negative effect caused by both the sparsity and noise data to improve recommendation quality.

#### A. Sparsity Optimization by Introducing Autoencoding with Indicators

Autoencoder adopts unsupervised learning model to train on non-labeled dataset, which constructs a three-layer network model to draw the compression characteristics of the input data. The encoding network structure is illustrated in Fig. 1. For encoding, the input  $x$  will be converted into a hidden feature  $h$ , which is computed as Equation. 1.  $W_1$  is a weight matrix,  $b_1$  is a bias vector.  $\sigma(x) = \frac{1}{1+e^{-x}}$  is a sigmoid function [8]. For decoding, the weight matrix  $W_2$  and the bias vector  $b_2$  are applied to restore the implied feature  $y$  from the compressed hidden layer  $h$ , which is showed in Equation. 2. We introduce stochastic gradient descent strategy to optimize the weight matrix  $W_1, W_2$  and the bias vector  $b_1, b_2$ , and use minimum average error to measure the encoding performance. The average error between the input  $x$  and the encoding output  $y$  is defined as their  $L2$  norm, which is presented in Equation. 3. The value  $h$  outputted by the hidden layer is just the encoding output of the initial data in the first phase.  $\| \cdot \|_2$  represents the  $L2$  norm of vectors or matrix.

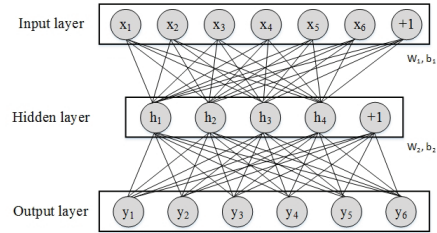


Fig. 1. A three-layer autoencoder

$$h = \sigma(W_1^T x + b_1) \quad (1)$$

$$y = \sigma(W_2^T h + b_2) \quad (2)$$

$$\iota(x, y) = \| x - y \|_2 \quad (3)$$

A large number of missing data are acting as negative samples when recommendation algorithms work on them [9]. It is difficult for a pure autoencoder to capture the

data feature accurately. The following will construct an autoencoding network with indicators to filter out missing data and to reduce their side effect on similarity computation. We define an indicator matrix as Formula. 4 to identify those missing data in the input data, where  $i, j$  represents the user ID and the ID of recommended objects. The indicator matrix can help to remove those missing data from the training process of autoencoders, whose structure is illustrated in Figure. 2.

$$indicator_{i,j} = \begin{cases} 1 & \text{if } data_{i,j} \neq 0 \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

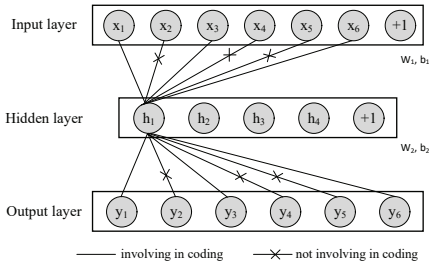


Fig. 2. A partial illustration of autoencoder with indicators

The specific process for training autoencoding network is detailed in Algorithm. 1, which is composed of two stages with the help of indicators, encoding and decoding. Those related symbols are as following.  $X$  is the initial input data,  $n_v$  is the node number in the input/visual layer,  $n_h$  is the node number in the hidden layer.  $l_r$  denotes the learning rate,  $\epsilon$  holds the threshold of the minimum average error, which respectively serve for the network constructing speed and the stopping condition. The weight matrix are denoted as  $W$  and  $W'$ , their corresponding bias vectors are  $b$  and  $b'$ . For functions, the encoding function is denoted by *get-hidden-units*, the decoding function is *get-reconstruction-units*, who correspond to the Formula. 1 and 2. The function for computing reconstruction error is *get-cost* that is expressed in 3. *get-gradient* is used to compute gradients, which cooperates with  $l_r$  and applies stochastic gradient descent [10] to update parameters and to minimize the reconstructed errors. By the indicators, the weights corresponding to the missing parts are ignored at both input and output, and an activation function is designed to compute both the hidden and the reconstructed representations for the initial data.

### B. Improving Recommendations by Stacked Denoising AutoEncoder(SDAE)

Noise is another factor disturbing the accuracy of recommendations. Based on the autoencoder model with indicators, a denoising autoencoder model is proposed to remove the disturbance caused by noise. In order to improve the local minimizing problem of gradient descent,

---

### Algorithm 1 Autoencoding with Indicators

---

**Input:** the initial data  $X$ ,

the indicator matrix *indicator*,

the node number in the input/visible layer  $n_v$ ,

the node number in the hidden layer  $n_h$ ,

the learning rate  $l_r$ ,

the threshold for minimum average error  $\epsilon$ ,

the initial weight matrix  $W$  and  $W'$ ,

the initial bias vectors  $b$  and  $b'$

**Output:** the encoding output  $Y_{out}$

1:  $X' = X * indicator$ ;

2: while  $cost > \epsilon$  do

3:  $Y = get - hidden - units(X', W, b, n_v, n_h)$ ;

4:  $Z = get - reconstruction - units(Y, W', b', n_h, n_v)$ ;

5:  $Z_{out} = Z * indicator$ ;

6:  $cost = get - cost(X', Z_{out})$ ;

7:  $gradient = get - gradient(cost, param)$ ;  
 $\quad \quad \quad \backslash \backslash param$  is  $(W, W', b, b')$

8:  $param = param - l_r * gradient$ ;

9: end while

10: **return**  $Y_{out} = Z$ ;

---

denoising autoencoders(DAE) are stacked together to form a new stacked denoising autoencoder (SDAE), an iteration computation is applied to capture the features of the initial input. The bottom-up and layered training strategy is used to decide the parameters of the whole network. In the phase of data preprocessing, a group of noise are firstly merged into the initial data to maintain the consistency between the initial data and the features learned from the merged data. This measure can help to improve the ability on resisting noise for recommendations.

Considering that most of noise in real life conform to Gaussian distribution [11], we use Gaussian noise to simulate those possible noise in the initial collected data, which are formalized as  $x' \sim q_D(x'|x)$ . Here,  $x$  is the initial collected data,  $x'$  is the data merged with noise and is also the input of recommendation applications,  $D$  represents the whole dataset. The structure of the denoising autoencoder is presented in Fig. 3. The Denoising AutoEncoder model (DAE) is responsible for minimizing the reconstructed error between the input merged with noise and its output, which is presented in Formula. 5. Multiple DAE are stacked by submitting the output from the previous DAE to the input of the next DAE to improve recommendation performance.

$$j(x, y) = (\|x' - y\|)_2 \quad (5)$$

Algorithm. 2 presents the detailed process to capture the features of the original data. Compared with Algorithm. 1, a specific value for the layers of neural networks,  $n_l$ , is

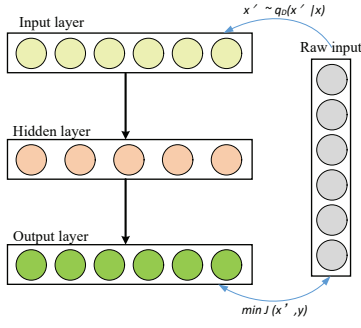


Fig. 3. The model for denoising autoencoder.

introduced to represent how many times for DAE model to be stack. In addition, an array holding the node numbers for each hidden layer,  $n_H[n_l]$ , and the signal-noise ratio  $c_r$  are designated. *get - input* is a new function for mixing Gaussian noise.

---

**Algorithm 2** *Stacked Denoising Autoencoding for Recommendations*

---

**Input:** the initial data  $X$ ,

- the layers of neural networks  $n_l$ ,
- the node number in the input/visible layer  $n_v$ ,
- the node number in the hidden layer  $n_h[n_l]$ ,
- the signal-noise ratio  $c_r$ ,
- the learning rate  $l_r$ ,
- the threshold for minimum average error  $\epsilon$ ,
- the initial weight matrix  $W$  and  $W'$ ,
- the initial bias vectors  $b$  and  $b'$

**Output:** the encoding output  $Y_{out}$

- 1: for  $k$  from 1 to  $n_l$  do
  - 2:  $X' = \text{get - input}(X, c_r)$
  - 3: while  $cost > \epsilon$  do
  - 4:  $Y = \text{get - hidden - units}(X', W[i], b[i], n_v, n_h[k]);$
  - 5:  $Z = \text{get - reconstruction - units}(Y, W'[i], b'[i], n_h[k], n_v);$
  - 6:  $cost = \text{get - cost}(X', Z);$
  - 7:  $gradient = \text{get - gradient}(cost, param);$   
 $\backslash\backslash param$  is  $(W[i], W'[i], b[i], b'[i])$
  - 8:  $param = param - l_r * gradient;$
  - 9: end while
  - 10: end for
  - 11: **return**  $Y_{out} = Z;$
- 

## V. EXPERIMENTS AND EVALUATION

### A. Experimental Setup and Datasets

The proposed method is tested and analyzed in this section. The concrete experimental configurations are composed of MacOS Sierra and Matlab2015b run on Intel core i7 with 2.2GHz and 16GB DDR3 memory. Three

types of datasets covering different scenarios are applied to verify the effectiveness of the proposed method, which are listed in Table. I. The dataset from *GroupLens* uses 0 to express all missing scorings, in which you can not make a clear distinction between a real score and a default value when meeting 0. Though 0 is still used as default value, the *EachMovie* dataset accompanies all 0s with a weight, which can help you to distinguish that this value is from a real user or is a default one. The dataset from *Jester* is about online joke recommendation, which presents different characteristics from the above two, and consists of less items for recommendations and a larger range for user scoring, and the default values are also not overlapped with the scoring range.

TABLE I  
THE DETAILS OF DATA SETS

Name	Num. of users	Num. of items	Scoring range	Default values
MovieLens-100k	943	1682	[0, 5]	0
EachMovie	61265	1623	[0, 1]	0 (with weight)
Jester Dataset1	24983	100	[-10, 10]	99

Since collaborative filtering is the most popular method for recommendations, we will execute collaborative filtering [12] on the preprocessed data contributed by our proposed method to verify its effectiveness. *Top - N* recommendation is used to evaluate the improvement on recommendations. Three evaluation metrics, *Precision*, *Recall*, and *F1-Score*, are used to evaluate our proposed method, which are computed as Formula. 6- 8. Here, *true positive* is the number of items that should be recommended and have also been in the top- $N$  list, *false positive* represents the number of items that should not be recommended but have been in the top- $N$  list, and *false negative* corresponds to the number of items that should be recommended but have not been in the top- $N$  list.

$$Precision = \frac{truepositive}{truepositive + falsepositive} \quad (6)$$

$$recall = \frac{truepositive}{truepositive + falsenegative} \quad (7)$$

$$F1 - Score = \frac{2 * precision * recall}{precision + recall} \quad (8)$$

In order to verify the improvement on the recommendation accuracy brought by the different compression dimensions in SDAE model, we introduce the user-based collaborative filtering, autoencoding with indicators, as well as a SAE model to compare with the SDAE model on recommendation performance. The specific SAE model

has both the same parameters and the same structure with the SDAE model, whose input dose not mix with noise. The comparison between the SAE model and the SDAE model can show the recommendation effectiveness contributed by noise suppression.

### B. Experiments on MovieLens-100k

In this part, we test the effectiveness of our proposed method on a traditional dataset, *MovieLens-100k*, which consists of 100 thousands of scores on 1,682 movies contributed by 943 users. Each score is between 0 and 5, a hidden challenge is that you can not tell whether the value is written by a real user or is just a default value when it is 0. First, we designed a group of experiments to test the recommendation performance under different proportions of noise, the experimental results are presented in Fig. 4. Different amount of noise present different effect on the recommendation accuracy. In the following experiments, 13% noise will be introduced into the datasets since it contributed the best performance on recommendations.

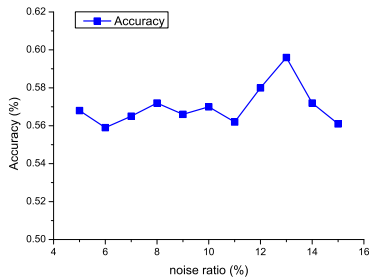


Fig. 4. Recommendation performance under different noise.

Second, we designed a group of experiments to observe the top- $N$  recommendation performance on *MovieLens-100k*, here  $N$  is designated to be 10. Fig. 5 presents the recommendation performance contributed by four methods, which are user-based collaborative filtering (User-based), autoencoder with indicators (AE-indicator), stacked autoencoder (SAE) and stacked denoising autoencoder (SDAE). Here, indicators are used to remove all values equal to 0 and do not consider whether this value is from a real user or is just a default value. Compared to the user-based collaborative filtering without data preprocessing, AE-indicator, SAE, SDAE all contributed a better performance on precision, which proved that both the indicators and the stacked encoding technologies have made a difference on the original dataset. Especially, SDAE presented the best precision among the four models. Considering recall, user-based collaborative filtering performed better than AE-indicator, SAE and SDAE. The above situation can be attributed to two factors, one is that the indicators removed some values from real users, and the other is due to the sparse data, the whole data sparsity is  $S_{MovieLens-100k} =$

$\frac{100000}{1682*943} = 0.063$ , which caused the introduction of noise to have a more obvious influence on a sparse dataset.

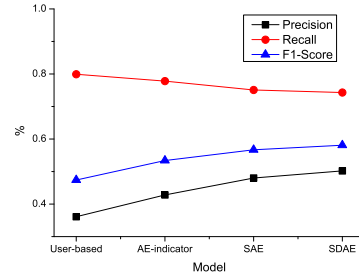


Fig. 5. Top- $N$  recommendation performance on *MovieLens-100k*.

### C. Experiments on EachMovie

*EachMovie* is a dataset collected by HP/Compaq, which consists of 2,811,983 scores on 1,623 movies contributed by 61,265 users. Except for more users than *MovieLens-100k*, the scoring weights are introduced to distinguish those default values from values contributed by real users. Fig. 6 presents the effects of scoring weights on improving recommendation accuracy, both user-based collaborative filtering and AE-indicators have a little improvement on accuracy when those real values with an appointed scoring weight are applied. The following experiments will associate indicators with scoring weights to reserve those values contributed by real users.

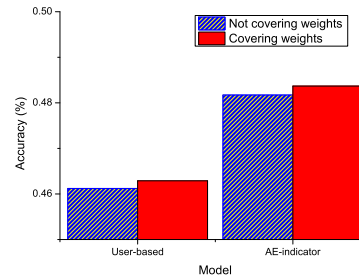


Fig. 6. The recommendation effectiveness test of scoring weight on *EachMovie*.

In this group of experiments, we chose 30,000 users and their scorings randomly to be the training data and then used the remaining data to test their recommendation performance, the experimental results are presented in Fig. 7. SDAE still contributed the best precision, which also declared that indicators, stacked technologies and denoising worked well on preprocessing data. But all the four models presented a lower recall than those on *MovieLens-100k* since the dataset is a more sparse one, whose sparsity is  $S_{EachMovie} = \frac{2811983}{1623*61265} = 0.028$ .

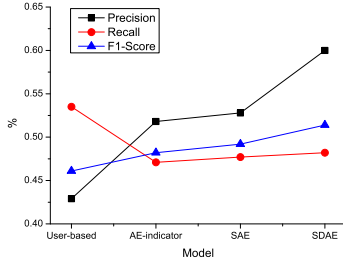


Fig. 7. Top- $N$  recommendation performance on *EachMovie*.

#### D. Experiments on Jester

In this part, we introduced *Jester* dataset to test the effectiveness of the proposed method when facing a small number of features since the node number of the hidden layer are often less than those of the input layer or the output layer in an autoencoder. *Jester* dataset includes scores on 100 jokes contributed by 24,983 users, each score is from -10 to 10 and all default values are expressed as 99. We only consider those values greater than 0 as the positive examples. The experimental results are presented in Fig. 8. Compared with the results on *MovieLens-100k* and *EachMovie*, all the four models presented a good accuracy though SDAE has a little advantage, which show that the data sparsity is a key factor for recommendation performance. One obvious change is that the recall contributed by SDAE covered the other three models, in fact, the effects contributed by indicators can be ignored since a default value is expressed as 99, the denoising and stacked technologies played a major role on improving recommendation performance for SDAE.

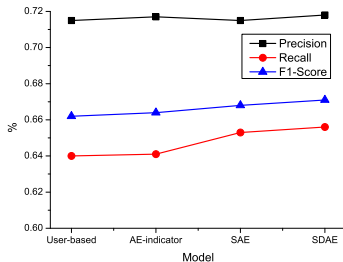


Fig. 8. Top- $N$  recommendation performance on *Jester*.

## VI. CONCLUSION

This paper provides an optimized data preprocessing model, SDAE, to provide more robust data for improving recommendation performance. SDAE synthesized the indicators, stacked encoding and denoising technologies to improve data quality. An autoencoder with indicators are firstly introduced to remove default values and to reduce the loss on recommendation accuracy brought by sparse data. Then a stacked denoising autoencoder is designed

to relieve recommendation disturbance caused by noise data, which provides an iterative computation on a three-layer neural network. We verified the effectiveness of the proposed model on different scenarios, including sparse datasets and small number of recommendation objectives. In future, we will apply neural networks to capture the user interest migration and to provide a dynamic recommendation model, the time cost on data preprocessing will also be covered.

*Acknowledgments:* This work is supported by the National Natural Science Foundation of China (No.61462017, 61363005, 61662015, U1501252, U1711263), Guangxi Natural Science Foundation of China(No.2017GXNSFAA198035), and Guangxi Cooperative Innovation Center of Cloud Computing and Big Data.

## REFERENCES

- [1] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th International Conference on World Wide Web*, ser. WWW '01. New York, NY, USA: ACM, 2001, pp. 285–295.
- [2] K. Chen, P. Han, and J. Wu, "User clustering based social network recommendation," *Chineser Journal of Computers*, vol. 36, no. 2, pp. 349–359, 2013.
- [3] M. Verbanck, J. Josse, and F. Husson, "Regularised pca to denoise and visualise data," *Statistics and Computing*, vol. 25, no. 2, pp. 471–486, Mar 2015.
- [4] K. A., "Rbf neural networks for ecg beat classification and arrhythmia detection," *International Journal of Artificial Intelligence and Knowledge Discovery*, vol. 3, no. 3, pp. 9–15, 2013.
- [5] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006.
- [6] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, Dec. 2010.
- [7] Y. Bengio, E. r. Thibodeau-Laufer, G. Alain, and J. Yosinski, "Deep generative stochastic networks trainable by backprop," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ser. ICML'14. JMLR.org, 2014, pp. II–226–II–234.
- [8] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, June 2005.
- [9] F. Zhang and H. Chang, "Employing bp neural networks to alleviate the sparsity issue in collaborative filtering recommendation algorithms," *Journal of Computer Research and Development*, vol. 43, no. 4, pp. 667–672, 2006.
- [10] T. Amaral, L. M. Silva, L. A. Alexandre, C. Kandaswamy, J. M. Santos, and J. M. de Sá, "Using different cost functions to train stacked auto-encoders," in *Proceedings of the 2013 12th Mexican International Conference on Artificial Intelligence*, ser. MICAI '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 114–120.
- [11] S. Gong, H. Ye, and H. Tan, "Combining memory-based and model-based collaborative filtering in recommender system," in *2009 Pacific-Asia Conference on Circuits, Communications and Systems*, May 2009, pp. 690–693.
- [12] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "The adaptive web," P. Brusilovsky, A. Kobsa, and W. Nejdl, Eds. Berlin, Heidelberg: Springer-Verlag, 2007, ch. Collaborative Filtering Recommender Systems, pp. 291–324.

# Towards business identification modeling: A Taobao Case Study

Rong Zhang<sup>1,2#</sup>, Yuyu Yin<sup>1,2\*</sup>, Meng Xi<sup>3#</sup>, Hao Jiang<sup>1,2</sup>

<sup>1</sup>School of Computer, Hangzhou Dianzi University, Hangzhou, 310018, China

<sup>2</sup>Key Laboratory of Complex Systems Modeling and Simulation of Ministry of Education, Hangzhou, 310018, China

<sup>3</sup>College of Computer Science and Technology, Zhejiang University, Hangzhou, 310007, China

\*corresponding author #co-first authors

1312898631@qq.com, yinyuyu@hdu.edu.cn, ximeng@zju.edu.cn, 344045838@qq.com

**Abstract**—With the appearance of new retail, e-commerce has broken the traditional pattern, different categories of goods have their own unique business attributes. Taking transaction business as an example, the traditional physical goods business needs to complete the transaction through the logistics, while the new electronic voucher business achieves that by involving shop verification and the transaction will be totally completed after the consumption of the virtual goods such as the QQ coin. Traditional integral modeling that describes all businesses through a process has been hard to meet such a scenario. In recent years, there have been a lot of studies on business process modeling. These methods mainly focus on the process and data level and do not support business modeling well. In this work, we construct a Business Identification Model(BIM) based on four business sources to handle unique complex business process. We develop a platform based on BIM and verify our model in the real processes of our cooperation company. In addition, BIM supports assembling and reusing in business-level in our case.

**Keywords**- business model; business resource; reuse; new retail

## I. INTRODUCTION

Due to the emergence of corresponding platforms, e-commerce is developing rapidly. Indeed, with the rapid development of globalization and intensification, these platforms must consider reusing the original resources to speed up the business development. E-commerce platforms must adjust their current business model if they want to remain competitive. However, in actual business development, random expression leads to shortage of logical compactness in that people with different business backgrounds may have different understandings. A new requirement needs to complete multiple docking processes, from demand to realization or vice versa. Since original business resources are not effectively combined and normalized, they cannot be reused. These problems result in a low reactivity to adapt business developments to frequently update requirements.

To solve these problems, the concept of “business modeling” has been put forward in recent years, and plenty of research results have been achieved. For example, Gomes J F [1] applied business models as the basis for analyzing learning environments and proposed three different business modeling options that further expand the scalability and feasibility of the business model. There is also a significant amount of research on business processes. Then, Chiara Di Francescomarino [2]

modeled the process of sequence, conjunction and disjunction logic relationships, but these processes are not suitable for complex loop and nested process structures. Based on analyzing the weak termination of business processes in detail, Hee K M V [3] proposed a top-down method of business process by using the Petri network. Yu H [4] proposed a new modeling method that combines IDEF and UML together to achieve a full life cycle of enterprise modeling goals. On the basis of the model established by the Petri network, the process was analyzed to complete the business modeling through the reachable matrix using the Petri network [5].

To a certain extent, the emergence of the above business modeling technology solves the problems of slow development and inefficient interactions between new and old businesses caused by business confusion. However, as most business modeling techniques focus on the business design and implementation level, the conceptual level is ignored. In addition, e-commerce business is complex, and traditional business modeling methods cannot match with the e-commerce business; These models cannot express the business resources needed in the operation process.

Therefore, this article draws on the excellent part of the above business modeling methods, re-analyzes the e-commerce business model and proposes a business modeling method—Business Identification Model(BIM)for the e-commerce field. The model takes the process as the core, models the business resources of different granularity, abandons the traditional model of business custom development and realizes the development of business configuration.

The main contributions of this paper are as follows: (1) propose the Business Identification Model(BIM). Based on reusable models, new businesses can quickly find candidate models that satisfy personalized business personalized requirements. Personalized selection and configuration of business can be realized based on candidate models; (2) formally define business resources with different granularity. By defining a set of unified description criteria for businesses, the criteria are used to express the relationship between business concepts and business resources, which can reduce ambiguity and reduce communication costs between business people and developers.

This paper is organized as follows. Section two is the motivation case, which illustrates the current problems in



nowadays business development. The detailed introduction and formalized definition of BIM are illustrated in section three. Section four discusses a case study and section five contains the comparison. Section six is related work and section seven is the conclusion.

## II. MOTIVATION CASE

Business patterns are increasingly diversified, such as B2B, C2C, O2O and B2C. The types of goods sold on the platforms are also more diversified, from single physical goods to service goods and virtual goods. Taking the electronic voucher business as an example, customers purchase goods on the e-commerce platform, obtain the corresponding consumption vouchers, and obtain the corresponding service or merchandise in the physical store by vouchers. The business development process is shown in Fig. 1 below.

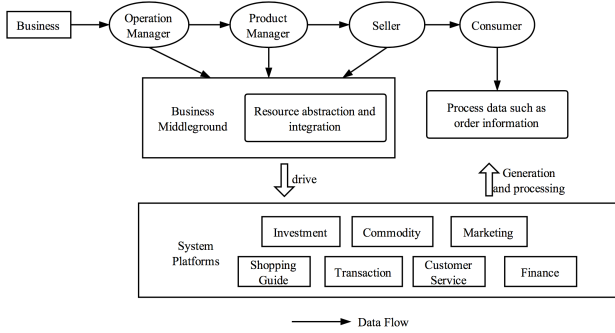


Figure 1. Development process diagram of the electronic voucher

After investigation and interview the actual business development in the Intermediate platform, we summarized the following three phenomena:

**Phenomenon 1:** A business may involve multiple teams. Taking the above electronic voucher business as an example, the following seven business teams are involved: Merchant Team, Commodity Team, Shopping Guide Team, Marketing Team, Trading Team, After-Sales Team and Capital Team. Due to the complexity of cooperation between business teams and the lack of strong business models, the overlapping of requirements of business teams can easily lead to logical conflicts and vulnerabilities.

**Phenomenon 2:** The requirement is not planned, and there is much concurrent and sudden requirement. As businesses diversify, requirement changes more frequently. The traditional custom development steps are as follows. First, the business people discuss the requirements, and then the developers develop the business. Without considering the reuse of the original resources, it is impossible to achieve high efficient development efficiency and achieve the vision of low cost trial and error.

**Phenomenon 3:** A lot of businesses in different scenes may be similar. The business of electronic voucher transaction completes the transaction through the verification of vouchers, while the business of physical transaction mainly relies on the logistics to complete the transaction. These businesses are very

similar. For example, they all must experience investment, product launches, shopping guide, marketing, trading, sales and other sectors. The main difference is that the logistics link is turned into a physical store.

The differences among these three phenomena result in problems which exist ever in intermediate platforms. As the business grows and changes, the problems become more intractable and urgent in new business situations:

**Problem1:** The requirement response cycle is long. A simple requirement may involve multiple business teams. High communication costs and long waiting times seriously would affect the efficiency of business development seriously.

**Problem2:** Reuse of business-level resources is difficult. In an actual e-commerce system, the business logic and platform logic are coupled together that causes difficulties in reuse. In addition, there are many business resources in existing systems with no uniform specifications, such as requirements, process, which further causes difficulties in reuse.

## III. BUSINESS IDENTIFICATION MODEL

The two problems have become increasingly severe in the actual business development as the Taobao business has evolved from a single-center simple process to a complex multi-center process. Supporting consumer products effectively through business modeling, realizing business reuse and expanding more business possibilities have become important issues in business development.

To solve these problems, this paper proposes a Business Identification Model. The model consists of four-layer with each layer defining a different granularity of business resources from the following: business process, page template, business service, and ability. The model manages these resources hierarchically. The business process is used to manage the activities in the business execution as well as the sequence and logic of the activity execution. A page template provides visual support for a business process that consists of multiple components. A component is also referred to as a business services. Business services encapsulate the business logic with a single portability. Decomposing business services to obtain the smallest logical unit is called ability. Ability does not involve business logic, and is the smallest size functional component. The UML class diagram of notions in SLM is shown in Fig. 2.

Next, we will introduce the BIM systematically. Here we provide our definitions of the nations in the BIM in a top-down manner. Below is the definition of a business identification (see Definition 1).

**Definition 1.** A business identification of the BIM is a seven-tuple  $(ID, D, BPs, PMs, BSs, CAs, R)$ , where  $ID$  is the identifier of business,  $D$  is the description of the business,  $BPs$  are a family of processes over  $ID$ ,  $PMs$  are a family of Page templates with respect to  $BPs$  and  $ID$ ,  $BSs$  are a family of business services with respect to  $PMs$  and  $ID$ ,  $CAs$  are a family of abilities services with

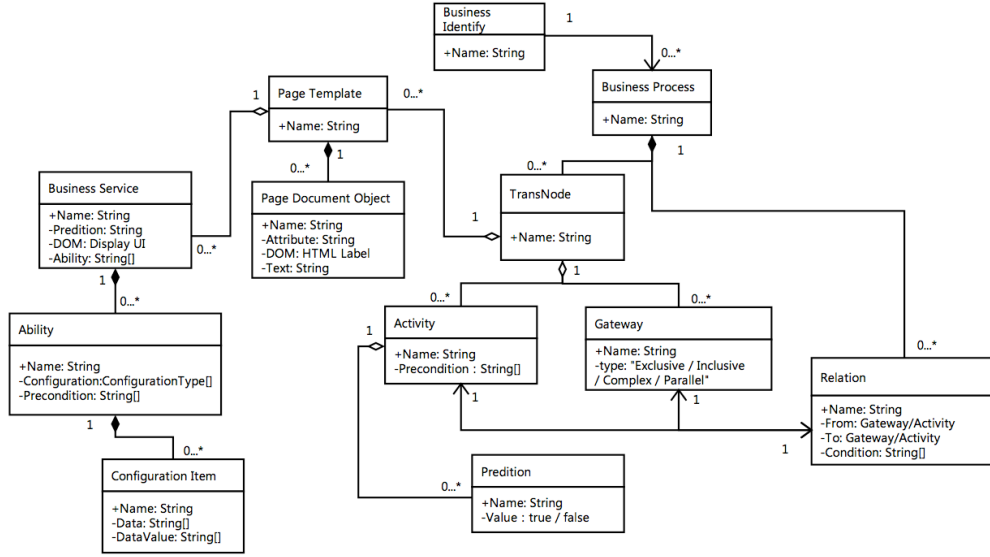


Figure 2. The UML class diagram of basic notions in BIM

respect to PMs and ID,  $R$  is the relationships between BPs, PMs, BSs and CAs.

#### A. Business Process

A business process is an abstract expression of activities and the logical sequence of execution in the process of business execution. The business process is composed of activity and gateway nodes and can facilitate the business staff to control the execution of the business at the macro level, allocate resources and reuse similar processes.

**Definition 2.** A business process is a five-tuple  $(PID, PD, a, g, r)$ , where  $PID$  is the identifier of the business process,  $PD$  is the description of the business process,  $a$  are a family of activities and  $g$  are a family of gateways and  $r$  are a family of the sequence relations of all nodes in the process.

**Definition 3.** An activity instance of the business process is a binary  $(a, Ca)$ , where  $a$  is the identifier of activity and  $Ca$  is the precondition of the activity.

**Definition 4.** A gateway instance is a binary  $(g, type)$ , where  $g$  is the identifier and  $type$  is either Exclusive / Inclusive / Complex / Parallel. The condition of the gateway is decoupling onto the relation condition behind it.

**Definition 5.** A relation is a four-tuple  $(r, from, to, c)$ , where  $r$  is the identifier,  $from$  and  $to$  are the name of the activity or gateway, respectively, and  $c$  is the condition on this relation.

#### B. Page Template

A page template is a visual support for a business process, and an active node or gateway node in a business process corresponds to one or more page templates. A page template consists of multiple components, and these components have a single portability.

**Definition 7.** A page document object is a four-tuple  $(TID, V, DOM, Text)$ , where  $TID$  is the identifier of the page document object,  $V$  is the attributes of the page document object,  $DOM$  is the HTML labels of the page document object, and  $Text$  is the text content of the page document object.

**Definition 8.** A document object attribute is a key value pair  $(k, v)$ , where  $k$  is the identifier of the attribute and  $v$  is the value of the attribute.

#### C. Business Service

The components on the page template are called business services. Business services encapsulate the logic of business operation, which can save the cost of page building, improve the efficiency of development, and realize the reuse of pages.

**Definition 9.** A business service is a four-tuple  $(BID, BD, Cb, Dom)$ , where  $BID$  is the identifier of the business service,  $BD$  is the description of the business service,  $Cb$  is the precondition of the business service configuration, and  $DOM$  is the display UI of the business service.

#### D. Ability

Decomposing business services and obtaining the smallest functional units are referred to as abilities. Ability does not involve business logic, and it is convenient for developers to focus on developing basic functional units.

**Definition 10.** An ability is a four-tuple  $(AID, FA, Ca, Cab)$ , where  $AID$  is the identifier of the ability,  $Ca$  is the configuration item of the ability,  $FA$  is the data involved in the ability, and  $Cab$  is the precondition of the ability.

**Definition 11.** A configuration item is a four-tuple  $(EID, ED, K, Vs)$ , where  $EID$  is the identifier of the configuration item,  $ED$  is the description of the configuration item,  $K$  is the data involved in the ability, and  $Vs$  is the range of values.

**Definition 12.** A precondition is a triple  $(PID, PD, Pv)$ , where  $PID$  is the identifier of the precondition,  $PD$  is the description,  $Pv$  is the Boolean value that is true or false.

By establishing the BIM, the application systems and business systems can manage the correspondence between the

business standard and the various modules. This model can isolate business logic, distinguish business data, and realize the reuse of business, which is of great application value.

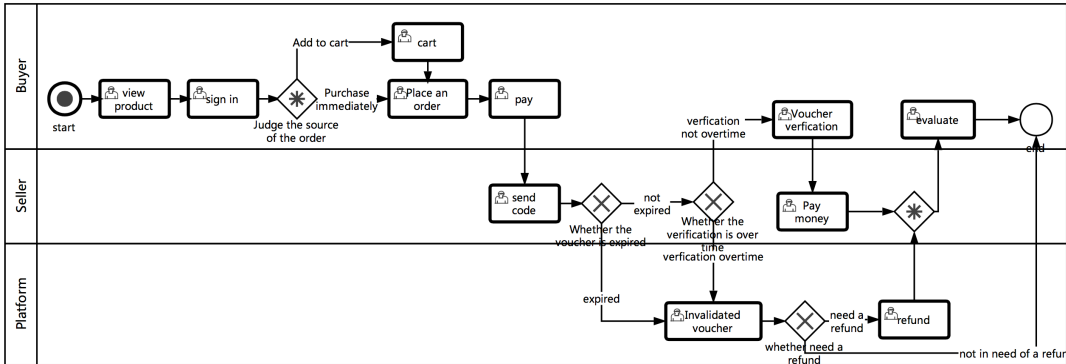


Figure 3. A trading process of the electronic voucher

#### IV. CASE STUDY

To illustrate the BIM, we applied it to an example of constructing a transaction business of electronic voucher. The transaction business of electronic voucher is a new transaction model in which users buy goods or services on an e-commerce platform and write off electronic vouchers provided by the platform in off-line stores. The development process of the business is discussed in the motivation case (see Figure 1 in section 2). We used the BIM to construct a simplified transaction business to illustrate how it works.

**Construct business process.** When building a new business, we must identify the activities involved in the business, define the performing roles of each activity, and then configure the gateway nodes according to the order of activities (see Fig. 3).

**Example 1.** In this example, a send code activity was used. First, we determined that the activity following a send code activity is a voucher verification activity. We then cleared the logic conditions between activities when and only when the voucher was not expired and the verification was not overdue. Finally, we configured the roles of the following two activities separately: the seller and the platform activities.

**Construct Page Template.** Each activity node or gateway node corresponds to one or more page templates designed by business personnel. The rationality and quantity of pages affect user experience and the commodity conversion rate.

**Example 2.** In this example, we used the activity of placing an order as an example to configure the page template corresponding to the activity. The activity corresponds to a detail page; The platform collects user information through the page, and displays the specific information of the order for user confirmation.

**Construct Business Service.** Business services are components of page templates, and page templates were obtained by configuring and reorganizing multiple business

services. The design of business services affects the efficiency of page building.

**Example 3.** In this example, in order to configure the detailed page of placing an order, we designed three business services: user information, order information, and price information. The business service of personal information is presented in detail in Table I.

TABLE I. BUSINESS SERVICE EXAMPLE: USER INFORMATION

<i>business service id</i>	confirmation_personal_information
<i>business service description</i>	select personal information
<i>pre_condition</i>	true
<i>show_UI</i>	Dropdown box

**Construct Ability.** Ability refers to a functional unit with minimal granularity, which can be configured through a configuration item, and one or more abilities constitute a business service.

**Example 4.** In this example, we assembled abilities in the business service of order information and then configured each of the abilities. The business services consist of the following four abilities: display attributes, select goods, select preferential methods, and select freight insurance. The ability of selecting freight insurance is presented in detail in Table II.

TABLE II. ABILITY EXAMPLE: SELECT FREIGHT INSURANCE

<i>ability_id</i>	select freight insurance
<i>business service description</i>	can get indemnity if return goods
<i>pre_condition</i>	true
<i>configuration item</i>	amount of compensation: [9,12]

## V. COMPARISON

In this section, we compare the BIM, BPMN2.0 and Artifact BPM as shown in table III, to illustrate the advantages of the BIM.

The e-TOM model has the following three large process domains: product process domain, operating domain, enterprise management domain. The three large process domains can be further decomposed into 23 process groups and a certain

number of two-level, three-level, and four-level processes. Graphical elements are used to represent the execution behavior of a business in BPMN2.0. In Artifact BPM, the process that is called lifecycle, are working around one “artifact”. The BIM divides the entire link of the e-commerce process into 11 sub-links, which not only regulates the e-commerce business operation mechanism, but also supports diversified business processes.

TABLE III. COMPARISON BETWEEN AND OTHER MODELING METHODS

<i>business service description</i>		eTOM	BPMN2.0	Artifact BPM	BIM
<i>Field</i>		Telecommunication	All	All	e-commerce
<i>process</i>		√	√	√	√
<i>service</i>		×	×	√	√
<i>page template</i>		×	×	×	√
<i>ability</i>	<i>data operations</i>	×	√	√	√
	<i>data</i>	×	×	√	√

The business process cannot be perceived by the end users. To describe the business operations process in a form that the end user can understand, business personnel defines processes by visual means. The traditional business modeling method lacks of intuitive traits and has a low degree of visualization. In the BIM, page template elements are added to support the show of business on different clients.

In different business models, services also have different meanings. The business service in the BIM is a combination of many abilities and represents a logical component with a special function. Business services are decoupled from each other, and we replace any business service will not affect the operation of entire system, greatly decreasing the system coupling degree.

In a traditional model, functions, interfaces, and services can all be called abilities. Abilities are often used in the management and maintenance at the code level, and only the program developers can understand the meaning of ability. In the BIM, every ability represents the operation of a data. We can get the data flow process in the whole life cycle of the business. Both developers and business people have a clear understanding of the use and applicable scenes of abilities, which play a very good role in fast building business.

## VI. RELATED EORK

### A. BPM

Business modeling is a modeling approach to describe the objects and elements involved in enterprise management and business, as well as the attributes, behaviors and relationships between them. In recent years, academic circles proposed many modeling methods toward process, organization, and domain. In 2004, White S A [6] formally proposed the concept of BPMN (Business Process Modeling Notation), which is as a symbolic standard for business process modeling. BPMN describes a business process with a series of graphical elements, for example, the rectangle represents activity, and the diamond

represents the condition so that the reader can simply identify the basic types of the element to understand the graphics. Based on BPMN, Rodríguez A [7] summarized the extension for BPMN on how to modeling secure business process, and apply the approach to a typical health-care business process.

P Wohead [8] developed an evaluation framework using the Workflow Patterns, to examine the suitability of the Business Process Modelling Notation (BPMN) for business process modelling. To save the time of compliance checking, Awad A [9] translated Compliance rules to temporal logic formulae as input to checkers. Then, checkers can verify whether the business process model meets the compliance rules. Aiming at BPMN 2.0, Krzysztof Kluza [10] proposed rule-based pattern perspective for process models, described how to perceive rules in the business processes. However, the common BPMLS such as EPC or BPMN2.0 provides a set of common process modeling elements, but it does not allow modeling of domain specific concepts. To solve the situation, K Kluza [11] made an extension to different business process modeling languages with domain specific concepts.

Business Process Execution Language (BPEL) is another major standard. BPEL is a language that uses Web services to define and execute business processes. The main function is to combine existing services to define a new service [12], [13], [14]. To define choreographies, G Decker [15] proposed BPEL4Chor to extend BPEL. G Decker distinguished extensions from the following three aspects: participant behavior descriptions, the participant topology, participant groundings.

### B. Artifact-centric BPM

As we all know, Nigam A et al [16] firstly proposed the concept of business artifact in 2003. Business Artifacts is used to record information in chunk that indivisible, identifiable, concrete, and self-describing. Then, Artifact-centric business process models were proposed and analyzed [17][18]. To deploy business artifacts, Joseph H R et al [19] proposed a

novel framework for integration of business artifacts. Estañol M et al [20] proposed using UML diagrams to specify business process that is artifact-centric to represent the dimensions of the artifact-centric approach.

Due to the advantages of business artifacts, both industrial and academic field are attracted by artifact-centric business process models. Considering flexibility of the business process, Truong T M et al [21] connected the behavior of business processes with business artifacts that is described conceptually, redefined process configuration and the context of process redesign. Then, to reuse business process components, Kabir M A [22] made an extension to BPMN and proposed a reusable process pattern and verified the applicability of the pattern. Besides, some further studies on improving reusability and flexibility of data-centric web services have been proposed [23].

## VII. CONCLUSION

In this paper, we propose Business Identification Model based on a case study of the e-commerce industry. The Business Identification Model is a four-layer hierarchical model, which supports views of the following business resources: business process, page template, business service and ability. This model extends traditional process-centric or data-centric process models and integrates four resources with different granularity. The model can standardize resources uniformly, make business visualizations, facilitate the reuse of business resources, and realize the on-requirement configuration and rapid development of new business simultaneously. In the cooperative enterprise, we have applied some practical data and resources to verify the feasibility and effectiveness of the model.

In future work, we will further study the impact of rules on business and improve the Business Identification Model. In the future, we will have further cooperation with e-commerce companies to verify the reliability of the model with more real and complex e-commerce data and scenarios. In addition, we will optimize the model and realize the economic efficiency and the social efficiency goal through application of this model to the actual project.

## ACKNOWLEDGMENT

This work was supported by following projects: This paper is supported by national key research and development program of China under grant (No.2017YFB1400601), Zhejiang Provincial Natural Science Foundation (No.LY12F02003), and National Natural Science Foundation of China (No.61100043).

## REFERENCES

- [1] Gomes J F, Ahokangas P, Moqaddamerad S. "Business Modeling Options for Distributed Network Functions Virtualization: Operator perspective," in *European Wireless conference*, 2016, pp. 37-42.
- [2] Chiara Di Francescomarino, Francesco Corcoglioniti, Mauro Dragoni, Piergiorgio Bertoli, Roberto Tiella, Chiara Ghidini, Michele Nori, and Marco Pistore. "Semantic-Based Process Analysis," in *International Semantic Web Conference*, 2014, pp. 228-243.
- [3] Hee K M V, Sidorova N, Werf J M V D. "Business Process Modeling Using Petri Nets," *Transactions on Petri Nets & Other Models of Concurrency VII*, vol. 7480, no. 5, pp. 116-161, 2013.
- [4] Yu H, Wu D. "Enterprise Modeling Based on IDEF and UML," in *International Conference on Advanced Information Technology and Sensor Application*, 2016, pp. 59-62.
- [5] Dong G, Qian-Sheng F U. "Study Simultaneous Happening Function of E-commerce System on the Basis of Petri Net," *Policy-making Reference*, vol. 12, no. 3, pp. 1-4, 2004.
- [6] Y. Li, Z. Luo, J. Yin, L. Xu, Y. Yin, and Z. Wu, "Enterprise pattern: integrating the business process into a unified enterprise model of modern service company," *Enterprise Information Systems*, vol. 11, no. 1, pp. 37-57, 2015.
- [7] A. Rodriguez, E. Fernandez-Medina, and M. Piattini, "A bpmn extension for the modeling of security requirements in business processes," *Leice Transactions on Information & Systems*, vol. E90D, no. 4, pp. pgs. 745-752, 2007.
- [8] P. Wohed, W. M. P. V. D. Aalst, M. Dumas, A. H. M. T. Hofstede, and N. Russell, "On the suitability of bpmn for business process modelling," *Lecture Notes in Computer Science*, vol. 4102, no. 3, pp. 161-176, 2006.
- [9] Awad A, Decker G, Weske M. "Efficient Compliance Checking Using BPMN-Q and Temporal Logic," in *International Conference on Business Process Management*, 2008, pp. 326-341.
- [10] Kluza K, Nalepa G J. "Towards rule-based pattern perspective for BPMN 2.0 business process models," in *Proceedings of the 2016 Federated Conference on Computer Science and Information Systems Computer Science and Information Systems*, 2016, pp. 1359-1364.
- [11] Radloff M, Schultz M, Nüttgens M. "Extending different Business Process Modeling Languages with Domain Specific Concepts: The Case of Internal Controls in EPC and BPMN," in *International Workshop on Enterprise Modelling and Information Systems Architectures*, 2015, pp. 2432-2442.
- [12] X. Fu, T. Bultan, and J. Su, "Analysis of interacting bpel web services," in *International Conference on World Wide Web*, 2004, pp. 621-630.
- [13] C. Ouyang, E. Verbeek, W. M. P. V. D. Aalst, S. Breutel, M. Dumas, and A. H. M. T. Hofstede, "Formal semantics and analysis of control flow in ws-bpel," *Science of Computer Programming*, vol. 67, no. 2, pp. 162-198, 2005.
- [14] M. Mongiello and D. Castelluccia, "Modelling and verification of bpel business processes," in *Model-Based Development of Computer-Based Systems and Model-Based Methodologies for Pervasive and Embedded Software*, 2006. MBD/MOMPES 2006. Fourth and Third International Workshop on. IEEE, 2006, pp. 5-pp.
- [15] G. Decker, O. Kopp, F. Leymann, and M. Weske, "Bpel4chor: Extending bpel for modeling choreographies," in *IEEE International Conference on Web Services*, 2007, pp. 296-303.
- [16] A. Nigam and N. S. Caswell, "Business artifacts: An approach to operational specification," *Ibm Systems Journal*, vol. 42, no. 3, pp. 428-445, 2003.
- [17] K. Bhattacharya, C. Gerede, R. Hull, R. Liu, and J. Su, "Towards formal analysis of artifact-centric business process models," in *International Conference on Business Process Management*, 2007, pp. 288-304.
- [18] D. Cohn and R. Hull, "Business artifacts: A data-centric approach to modeling business operations and processes," in *Enterprise Systems Conference*, 2009, pp. 3-9.
- [19] Joseph H R, Badr Y. "Business Artifact Modeling: A Framework for Business Artifacts in Traditional Database Systems," in *Enterprise Systems Conference*, 2015, pp. 13-18.
- [20] M Estañol, A Queralt, MR Sancho, E Teniente. "Artifact-Centric Business Process Models in UML," in *Business Process Management Workshops*, 2012, pp. 292-303.
- [21] Truong T M, Lê L S. "On Business Process Redesign and Configuration: Leveraging Data Mining Classification & Outliers and Artifact-Centric Process Modeling," in *International Conference on Advanced Computing and Applications*, 2017, pp. 59-66.
- [22] Kabir M A, Xing Z, and Chandrasekaran P. "Process Patterns: Reusable Design Artifacts for Business Process Models," in *Computer Software and Applications Conference*, 2017, pp. 714-721.
- [23] R. Vaculn, T. Heath, and R. Hull, "Data-centric web services based on business artifacts," in *IEEE International Conference on Web Services*, 2012, pp. 42-49.

# Ontology-based Software Architectural Pattern Recognition and Reasoning

Nacha Chondamrongkul<sup>1</sup>, Jing Sun<sup>2</sup> and Ian Warren<sup>3</sup>

Department of Computer Science, University of Auckland, New Zealand

<sup>1</sup>ncho604@aucklanduni.ac.nz, <sup>2</sup>jing.sun@auckland.ac.nz, <sup>3</sup>i.warren@auckland.ac.nz

**Abstract**— Designing software architecture is a knowledge-intensive task that typically involves textual and diagrammatic notation. Using these kinds of notation is often inconsistent, misleading, and ambiguous. Ontology representation is, therefore, a suitable approach, as it can semantically define architectural design model that can be automatically verified through reasoning. However, a large-scale software system is usually complex and applies more than one architectural styles with various behavioral patterns. Therefore, the scalability of automated verification for a complex software architecture design is a challenge. We propose an approach that helps to formally define complex architectural design model and automate different verifications such as consistency checking, architectural styles recognition, and behavioral sequence inference. Ontology Web Language (OWL) is used to semantically define basic architectural elements and architectural styles, while a set of rules defined in Semantic Web Rule Language (SWRL) helps to capture behavioral pattern according to style. We evaluated the scalability of our approach. The result shows that different levels of complexity in architectural design model has a minor impact on the verification performance.

## I. INTRODUCTION

Software architecture is typically a conceptual design that decomposes a software system into a set of logical components. At the early phase of software development process, software architecture is designed to meet specific functional requirements, non-functional requirements and business goals. Software architecture, therefore, encapsulates set of early design decisions tradeoffs and constraints, which provide a guideline to implement software system throughout the development lifecycle. Unfortunately, software architecture is often abstract and informally presented by the combination of textual and graphical notation that are often misleading, ambiguous and inconsistent. Although, several standardized architecture description languages (ADL) have been proposed, such as ISO/IEC/42010 [1] and UML [2], they have little or no formal semantic support. Without semantic constraints, the verification of architecture design is, therefore, a daunting task. Moreover, the large-scale software system is usually a complex entity that applies predefined architectural styles, each style characterizes specific type of component and their behavior. Even though, a few ADLs, such as ACME [3], support abstraction of architecture into reusable styles but they have no semantic that enforces style constraints. ADLs has little popularity among practitioners because they are lack of tools to support, and yet require high learning curve [4]. The inadequate

mechanism of producing accurate software architecture model catalyzes applying formal methods into this area.

Formal methods have played an important role in software engineering research for some time. A number of researchers have applied ontology technique to software development lifecycle [5], in order to resolve ambiguous, prevent errors and minimize cost in different phases, from requirement gathering [6] to software maintenance [7]. For software architecture, in particular, architectural design model is formally specified, in pursuance of automated verification [8]. However, the performance of automated verification in large-scale software is still an open issue for existing approach such as Alloy [9]. Although, Wong et al [10] proposed a solution that allows model to be decomposed, in order to parallelize verification process. However, dependencies between components still require verification process to be executed in sequential manner. The ontology has been proposed to apply in designing software architecture because of its strength in effective large-scale reasoning that can automate consistency checking and hierarchy inference in the design model [11, 12, 13]. Pahl et al. [14] integrated ontology into ACME, in order to verify consistency of an architecture and its behaviors, but process modeling notation is still a limitation. In pursuance of consistency checking automation, style recognition, and communication inference, Sun et al. [15] proposed to use Ontology Web Language (OWL) [16] to formally specify different entities and relationships in architectural design. The communication flow can be captured by rules based on Semantic Web Rule Language (SWRL) [17]. However, the performance of automated verification was not evaluated, and the range of provided architectural styles is limited.

The main challenge is how we can semantically define complex architectural design based on multiple styles, and evaluate how the proposed method impacts to the automated verification performance. We propose an approach as shown in Figure 1. The ontology library includes basic architectural element, architectural styles, and behavioral rules. OWL is used to define basic architectural elements, such as component and connector. These basic architectural elements can be extended to define various architectural styles, and a design instance that represents the architectural design model of a specific software system. As the ontology for architecture design is inevitable complex, we use description logic (DL) based languages, in order to take advantage of existing DL reasoning engine that is

effective in performing large-scale automated reasoning. The consistency in architectural styles and design instance can be automatically checked by the reasoning engine, based on the ontology's constraints defined in basic architectural elements. In order to recognize architectural styles applied in the design, reasoning engine classifies architectural elements into different ontological classes specific to architectural style. After styles are recognized, reasoning engine processes behavioral rules to capture architectural configuration and generate behavioral sequences, which manifest interactions between components.

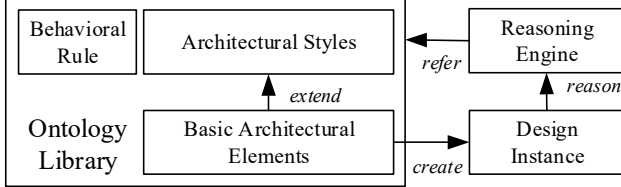


Figure 1 Overview of architectural design approach

The rest of the paper is organized as follows. We present ontology-based architectural styles in Section 2. Section 3 illustrates how a software architecture can be modeled and verified against architectural styles. This paper concludes in Section 4 with future research direction.

## II. ONTOLOGY MODELING FOR ARCHITECTURE STYLES

The ontology library is based on Component & Connector (C&C) [18] view, which aims to exhibit how the system works at runtime. Software architects use C&C view for reasoning about key system quality attributes such as performance, security, and reliability [19]. In C&C, a component represents a processing unit within the software system, while connectors define interaction mechanism between components. The component has a set of ports that serves communication to another component, whereas connector has a number of roles, each has specified set of actions it specifically performs. A component can be associated with a connector by attaching its port to a connector's role. Based on C&C view's concept, our ontology library has consisted of ontological classes representing fundamental architecture elements, namely *Component*, *Connector*, *Port*, *Role*, and *Action*. *hasAttachment* is defined as an object property to associate component's port with connector's role. *Action* is assigned to *Role* via *hasAction* property with minimum cardinality restriction, in order to make sure that a role has at least one assigned action. Below are ontology classes expressed in description logic syntax.

$$\begin{aligned} \text{ArchElement} &\sqsubseteq \text{Component} \sqcup \text{Connector} \sqcup \text{Interface} \sqcup \text{Action} \\ \text{Interface} &\sqsubseteq \text{Port} \sqcup \text{Role} \\ \text{Component} \sqcap \text{Connector} \sqcap \text{Interface} \sqcap \text{Action} &\equiv \perp \\ \text{Port} \sqcap \text{Role} &\equiv \perp \\ \text{Component} &\equiv \text{ArchElement} \sqcap \exists \text{hasPort Port} \\ \text{Connector} &\equiv \text{ArchElement} \sqcap \exists \text{hasRole Role} \\ \text{Port} &\equiv \text{Interface} \sqcap \exists \text{hasAttachment Role} \\ \text{Role} &\equiv \text{Interface} \sqcap \geq 1 \text{hasAction Action} \end{aligned}$$

C&C can be characterized by various architectural styles. Each style specifies a particular set of component, connector, and

behavioral pattern. The following are some architectural styles included in our ontology library.

### A. Client-Server Style

The client and server are two key component types in this style. *Cns:Client* and *Cns:Server* are defined as classes, extended from *Component*. *Request* and *Response* are port type attached to client and server respectively. The port attachment is defined by *hasPort* property with existential restriction, as a component can be classified as several component types.

$$\begin{aligned} \text{CnS:Client} &\equiv \text{Component} \sqcap \exists \text{hasPort Request} \\ \text{CnS:Server} &\equiv \text{Component} \sqcap \exists \text{hasPort Response} \\ \text{Request} &\equiv \text{Port} \sqcap \exists \text{hasAttachment Consumer} \\ \text{Response} &\equiv \text{Port} \sqcap \exists \text{hasAttachment Provider} \end{aligned}$$

A corresponding connectivity is defined as a connector class to incorporate two roles, *Consumer*, and *Provider*. *Consumer* requests services on the server, while *Provider* performs actions to process the request and return the result back to *Consumer*. *SendRequest*, *ReceiveResult*, *ServerInvoked* and *ReturnResult* are defined as subclasses of *Action*, in order to represent different activities and events in client-server style.

$$\begin{aligned} \text{CnSConnector} &\equiv \text{Connector} \sqcap \exists \text{hasRole Consumer} \\ &\quad \sqcap \exists \text{hasRole Provider} \end{aligned}$$

$$\begin{aligned} \text{CnS:Consumer} &\equiv \text{Role} \sqcap \exists \text{hasAction SendRequest} \\ &\quad \sqcap \exists \text{hasAction ReceiveResult} \\ \text{CnS:Provider} &\equiv \text{Role} \sqcap \exists \text{hasAction ServerInvoked} \\ &\quad \sqcap \exists \text{hasAction ReturnResult} \\ \text{SendRequest} &\sqsubseteq \text{Action} & \text{ServerInvoked} &\sqsubseteq \text{Action} \\ \text{ReceiveResult} &\sqsubseteq \text{Action} & \text{ReturnResult} &\sqsubseteq \text{Action} \end{aligned}$$

The behavioral rules are defined to associate relevant actions as a sequence according to the style's behavioral pattern. In order to generate a behavioral sequence, the *hasNextAction* property is used to define what action comes next in the sequence. Below is a rule defined in SWRL, it captures behavioral pattern as follows. At first, when the client sends a request, the server will be invoked. This rule hence implies *ServerInvoked* as the next action to *SendRequest*. After the server finishes processing and returns the result, the client will receive the result. This rule, therefore, implies *ReceiveResult* as the next action of *ServerReturn*.

$$\begin{aligned} &\text{CnSConnector}(\text{?cns}) \sqcap \text{hasRole}(\text{?cns}, \text{?cr}) \sqcap \text{CnS:Server}(\text{?server}) \sqcap \\ &\text{isPortOf}(\text{?p}, \text{?server}) \sqcap \text{SendRequest}(\text{?sreq}) \sqcap \text{hasAction}(\text{?cr}, \text{?sreq}) \sqcap \\ &\text{Provider}(\text{?pr}) \sqcap \text{isAttachmentOf}(\text{?pr}, \text{?p}) \sqcap \text{Consumer}(\text{?cr}) \sqcap \\ &\text{ReceiveResult}(\text{?rres}) \sqcap \text{hasRole}(\text{?cns}, \text{?pr}) \sqcap \text{hasAction}(\text{?pr}, \text{?invs}) \sqcap \\ &\text{ServerInvoked}(\text{?invs}) \sqcap \text{ServerReturn}(\text{?sret}) \sqcap \text{hasAction}(\text{?pr}, \text{?sret}) \sqcap \\ &\rightarrow \text{hasNextAction}(\text{?sreq}, \text{?invs}) \sqcap \text{hasNextAction}(\text{?sret}, \text{?rres}) \end{aligned}$$

Below is another rule that captures an occurrence when the server is invoked to process the request. After that, the result will be returned to the client. This rule hence implies *ServerReturn* as the next action of *ServerInvoked*

$$\begin{aligned} &\text{Response}(\text{?p}) \sqcap \text{isPortOf}(\text{?p}, \text{?server}) \sqcap \text{hasAction}(\text{?p}, \text{?iser}) \sqcap \\ &\text{Provider}(\text{?pr}) \sqcap \text{isAttachmentOf}(\text{?pr}, \text{?p}) \sqcap \text{CnSConnector}(\text{?cns}) \sqcap \\ &\text{CnS:Server}(\text{?s}) \sqcap \text{Consumer}(\text{?cr}) \sqcap \text{hasRole}(\text{?cns}, \text{?pr}) \sqcap \\ &\text{Request}(\text{?r}) \sqcap \text{hasAction}(\text{?p}, \text{?sret}) \sqcap \text{ServerInvoked}(\text{?iser}) \sqcap \end{aligned}$$

$ServerReturn(?sret)$ ,  $isAttachmentOf(?cr, ?r) \sqcap hasRole(?cns, ?cr)$   
 $\rightarrow hasNextAction(?iser, ?sret)$

### B. N-Tier Style

A number of clients and servers can form a multi-level hierarchy, a tier has consisted of clients that invoke servers on the upper tier. Each tier runs on the separate physical environment so it can be maintained independently of other tiers, however, interaction between tiers rely on each other. For example, the business application typically has 3 tiers namely client, business logic, and data management. A request to service on business logic consequently triggers a request to data management tier. To semantically define this, connector's reliance is defined by *hasLink* property, so a class for tier can be formally expressed as follows:

$NTier: Tier \equiv Component \sqcap \exists hasPort$   
 $(Port \sqcap \exists hasAttachment$   
 $(Role \sqcap \exists isRoleOf$   
 $(Connector \sqcap \exists hasLink Connector)))$

Below is a rule defined to capture behavioral pattern between tiers. When a server on a tier is requested and invoked, it may make a request to the upper tier. When the result is received, it is forwarded to the lower tier. The actions between tiers are related by *hasDivertNextAction* property.

$CnSConnector(?cns) \sqcap hasRole(?cns, ?pr) \sqcap hasRole(?cns, ?cr) \sqcap$   
 $Provider(?pr) \sqcap Consumer(?cr) \sqcap hasAction(?pr, ?invs1) \sqcap$   
 $hasAction(?pr, ?sret1) \sqcap ServerInvoked(?invs1) \sqcap$   
 $ServerReturn(?sret1) \sqcap hasLink(?cns, ?cns2) \sqcap$   
 $CnSConnector(?cns2) \sqcap hasRole(?cns2, ?pr2) \sqcap hasRole(?cns2,$   
 $?cr2) \sqcap Provider(?pr2) \sqcap Consumer(?cr2) \sqcap hasAction(?cr2, ?rret2)$   
 $\sqcap hasAction(?cr2, ?sreq2) \sqcap ReceiveResult(?rret2) \sqcap$   
 $SendRequest(?sreq2)$   
 $\rightarrow hasDivertNextAction(?invs1, ?sreq2) \sqcap$   
 $hasDivertNextAction(?rret2, ?sret1)$

### C. Publish-Subscribe Style

This style has components interacting to each other through events. *Pns:Publisher* is a subclass of *Component* for publisher, a component type that announces events to subscribed component, while *Pns:Subscriber* is a subclass for subscriber, a component type that listens to the events. *Announce* and *Register* are ports for publisher and subscriber respectively. The defined classes for component types and its ports type can be formally expressed as follows:

$PnS: Publisher \equiv Component \sqcap \exists hasPort Announce$   
 $PnS: Subscriber \equiv Component \sqcap \exists hasPort Register$   
 $Announce \equiv Port \sqcap \exists hasAttachment Publisher$   
 $Register \equiv Port \sqcap \exists hasAttachment Subscriber$

*Publisher* and *Subscriber* are defined as role class in this style. The connector is an event bus that coordinates these two roles.

$PnSConnector \equiv Connector \sqcap \exists hasRole Publisher$   
 $\sqcap \exists hasRole Subscriber$   
 $Publisher \equiv Role \sqcap \exists hasAction SubscribeToEvent$   
 $\sqcap \exists hasAction EventAnnounced$   
 $\sqcap \exists hasAction DeliverEvent$   
 $Subscriber \equiv Role \sqcap \exists hasAction ReceiveEvent$   
 $\sqcap \exists hasAction RequestSubscription$

$RequestSubscription \sqsubseteq Action$        $EventAnnounced \sqsubseteq Action$   
 $SubscribeToEvent \sqsubseteq Action$        $DeliverEvent \sqsubseteq Action$   
 $ReceiveEvent \sqsubseteq Action$

The behavioral rule for publish-subscribe style is shown below. This rule captures two occurrences in this style: 1) Subscription: If a subscription is requested by a component, the publisher will acknowledge and subscribe requesting component to an event. Therefore, this rule implies *SubscribeToEvent* to be the next action of *RequestSubscription*. 2) Event Publishing: When a publisher announces an event, the event will be delivered to all subscriber. This rule below infers the sequence of actions as *EventAnnounced*, *DeliverEvent* and *ReceiveEvent* respectively. This sequence is sorted through last two *hasNextAction* property assertions in the rule's implication.

$PnSConnector(?cns) \sqcap Publisher(?p) \sqcap Subscriber(?s) \sqcap$   
 $hasAction(?p, ?feven) \sqcap hasAction(?p, ?seven) \sqcap hasAction(?p,$   
 $?neven) \sqcap FireEvent(?feven) \sqcap NewEventOccur(?neven) \sqcap$   
 $SubscribeToEvent(?seven) \sqcap hasAction(?s, ?reqs) \sqcap hasAction(?s,$   
 $?reven) \sqcap RequestSubscription(?reqs) \sqcap ReceiveEvent(?reven)$   
 $\rightarrow hasNextAction(?reqs, ?seven) \sqcap$   
 $hasNextAction(?neven, ?feven) \sqcap hasNextAction(?feven, ?reven)$

### D. Repository Style

The repository style organizes how data is accessed and stored in software system through centralized repositories. Data repository and data accessor are two major component types in this style. Data repository (*RP:DataRepository*) persists data, manages concurrent access, and supports access control. Data accessor (*RP:DataAccessor*) reads and writes data at one or more repositories.

$RP:DataRepository \equiv Component \sqcap \exists hasPort$   
 $(Port \exists hasAttachment Store)$   
 $RP:DataAccessor \equiv Component \sqcap \exists hasPort$   
 $(Port \forall hasAttachment (Reader \cup Writer))$

We create two connector classes corresponding to writing and reading function in this style. Both connectors associate *Store* role to address where the data persists. *Writer* role identifies the component that requests to write data on the repository, whereas *Reader* role identifies the component that requests to read data on the repository.

$DataReadConnector \equiv Connector \sqcap \exists hasRole Store$   
 $\sqcap \exists hasRole Reader$   
 $DataWriteConnector \equiv Connector \sqcap \exists hasRole Store$   
 $\sqcap \exists hasRole Writer$   
 $Store \equiv Role \sqcap \forall hasAction (ReadData \cup WriteData)$   
 $Writer \equiv Role \sqcap \exists hasAction RequestWrite$   
 $Reader \equiv Role \sqcap \exists hasAction RequestRead$   
 $ReadData \sqsubseteq Action$        $RequestRead \sqsubseteq Action$   
 $WriteData \sqsubseteq Action$        $RequestWrite \sqsubseteq Action$

The behavioral sequence is captured by two rules below. The first rule support reading function so it implies *RequestRead* as precedence action to *ReadData*, likewise, the second rule implies *RequestWrite* as precedence action to *WriteData* action.

$DataReadConnector(?con) \sqcap RequestRead(?reqr) \sqcap$   
 $hasAction(?store, ?read) \sqcap ReadData(?read) \sqcap$



$hasAction(?reader, ?reqr) \sqcap hasRole(?con, ?reader) \sqcap$   
 $hasRole(?con, ?store) \sqcap Store(?Store) \sqcap Reader(?reader)$   
 $\rightarrow hasNextAction(?reqr, ?read)$

$DataWriteConnector(?con) \sqcap Writer(?writer) \sqcap$   
 $RequestWrite(?reqw) \sqcap hasAction(?writer, ?reqw) \sqcap hasRole(?con,$   
 $?store) \sqcap Store(?store) \sqcap WriteData(?write), hasRole(?con,$   
 $?writer) \sqcap hasAction(?store, ?write)$   
 $\rightarrow hasNextAction(?reqw, ?write)$

### III. CASE STUDY & EVALUATION

The online shopping application system is used as a case study to demonstrate our approach. This case study is a sample of complex software system that applies multiple architectural styles. Figure 2 shows its software architecture design that has consisted of four components namely, *TransactionLog*, *PaymentGateway*, *Shopping Mobile App* and *ShopService*. The ports are depicted as small box attached to the components such as *LoggingRequest*, and *PayResponse*. Shopping mobile app has user interfaces that allow the user to purchase the products and make a payment through payment gateway. When a payment is submitted to the payment gateway, a transaction will be recorded by transaction logger. If the user subscribes to price alert service, the notification will be sent when price is updated.

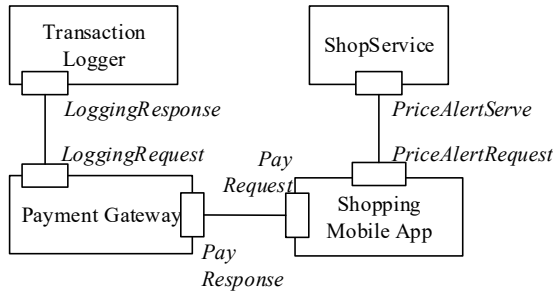


Figure 2 Software architecture design for shopping application

We create ontology instance representing an architectural design model for our case study. The design elements namely actions, roles, port, connectors, and components, are created as individuals that are instances of classes defined in the ontology library. The object properties are used to relate these individuals together, in order to establish a structure in the design model. Due to page limit, we can not show all individuals definition in this paper. The complete definition for this case study can be found at <https://goo.gl/4ugkLB>

The individuals are created for actions with one or more types specified, and they can be formally expressed in OWL abstract syntax as follows:

$Individual(ex: ActRequestToPay \text{ type}(ex: SendRequest))$   
 $Individual(ex: ActRequestToLog \text{ type}(ex: SendRequest)$   
 $\text{ type}(ex: RequestWrite))$   
 $Individual(ex: ActLogTransaction \text{ type}(ex: ServerInvoked)$   
 $\text{ type}(ex: WriteData))$

The roles are defined as individuals with *hasAction* property to the action individuals defined previously. Below are some individuals defined for roles.

$Individual(ex: PaymentProvider$   
 $\text{ value}(ex: hasAction \text{ ex: ActProcessPayment})$   
 $\text{ value}(ex: hasAction \text{ ex: ActReturnPayResult}))$

$Individual(ex: PaymentRequester$   
 $\text{ value}(ex: hasAction \text{ ex: ActRequestToPay})$   
 $\text{ value}(ex: hasAction \text{ ex: ActReceivePayResult}))$

The following are sample individual defined for port. These individuals have one or more relationship to the role individuals through *hasAttachment* property.

$Individual(ex: PayRequest$   
 $\text{ value}(ex: hasAttachment \text{ ex: PaymentRequester}))$

$Individual(ex: PayResponse$   
 $\text{ value}(ex: hasAttachment \text{ ex: PaymentProvider}))$

A number of individuals are created corresponding to communication lines shown in Figure 2. The roles individual are assigned to each connector through *hasRole* property.

$Individual(ex: PaymentService$   
 $\text{ value}(ex: hasRole \text{ ex: PaymentProvider})$   
 $\text{ value}(ex: hasRole \text{ ex: PaymentRequester})$   
 $\text{ value}(ex: hasLink \text{ ex: LoggingService}))$

$Individual(ex: LoggingService$   
 $\text{ value}(ex: hasRole \text{ ex: LoggingProvider})$   
 $\text{ isolate value}(ex: hasRole \text{ ex: LoggingRequester}))$

$Individual(ex: NotificationService$   
 $\text{ value}(ex: hasRole \text{ ex: NotificationPublisher})$   
 $\text{ value}(ex: hasRole \text{ ex: NotificationSubscriber}))$

Each of the components has the corresponding individual created, each component individual is attached with one or more port individuals through *hasPort* properties.

$Individual(ex: PaymentGateway$   
 $\text{ value}(ex: hasPort \text{ ex: PayResponse})$   
 $\text{ value}(ex: hasPort \text{ ex: LoggingRequest}))$

$Individual(ex: ShoppingMobileApp$   
 $\text{ value}(ex: hasPort \text{ ex: PayRequest})$   
 $\text{ value}(ex: hasPort \text{ ex: PriceAlertRequest}))$

$Individual(ex: ShopService$   
 $\text{ value}(ex: hasPort \text{ ex: NotificationPublisher}))$

$Individual(ex: TransactionLogger$   
 $\text{ value}(ex: hasPort \text{ ex: LoggingProvider}))$

As mention previously, the architectural design is defined based on OWL/SWRL, in order to take advantage of classification performed by reasoning engine. The classification results in automating architectural consistency checking, architectural style recognition, and behavioral sequence generation.

#### A. Architectural Consistency Checking

The architectural consistency checking relies on the ontology classification process that verifies consistency in the ontology model and computes hierarchies of defined classes. Figure 3 (a) shows inferred hierarchy of the ontology library when it is consistent. If the classes are consistent, they will be classified into subclasses of basic architectural elements such as component, connector, port, and interface. The inconsistency may be caused by a number of reasons. For example, incompatible classes are associated with domain or range of an object property, or a class has two parents that are disjoint classes. Figure 3 (b) shows a scenario when the ontology library is inconsistent. In this scenario, a class definition for tier

(*NTier:Tier*) contains an axiom  $\exists \text{ hasLink Component}$ , which violates *hasLink* property's constraints that requires *Connector* class as a range. *NTier:Tier* class is, therefore, inconsistent, and it is denoted as a subclass of *owl:Nothing*.

(a) Consistent ontology (b) Inconsistent ontology

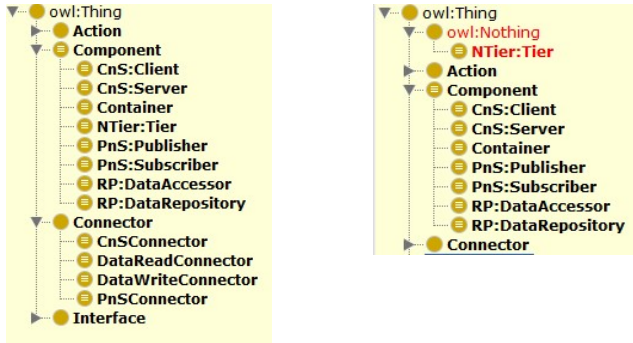


Figure 3 Inferred Hierarchy

When a set of individuals are defined for an architectural design model, careful reader may notice that action is the only design element defined as individual with explicit types. If the architectural design model is consistent, the type of individuals, representing role, port, component and connector, is transitively inferred based solely on their relationships. The architectural styles can be recognized along with inferred types. For example, *Cns:Consumer* in client-server style is a role that has actions namely *ReceiveResult* and *SendRequest*. *PaymentRequester* is thus inferred as an instance of *Cns:Consumer* role, because it has relations to two action individuals namely *ActReceiveResult* and *ActRequestToPay*, which are instances of *ReceiveResult* and *SendRequest* respectively. According to ports class definition, *Request* is a port with some attachment to consumer role. *PayRequest* is thus inferred as an instance of *Request* port, because *PayRequest* is attached to *PaymentRequester* as shown in Figure 4. As defined in *Cns:Client* class, a component is client, if it has some *Request* port. Therefore, *ShoppingMobileApp* is inferred as an instance of *Cns:Client* due to its relation to *PayRequest* port, as shown in Figure 5.

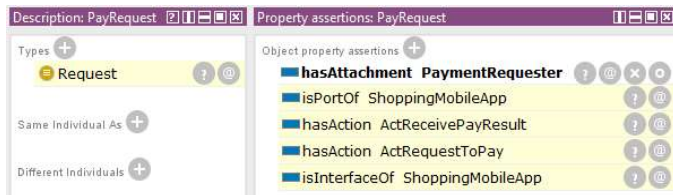


Figure 4 PayRequest Port

### B. Composite Architectural Style Recognition

When more than one architectural styles are applied to the design model, the classification can also identify a composite architectural styled component. The composite architectural styled component is a component that is an instance of several classes from not only the same style but also different styles. As shown in Figure 5, *ShoppingMobileApp* is not only a client but also a tier (*NTier:Tier*) in the multi-tier style. Because

*ShoppingMobileApp* transitively relates to *PaymentService* connector, which has a link to *LoggingService* connector. According to *NTier:Tier* class definition, a tier is a component that transitively relates to a connector, which has a link to another connector. For the same reason, *PaymentGateway* and *TransactionLogger* are also denoted as instances of *NTier:Tier*. Also, *ShoppingMobileApp* has a *Subscribe* port namely *PriceAlertRequest*, it is hence a subscriber in publish-subscribe styles too.



Figure 5 ShoppingMobileApp Component

### C. Behavioral Sequence Generation

After the reasoning engine identifies the type of individuals and the architectural styles are recognized, the reasoning engine will automatically capture the sequence of behavioral activities based on the behavioral rules specific to style. Figure 6 depicts the payment sequence in the online shopping application system. The behavioral rules logically imply *hasNextAction* and *hasDivertNextAction* properties to the action individuals, in order to connect series of action individuals as a sequence. The behavioral rule of client-server style implies *ActProcessPayment* as the next action of *ActRequestToPay*. As *ActProcessPayment* is also involved in N-tier style, it has thus value of *hasDivertNextAction* property as *ActRequestToLog*, implied by the behavioral rule of N-Tier style. According to the behavioral pattern of N-Tier style, when a payment is requested to *PaymentGateway*, *PaymentGateway* will process the request and call *TransactionLogger* in the upper tier to log a transaction. The behavioral rule of client-server style also implies *ActReturnPayResult* as the next action of *ActProcessPayment*, in case the payment result is returned without logging transaction (for example, when an error occurs during processing a payment). Other behavioral sequence, such as price alert, can also be generated in the same way using the behavioral rule for publish-subscribe style.

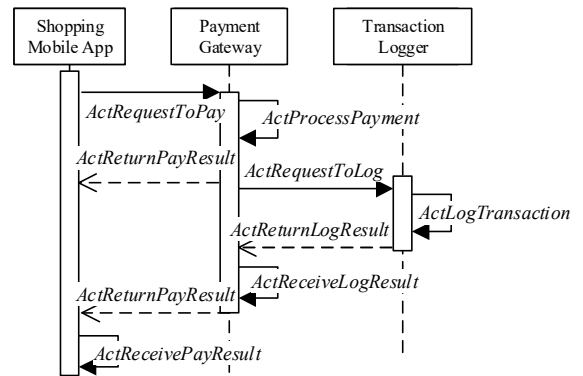


Figure 6 Part of payment process

#### D. Performance Evaluation

We evaluated performance of reasoning process. This evaluation focuses on measuring two parameters that impact the performance of automated verification: 1) number of architectural style applied to software design, and 2) software size that can be reflected by the number of axioms. The more axioms the ontology has, the larger scale a software is. We ran regression testing 50 times on four ontologies that have different parameter values as follows, A contains 0 styles with 144 axioms, B contains 2 styles with 216 axioms, C contains 3 styles with 246 axioms, and D contains 4 styles with 276 axioms.

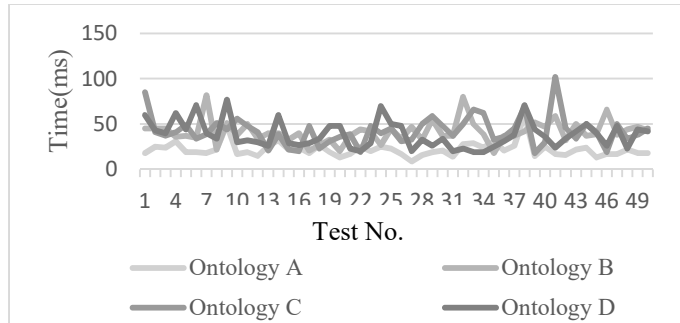


Figure 7 Result of performance testing

This evaluation was carried out using an Intel Core i7-7500U CPU @ 2.7GHz with 8.00 GB Ram computer, and we used HerMiT as the reasoning engine. The time taken to reason ontologies are shown as a graph in Figure 7. The horizontal axis represents the number of time we run reasoning process. According to the test result, average time spending on reasoning is between 20-60 milliseconds and shows insignificant variation between test ontologies. Therefore, we can conclude that our approach supports scalability for complex software architectural design, as the number of applied styles and software size has minor impact on the reasoning performance.

#### IV. CONCLUSION

An architectural design model for a complex software system can be formally specified and verified with our approach. The ontology library includes extensible architectural elements that are defined semantically by OWL, whereas SWRL rules are used to capture dynamic behavior within the design. We demonstrate our approach by creating an ontology instance for an architectural design model. The reasoning engine performs classification that automates verification as follows: 1) architectural consistency is checked against constraints in the ontology, 2) architectural elements and styles are recognized, 3) behavioral sequences are automatically generated according to rules specific to architectural style. We found that complexity level in architectural design has minor impact on the automated verification performance. With automated verification, the user can concentrate on determining whether the design meets requirements, which are the most significant aspect of the software architecture design.

This paper only takes a small step toward our ultimate goal, which we aim to prevent architectural design erosion and lower

maintenance cost. We plan to achieve this by extending proposed approach in this paper and integrate it to the software evolution cycle.

#### REFERENCES

- [1] R. Hilliard, "ISO/IEC/IEEE 42010," [Online]. Available: <http://www.iso-architecture.org/42010/>.
- [2] J. Rumbaugh, I. Jacobson and G. Booch, Unified Modeling Language Reference Manual, The (2nd Edition), Pearson Higher Education, 2004.
- [3] D. Garlan, R. T. Monroe and D. Wile, "Acme: Architectural Description of Component-Based Systems," in *Foundations of Component-Based Systems*, Cambridge University Press, 2000, pp. 47-68.
- [4] M. Ozkaya, "Do the informal & formal software modeling notations satisfy practitioners for software architecture modeling?," *Information and Software Technology*, vol. 95, pp. 15-33, 2018.
- [5] L. Kaur and AshutoshMishra, "Software component and the semantic Web: An in-depth content analysis and integration history," *Journal of Systems and Software*, no. 125, pp. 152-169, 2017.
- [6] H. Kaiya and M. Saeki, "Using Domain Ontology as Domain Knowledge for Requirements Elicitation," in *14th IEEE International Requirements Engineering Conference (RE'06)*, 2006.
- [7] H.-H. Song and Z.-X. Zhang, "Study on Approach of Software Maintenance Based on Ontology Evolution," in *International Conference on Computer Science and Software Engineering*, 2008.
- [8] R. Allen and D. Garlan, "A Formal Basis for Architectural Connection," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 6, no. 3, pp. 213-249, 1997.
- [9] J. S. Kim and D. Garlan, "Analyzing Architectural Styles with Alloy," in *Workshop on the Role of Software Architecture for Testing and Analysis 2006 (ROSATEA06)*, Portland, 2006.
- [10] S. Wong, J. Sun, I. Warren and J. Sun, "A Scalable Approach to Multi-style Architectural Modeling and Verification," in *13th IEEE International Conference on Engineering of Complex Computer Systems (iceccs)*, 2008.
- [11] S. Schröder and M. Riebisch, "Architecture Conformance checking with Description Logic," in *The 11th European Conference on Software Architecture*, 2017.
- [12] J. Simmonds and M. C. Bastarrica, "Description Logics for Consistency Checking of Architectural Features in UML 2.0 Models," 2004.
- [13] E. Yuan, "Towards Ontology-Based Software Architecture Representations," in *IEEE/ACM 1st International Workshop on Establishing the Community-Wide Infrastructure for Architecture-Based Software Engineering (ECASE)*, 2017.
- [14] C. Pahl, S. Giesecke and W. Hasselbring, "Ontology-based modelling of architectural styles," *Information and Software Technology*, vol. 51, no. 12, pp. 1739-1749, 2009.
- [15] J. Sun, H. H. Wang and T. Hu, "Design Software Architecture Models using Ontology," in *International Conference on Software Engineering and Knowledge Engineering (SEKE)*, 2011.
- [16] I. Horrocks, P. F. Patel-Schneider and F. Harmelen, "From SHIQ and RDF to OWL: the making of a Web Ontology Language," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 1, no. 1, pp. 7-26, 2003.
- [17] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Groszof and M. Dean, "SWRL: A Semantic Web Rule Language," 2004. [Online]. Available: <https://www.w3.org/Submission/SWRL/>.
- [18] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, P. Merson, R. Nord and J. Stafford, *Documenting Software Architectures: Views and Beyond (2nd Edition)*, Pearson Education, 2011.
- [19] L. Bass, P. Clements and R. Kazman, *Software Architecture in Practice (2nd Edition)*, Addison-Wesley, 2003.

# Object-oriented Software Modeling with Ontologies Around

## A Survey of Existing Approaches (*SESE*)<sup>1</sup>

Sohaila Baset

Information Management Institute  
University of Neuchatel  
Neuchatel, Switzerland  
sohaila.baset@unine.ch

Kilian Stoffel

Information Management Institute  
University of Neuchatel  
Neuchatel, Switzerland  
kilian.stoffel@unine.ch

**Abstract**—Despite the many integration tools proposed for mapping between OWL ontologies and the object-oriented paradigm, developers are still reluctant to incorporate ontologies into their code repositories. In this paper we survey existing approaches for OWL to OOP mapping trying to identify reasons for this shy adoption of ontologies among conventional software developers. We present a classification of the surveyed approaches and tools based on the characteristics of their resulting artifacts. We finally provide our own reflection for other potential reasons beyond those addressed in the literature.

**Keywords**- Knowledge representation, Object Oriented Programming, Software modeling, Ontologies; OWL; Language transformation.

### I. MOTIVATION

In software development, like in other engineering disciplines, model sharing is always an encouraged practice. It explains the industry's constant pursuit of open standards for modeling languages that allow for seamless incorporation of models pertaining to a certain modeling school into another. Ontological modeling is no exception. After a few predecessors, Ontolingua [1] [2] and DAML+OIL [3], the Web Ontology Language OWL [4] [5] is now the standard language for developing and sharing ontologies in the semantic web as well as many other fields such as the biomedical domain. In the literature, there exist many attempts at integrating ontologies into mainstream development. These attempts vary from loose integration, i.e. accessing ontologies from a programming language, to a more solid transformation from OWL ontologies into software models.

The synergies between ontologies and software models might seem so evident that in many cases an effortless mapping between the two paradigms is taken for granted. This assumption is further supported by a considerable number of proposed development frameworks such as the Ontology Driven Software

Development ODS [6] or the Ontology Oriented Programming [7]. However, shifting a bit from the research state-of-the-art into the circles of conventional software development, we observe a different image than the one painted in research papers. Despite the many integration tools proposed, developers are still reluctant to incorporate ontologies into their code repositories.

In this paper, we try to examine the different reasons behind the modest adoption of ontologies in software modeling. We survey existing integration approaches and we deduce some of their common characteristics with the goal of providing a classification framework that researchers interested in the topic can refer to. We then discuss some of the common challenges reported in the literature before concluding with our own reflection on the current state of OWL to OOP integration.

### II. METHOD AND SCOPE

Given the qualitative nature of literature and the particular topic in question, it is difficult to establish a procedure for automatic classification of existing approaches. When selecting papers to review, we started with three of the earliest papers as seeds for collecting other related papers: OntoJava 2002 [8], Kalynapur 2004 [9] and Knublauch 2004 [6]. Using Google scholar, we harvested all papers that cited at least one of the seed papers. This resulted in around 309 papers. We first grouped similar papers in sets (e.g. papers originating from the same author and/or proposing the same tool) and we chose a representative paper of each group. We then used Google citations metrics as an indicative measure of the impact of a paper (especially for early published papers) to pivot our manual inspection of papers. We dropped irrelevant papers such as papers accentuating the application domain rather than the integration approach. At the end of this process we retained 24 papers that we are surveying in this article. While by no means an exclusive list of all papers in the field, the retained 24 papers give a good indicator on the current state of research.

When reviewing papers, we focused on certain aspects like the extent of integration and the challenges the authors faced

---

<sup>1</sup> DOI reference number: 10.18293/SEKE2018-198

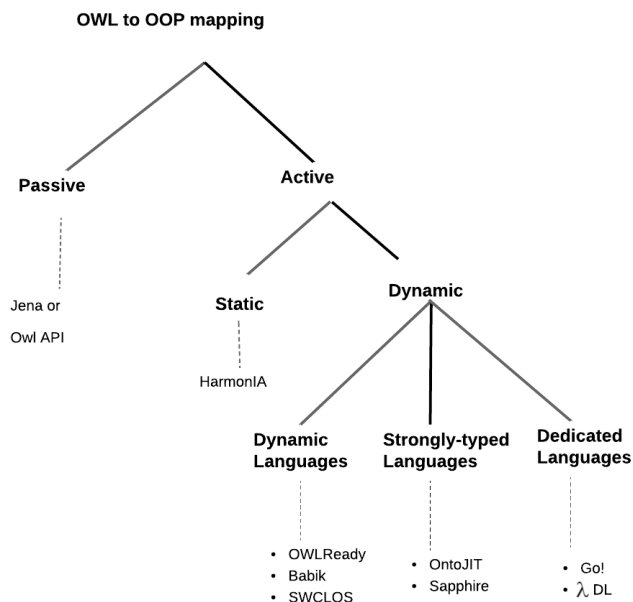


Figure 1. A taxonomy of existing OWL to OOP mapping

rather than focusing on the motivation behind each contribution which was, more or less, common behind most of the reviewed papers.

### III. ONTOLOGY MAPPING

Ontologies, by design, are not intended as standalone software units [10]. They need to be considered in the context of an application that is responsible for accessing and manipulating the concepts they represent. For that reason, it is essential to provide some mapping between the content of an ontology and the application environment in which it resides. Scanning the literature, we can find different terms and expressions denoting this sort of mapping. In this paper, we will be using the term OWL to OOP mapping as an umbrella term for integrating OWL ontologies to the object-oriented paradigm in general. This includes both OOP modeling and programming languages.

We can differentiate between two main approaches for OWL to OOP mapping. In the following sections we define these two approaches, their main characteristics as well as their sub-categories.

#### A. Passive OWL to OOP mapping

Passive OWL to OOP mapping depicts a generic and a rather loose mapping between the content of an ontology and its programming environment. Nevertheless, this approach is more dominant for most applications on the semantic web. In this approach, ontologies are integrated into the mainstream OOP language simply by loading them into memory. Loading is achieved by an ontology loader that transforms the ontology from its syntactic form, e.g. RDF/XML, into an in-memory representation. This in-memory representation can be an

Abstract Syntax Tree (AST) like in the case of OWL API loader [11] [12], or an RDF triple-based structure like the one used in Jena [13]. In either case, the loaded ontology is treated as data and will reside in the data segment of the program allocated memory, hence the name passive approach. This approach is also less challenging as there are no constraints imposed by the target programming language on the kind of data structures used to encapsulate the concepts of the source ontology.

#### B. Active OWL to OOP mapping

In contrast to passive ontology mapping, the active mapping approach will transform an ontology from its serializable format into code statements in the target programming language. The resulting ontology is then executable and belong to the code segment of the allocated memory. This approach is more challenging as it requires finding a native equivalent in the target programming language for each axiom in the source ontology; a requirement that happens to be problematic in many cases as we will demonstrate in the following sections.

Active OWL to OOP mapping can itself be further classified into static or dynamic [14]. In static mapping, the translation, i.e. code generation from owl axioms (concepts, properties and individuals) is done in one shot as a separated prior step. In this case and depending on the target language support for dynamic typing, type checking is mostly limited to compile time. Under this category, we can classify the work done by Kalynapur [9] and Zimmerman [15] to translate OWL ontologies into Java or the .Net ontology compiler by Goldman [7].

Dynamic mapping, on the other hand, will also consider reasoning possibilities about the executable ontology. By dynamically translating OWL axioms at runtime, active OWL to OOP mapping permits certain inference tasks like for example entailing the class of an individual and assigning its type at runtime. Under this approach we can find many tools proposed in the literature and they all provide different degrees of reasoning support. Here again we differentiate between:

1) Tools that have a dynamic language as output and will rely on its dynamic typing features. As examples we can name SWCLOS [16], an OWL processor built on top of Common Lisp Object System (CLOS) and Owlready [14], a python module that beside relying on python interpreter for dynamic typing, can also make use of an external reasoning component (HerMiT reasoner [17]) for further reasoning tasks. Under this category as well, we classify the approach proposed by Babik and Hluchy [18], an approach that uses python metaclasses to represent OWL concepts and perform type checking dynamically.

2) Tools that have a strongly-typed language as output but can still offer some degree of flexibility for type changes at runtime. Examples are the Sapphire tool [19] that relies on the concept of cascade wrapping, or un-wrapping, of proxy objects to handle type changes at runtime and the C# OntoJIT parsing component [20] that exploits a mix of metaprogramming techniques, namely C# reflection, with the dynamic compilation support of CLR, the common language runtime of .Net languages.

3) And finally, some of the proposed tools for dynamic OWL to OOP mapping have gone to the extent of proposing a

Table 1- List of main tools and approaches for OWL to OOP mapping

Year	Tool/Approach	Source language	Target language	Mode	Reasoning support
2002	OntoJava [23]	RDF(s)	Java	Active/Static	None
2003	Goldman [7]	OWL	C#	Active/Static	None
2003	OWL API [11]	OWL	Java	passive	External
2004	Knublauch [6]	OWL	Java	—	—
2004	HarmonIA [9]	OWL	Java	Active/Static	None
2004	Jena [13]	OWL	Java	Passive	External
2005	SWCLOS [16]	OWL	CommonLisp	Active/Dynamic	Limited
2005	RDFReactor [24]	RDF/RDF(S)	Java	Active/Static	None
2006	Atkinson et al. [25]	OWL	UML	—	—
2006	Babik [18]	OWL	Python	Active/Dynamic	Limited
2006	Clark et al. [21]	OWL	Go!	Active/Dynamic	Supported
2007	ActiveRDF [26]	RDF(s)	Ruby on Rails	Active/Static	None
2007	Athanasiadis [27]	RDF / OWL	JavaBeans	Active/Static	—
2007	Owlet [28]	OWL	Java	passive	Supported
2008	Puleston et al. [29]	OWL	Hybrid OWL/Java	—	—
2009	OWL2Java [15]	OWL	Java	Active/Static	None
2011	Sapphire [19]	OWL	Java	Active/Dynamic	Limited
2014	LITEQ [30]	RDF(s)	F#	Active/Static	Limited
2016	OntoJIT [20]	OWL	C#	Active/Dynamic	Limited
2017	Owready [14]	OWL	Python	Active/Dynamic	Indirect
2017	Leinberger et al. [22]	OWL	$\lambda DL$	Active/Dynamic	Supported

dedicated programming language for that purpose such as Go! [21] or the more recent language  $\lambda DL$  [22]. Figure 1. provides a treelike representation of existing OWL to OOP mapping approaches while table 1. provides a somewhat extended list of main tools and approaches.

#### IV. SEMANTIC GAP

The most prominent challenge that is present in active OWL to OOP mapping approaches is the semantic gap between the ontological and object-oriented paradigms. The semantic richness of ontological languages makes it very difficult to find an OOP counterpart to express OWL semantic constructs. One of the most obvious examples is perhaps the different interpretation of class inheritance. OWL, or Description Logics DL in general, has a looser interpretation of a class being the subclass of another. In OWL, the term “`rdfs:subclassOf`” is the manifestation of the subsumption operator of DL. An OWL class is allowed to have many parent classes (named or anonymous) as long as it is subsumed by all these parents. On the other hand, pure OOP languages like Java or C# have a stricter definition of class inheritance, OOP classes are disjoint by design and that is why a class cannot be a subclass of two different parent classes and multiple inheritance is, generally

speaking, not supported. Multiple inheritance is not the only example of the missing semantic equivalence. A similar argument goes for OWL axioms such as “`owl:equivalentClass`”, “`owl:sameAs`” or “`owl:disjointWith`”.

Many of the approaches we surveyed did not attempt at bridging this gap and have instead limited the mapping scope to what is expressible in the target programming languages. On the other hand, some of the active approaches proposed interesting solutions ranging from a “Keep it simple, stupid” approach of adding a meta-layer of code as a substitute for the missing semantics in formal programming languages [20] to a more sophisticated approach of stretching the expressiveness of modeling in Java to that of OWL DL by enforcing some constraints and design patterns [9]: Interfaces with shadow classes for multiple inheritance, special listeners on property accessors, type checking for domain and range properties, etc.

#### V. DISCUSSION

One of the main motivations behind most of the work surveyed in this paper is the difficulty of manipulating ontologies in mainstream software development and the scarcity of options for an ontology programming interface. Nevertheless, as we can see from earlier sections, a retrospective scan on work done in this area revealed a different story. In fact, there exist many options both for accessing or integrating ontologies in an OOP paradigm, yet we still did not witness ontologies spanning new development territories.

Below we try to list some of the potential reasons for this shy adoption of ontologies beyond what has been proposed in the literature:

1) *Too many options*: The real problem developers may have with integrating ontologies is not necessarily the lack of ontology programming interfaces – there exist well many – but rather the lack of consensus on a standardized option. Unlike the case of ontological modeling where OWL is “the language” and Stanford protégé is “the editor”; when it comes to integrating ontologies as software models, there exist many options but none of them has reached a good level of maturity to gain community consensus. As a result, the developer has to go through the hassle of sorting them out before being able to judge on the pertinence of any of these options; a task that is not affordable in most of today’s agile software projects.

2) *Paradigm shift*: Although largely addressed in the literature, the paradigm shift the developer has to go through when integrating ontologies is still present. Providing tool support is one thing, but it takes much more to overcome the conceptual switch behind ontological modeling. Translating ontologies into a program does not change the fact that ontological modeling is explicit and most of the time based on an Open World Assumption OWA in contrast to implicit closed-world modeling in UML and OOP languages.

3) *Legacy projects*: From a pure practical point of view, introducing ontologies to mainstream software projects is especially challenging when there is some legacy code to maintain and respect; which is the case of the majority of software projects in large scale organizations.

4) *Resistance to change*: For most “right-wing” software engineers, adopting ontological approaches, or generally speaking linked data approaches from the semantic web, means, in a way or another, shifting towards a more volatile domain model. A move that may not be perceived as a positive step in the circles of software engineering with strong preferences for tidy and well-engineered domain models. It provokes a lot of philosophical discussions similar to the dynamic vs. static-style of coding in languages that permits the two possibilities. Eventually, such a change may very well be welcome, but just when the right time comes.

## VI. CONCLUSION

In this paper, we surveyed the literature for existing approaches for mapping between OWL ontologies and object-oriented programming paradigms. We presented a classification of the surveyed mapping tools based on the characteristics of their resulting artifacts. We highlighted some of the common challenges encountered in the literature before finally providing our own reflection on why software engineers are still reluctant to incorporate ontologies into their code repositories. Unfortunately, as we mentioned before, most of the tools and prototypes, especially the early ones, did not yield a concrete body of use cases in software industry. It would be interesting therefore to see how the more recent propositions will evolve given the re-awakened interest of the semantic web in the last few years.

## VII. REFERENCES

- [1] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge acquisition*, vol. 5, pp. 199-220, 1993.
- [2] A. Farquhar, R. Fikes and J. Rice, "The ontolingua server: A tool for collaborative ontology construction," *International journal of human-computer studies*, vol. 46, no. 6, pp. 707-727, 1997.
- [3] I. Horrocks and others, "DAML+OIL: A Description Logic for the Semantic Web," *IEEE Data Eng. Bull.*, vol. 25, pp. 4-9, 2002.
- [4] I. Horrocks, P. F. Patel-Schneider and F. Van Harmelen, "From SHIQ and RDF to OWL: The making of a web ontology language," *Web semantics: science, services and agents on the World Wide Web*, vol. 1, pp. 7-26, 2003.
- [5] B. Motik, B. C. Grau, I. Horrocks, Z. Wu, A. Fokoue, C. Lutz and others, "Owl 2 web ontology language: Profiles," *W3C recommendation*, vol. 27, p. 61, 2009.
- [6] H. Knublauch, "Ontology-driven software development in the context of the semantic web: An example scenario with Protege/OWL," in *1st International Workshop on the Model-Driven Semantic Web (MDSW2004)*, 2004.
- [7] N. M. Goldman, "Ontology-oriented programming: static typing for the inconsistent programmer," in *International Semantic Web Conference*, Florida, 2003.
- [8] A. Eberhart, "OntoJava--Applying Mainstream Technology to the Semantic Web," in *Workshop on Semantic Web-based E-Commerce and Rules Markup Languages at the ICEC*, Vienna (Austria), 2001.
- [9] A. Kalyanpur, D. J. Pastor, S. Battle and J. A. Padget, "Automatic Mapping of OWL Ontologies into Java.," in *SEKE*, 2004.
- [10] Wache, Holger and Voegele, Thomas and Visser, Ubbo and Stuckenschmidt, Heiner and Schuster, Gerhard and Neumann and Holger and Hübner, Sebastian, "Ontology-based integration of information-a survey of existing approaches," in *IJCAI-01 workshop: ontologies and information sharing*, Seattle, USA, 2001.
- [11] S. Bechhofer, R. Volz and P. Lord, "Cooking the Semantic Web with the OWL API," in *International Semantic Web Conference*, 2003.
- [12] S. K. Bechhofer and J. J. Carroll, "Parsing owl dl: trees or triples?," in *Proceedings of the 13th international conference on World Wide Web*, 2004.
- [13] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne and K. Wilkinson, "Jena: implementing the semantic web recommendations," in *Proceedings of the 13th international World Wide Web conference on Alternate track papers \& posters*, 2004.
- [14] J.-B. Lamy, "Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies," *Artificial intelligence in medicine*, vol. 80, no. 2017, pp. 11-28, 2017.
- [15] M. Zimmermann, "OWL2Java: a Java Code Generator for OWL, website," -, -, 2009.
- [16] Koide, Seiji and Takeda and Hideaki, "OWL-Full reasoning from an object oriented perspective," in *Asian Semantic Web Conference*, Beijing, 2006.
- [17] Shearer, Rob and Motik, Boris and Horrocks and Ian, "HerMiT: A Highly-Efficient OWL Reasoner.," in *OWLED*, Washington, DC, 2008.
- [18] M. Babik and L. Hluchy, "Deep integration of python with web ontology language," in *In Proceedings of the 2nd Workshop on Scripting for the Semantic Web*, 2006.
- [19] Stevenson, Graeme and Dobson and Simon, "Sapphire: Generating Java runtime artefacts from OWL ontologies," in *International Conference on Advanced Information Systems Engineering*, Gdańsk, Poland, 2011.
- [20] S. Baset and K. Stoffel, "OntoJIT: Parsing Native OWL DL into Executable Ontologies in an Object Oriented Paradigm," in *International Experiences and Directions Workshop on OWL*, 2016.
- [21] Clark, Keith L and McCabe and Frank G, "Ontology oriented programming in Go!," *Applied Intelligence*, vol. 24, no. 3, pp. 189--204, 2006.
- [22] M. Leinberger, R. Lämmel and S. Staab, "The essence of functional programming on semantic data," in *European Symposium on Programming*, Uppsala, Sweden, 2017.
- [23] A. Eberhart, "Automatic generation of java/sql based inference engines from rdf schema and ruleml," in *International Semantic Web Conference*, Sardinia,Italy, 2002.
- [24] M. Völkel and Y. Sure, "RDFReactor-from ontologies to programmatic data access," in *Poster Proceedings of the Fourth International Semantic Web Conference*, Galway,Ireland, 2005.
- [25] C. Atkinson, M. Gutheil and K. Kiko, "On the Relationship of Ontologies and Models.," *WoMM*, vol. 96, pp. 47-60, 2006.
- [26] E. Oren, R. Delbru, S. Gerke, A. Haller and S. Decker, "ActiveRDF: object-oriented semantic web programming," in

*Proceedings of the 16th international conference on World Wide Web*, 2007.

- [27] I. N. Athanasiadis, F. Villa and A.-E. Rizzoli, "Ontologies, JavaBeans and Relational Databases for enabling semantic programming," in *Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International*, 2007.
- [28] A. Poggi, "OWLET: an object-oriented environment for OWL ontology," in *Proceedings of the 11th WSEAS International Conference on Computers*, Agios Nikolaos, Crete Island, Greece, 2007.
- [29] C. Puleston, B. Parsia, J. Cunningham and A. Rector, "Integrating object-oriented and ontological representations: a case study in Java and OWL," in *International Semantic Web Conference*, 2008.
- [30] M. Leinberger, S. Scheglmann, R. Lämmel, S. Staab, M. Thimm and E. Viegas, "Semantic web application development with LITEQ," in *International Semantic Web Conference*, Riva del Garda, Italy, 2014.
- [31] C. Atkinson and T. Kuhne, "Model-driven development: a metamodeling foundation," *IEEE software*, vol. 20, pp. 36-41, 2003.



# Methods for Estimating Agile Software Projects: Systematic Literature Review

Edna Dias Canedo      Dandara Pereira Aranha      Maxwell de Oliveira Cardoso  
Ruyther Parente da Costa      Leticia Lopes Leite

University of Brasília (UnB), P.O. Box 4466 – Brasília-DF, CEP 70910-900, Brazil  
ednacanedo, lleteite@unb.br, dandaraaranha, maxwell.oliveira2@gmail.com, ruyther@me.com

1

## Abstract

*In recent years, agile methods of software development have gained a lot of attention in the field of software engineering. Several estimation techniques have been proposed by several authors and developers in recent years. This paper performs a Systematic Literature Review aiming to identify the most used metrics e/or methods in the development of agile software and the most used size metrics regarding effort estimates, deadlines and costs in a planning of agile software project. The results suggest that Planning Poker is the most popular technique for agile teams in the planning phase, Story Point and Point of Function are the most used metrics in agile projects for estimating size, time, effort, productivity and cost.*

Keywords: Agile Software Development; Software Estimates; Software Metrics; Story Point; Planning Poker.

## 1. Introduction

The concept of agile development was proposed in 2001 by the Agile Team, and then several software development teams and companies recognized and accepted the concept, so its use gradually increased in software projects [1].

The methods for agile software development are a set of practices that have been created by experienced professionals. These methods can be seen as a reaction to the traditional method of development, which emphasizes a rationalized, engineering-based approach stating that problems are fully specifiable, and that, optimal and predictable solutions exist for any kind of problem [2].

Software metrics are often used to understand, control and improve what is done and how it is done in a software development process. Some of the motivations for using metrics are: project planning and estimation, project management and follow-up, quality understanding and business objectives, communication, processes, and improved tools for software development [3]. Thus, the measurement is applied in the process of software development or attributes of a product with the objective of improving it continuously. This technique used throughout the software development

project assists in estimating, quality control, productivity assessment and project control [4].

When estimating effort, duration, and cost of software development projects, software size is a prerequisite. The functional size of the software to be delivered is a solid basis for estimating a software development project, and one of the most common ways of obtaining the functional size of software is through Function Point Analysis (FPA).

Combining the size of a system, in function points for example, with other metrics, allows the accomplishment of estimates for the development project and definition of a plan of actions focused on meeting the goals [5]. The success of agile software development and estimates still continue to challenge current projects and organizations. Despite the importance of metrics and estimates for software development projects, research related to the theme in the context of agile projects still remains scarce, making estimates and planning inefficient and/or imprecise [6].

The main contribution of this paper is to identify which methods and metrics are considered more adequate for estimating agile software development.

## 2. Planning and Estimating Software Projects

The basis for software metrics was established in the 1970s. The first article on the subject was published in 1968 [7]. From these works, more work and interesting results emerged.

Measurements were mainly created to ensure that indicators could be obtained for optimization of production costs, since in the 1990s billions of dollars were spent on software that did not meet the needs of companies at the time [4].

When calculating metrics, you can refine one of the most important tasks of Project Management, which is planning. According to [4], software measurement makes possible a better understanding of the software engineering process and the product (which it produces) to managers and professionals. Using direct and indirect measures, productivity and quality metrics can be defined. It is also possible to identify the estimated effort, cost and time for a project project.

Estimation is one of the main activities of software planning. They provide data that allows you to predict the time required and the costs of the project. It is not possible to prepare a schedule and budget without the use of estimates.

<sup>1</sup>DOI reference number: 10.18293/SEKE2018-031

Estimates are performed based on metrics. With the application of estimates it is possible to collect metrics that allow to predict the amount of people needed, the time required and the costs for the development of the project. "Thus, it becomes important to invest in the implementation of an estimation process" [8]. Size, duration, productivity, and effort metrics are the most commonly used [9].

## 2.1 Planning and Estimating Agile Software Projects

Agile methodologies propose a large set of techniques for estimating and planning projects, especially in terms of non-algorithm-based models [10], [11]. Most agile estimation techniques focus on the use of User Stories (US). User Stories were first introduced by eXtreme Programming (XP) [12] and then popularized by [10].

There are several techniques for estimating agile software projects, however the ones that will be discussed here are: Planning Poker [10], and Ideal Day [13].

### 2.1.1 Planning Poker

Estimates are limited to specific numbers (each number is written in a card). Each member is holding a deck of Planning Poker cards with values like 0, 1, 2, 3, 5, 8, 13, 20, 40 and 100, which is the sequence that are more popular [14], [15]. However, it is not a standard. Each value represents the number of story points [16] which the team estimates. Story points are the most common unit of measure used for estimating the effort involved in completing a user story or resolving an issue.

Story points refer to customer requirements and describe specific functionalities. When the feature has been fully discussed, each estimator privately selects one card to represent his or her estimate. If they are all the same, the functionality receives the estimate; If not, members proposing the highest and lowest estimates explain their points of view and new rounds are played until a consensus is reached. So the team predicts the speed (the number of points they can deliver in an iteration, through historical data, a test iteration or an educated guess) and down the length of the iteration. The number of iterations is obtained by dividing the total number of points by velocity. Likewise, the project duration is calculated by multiplying the number of iterations by size.

### 2.1.2 Ideal Day

Widely used in Agile Software Projects, Ideal Day corresponds to the amount of work that a professional in the area can complete in a day of work [13], [17].

The speed is calculated from the number of hours the team spends to implement a work equivalent to an Ideal Day [18]. If the item goes through a day of work, it is suggested to decompose this item into smaller items that can be implemented in just one day. According to [18], calculate

the estimated days use the following formula:

$$DE = \frac{IED}{1 - IEDREAL}$$

At where: **DE**: represents the estimated number of days to complete the task; **IED**: represents the time needed to implement the item. This deadline is defined by the team; **IEDREAL**: represents the percentage that indicates the estimate of how much time of day the developer will be dedicated to the implementation of the item.

## 3 Systematic Literature Review

The objective of the systematic review [19] was to identify scientific works that present methodologies and metrics solutions for agile software development methodologies to identify: Common project measurement and control practices; Size metrics used; Research trends in agile development; and Open questions and research topics related to improving the estimates of agile development projects. The research was performed in four steps:

- **Step 1:** Perform automatic search and manual in order to identify a preliminary list of studies. Duplicate studies were discarded. The StArt tool was used as support for the documentation, extraction and structuring of the primary studies.
- **Step 2:** Identification of potentially relevant studies, based on title and abstract analysis, discarding studies that are clearly irrelevant to the research. If there was any doubt about a study regarding its inclusion or exclusion, the next step was to check whether the study was relevant or not.
- **Step 3:** Selected studies in previous steps were reviewed by reading the introduction, methodology section and conclusion and applying the inclusion and exclusion criteria. If reading the above items was not enough to make a firm decision, the study was read in its entirety.
- **Step 4:** thus, a list of primary studies was obtained and subsequently subjected to critical examination using the criteria established.

In order to characterize the methods for estimating agile software projects, two different questions were formulated:

- **RQ.1.** What are the metrics and methods used to make effort estimates, deadlines, and costs for agile software project planning?
- **RQ.2.** Function point metrics can be used to make estimates of effort, deadlines, and costs for agile software planning? If so, is it the most appropriate estimate?

The search string definition was based on the population, intervention, comparison and the result [20]. **Population:** The population is the agile software development. To search the population, the keyword "Agile Software Development" were used. **Intervention:** The intervention is to estimate effort, cost, time and to identify how the Function Points Analysis is performed in agile software projects. Therefore, the terms: metric, prediction, estimation and function points were considered. **Comparison:** The focus of the study was not limited to comparative studies. Therefore, the comparison was not considered in the research strategy. **Result:** The main focus is for the research for estimation metrics and methods that are used in scientific studies and/or industry, as reported by researchers. Thus, the research contained words like empirical, validation, evaluation, etc.

### 3.1 Study Selection Criteria

The following selection criteria were defined for the selection of primary studies:

1. The year of publication of the studies should be between 2007 and 2018. However, classical sources with definitions (books with classical concepts or pioneering articles) were also considered.
2. Works that propose methods or metrics to realize estimations and agile software projects planning.
3. The work has reference to software metrics and function points in agile development.

As criterion of exclusion of the studies was considered the non-fulfillment of some of the inclusion criteria, as well as:

1. Works that do not propose methods and metrics to realize agile software projects estimations.
2. Duplicated works or published as Short Paper.

### 3.2 Results

The Systematic Literature Review (SLR) is a form of secondary study that aims to identify and analyze the relevant research for a given research question [21]. As a result, a total of **291 papers** were found. After applying the inclusion criteria, **27 primary studies** were classified.

In summary, in the first stage, where the search string was inserted in the digital libraries, a total of 291 articles were obtained. In the second stage, the results were filtered by reading their title, abstract, and keywords, and a total of 189 articles were selected. In the fourth stage, another filter was applied, through the complete reading of the article, with the object of finding the answers to the research questions. After this stage, 22 articles had been chosen to answer the research questions.

In addition to the **22 studies** selected in the automatic search, **5** other studies were selected through manual search. After finishing the 3 analysis stages, **27 articles** had been selected for data extraction. Table 1 shows all the selected papers.

The results of the SLR according to each defined research question were: **RQ.1:** The most appropriate estimation model for agile projects is **Story Point** [36]. Thereby, it has been realized that **Planning Poker** is one of the most popular techniques for agile teams in planning and estimating effort before starting each iteration.

The size estimation techniques are grouped together with the effort estimation techniques, because within the context of agile development, effort estimation is often derived from Velocity estimates and calculations [36] and [35]. The Expert Judgment and Use Case Points technique are also frequently used estimation techniques in the context of agile software development. The techniques of estimating effort in the agile context and its occurrence in the selected works: Expert Opinion - 8; Estimate based on model (COCOMO, etc.) - 5; Planning Poker - 11. Use-Case Points - 3; Custom Templates - 2; Number of Lines of Code - 4; Fuzzy based Framework for Estimation - 1.

Through the number of Story Points and Velocity of the development team you can calculate the term of development of a certain functionality [36]. For example, if the total number of Story Points for the desired functionalities is 200 and the Velocity of the team is 20, then it can be concluded that the team will need 20 iterations to complete the development of the respective functionalities. However, the User Story Point (USP) is not objective and can not define a standard practice for estimating the size and complexity of the software [25].

Innovative work has been identified in the area of agile project estimates. The paper [27] for example, proposes a structure that depends on the use of fuzzy logic and aims to help in the production of accurate estimates. In the paper presented by [23] it is proposed to modify the Use-Case-Points (UCP) method to make it suitable for agile software development by naming this new version for interactive UCP version (iUCP).

The work proposed by [38] presents a proposed model of effort prediction caused by changes in software requirements. The model integrates the analysis of the impacts of the changes with the COCOMO effort estimation model to improve the precision of the effort estimates from changes in agile software development projects.

The Bayesian Network model was proposed to help agile project managers estimate project effort. It is a graphical model that describes the probabilistic relations between the related variables [47].

According to the works analyzed, Story Points is the most used size metric to carry out the estimations of agile

Table 1: Selected Papers for Data Extraction

Number	Title	Author
1	Measuring and predicting software productivity: A systematic map and review	[22]
2	iUCP: Estimating Interactive-Software Project Size with Enhanced Use-Case Points	[23]
3	Enhancing Quality in Scrum Software Projects	[24]
4	On the Current Measurement Practices in Agile Software Development	[25]
5	Survey on agile metrics and their inter-relationship with other traditional development metrics	[26]
6	Towards a Fuzzy based Framework for Effort Estimation in Agile Software Development	[27]
7	Function Points, Use Case Points, Story Points: Observations From a Case Study.	[28]
8	Estimating, planning and managing Agile Web projects under a value-based perspective	[29]
9	Effort, duration and cost estimation in agile software development	[30]
10	Cost and effort estimation in agile software development	[31]
11	How is effort estimated in agile software development projects?	[32]
12	Identification of inaccurate effort estimates in agile software development	[33]
13	Effort estimation in Agile software development: A survey on the state of the practice	[34]
14	Effort estimation in Agile Software Development: A systematic literature review	[35]
15	Model-based dynamic cost estimation and tracking method for agile software development	[36]
16	NORPLAN: Non-functional requirements planning for agile processes	[37]
17	Predicting effort for requirement changes during software development	[38]
18	Method for personal capability assessment in agile teams using personal points	[39]
19	Understanding and improving effort estimation in agile software development	[40]
20	Agile metrics for a university software engineering course	[41]
21	Applying Software Metrics with Scrum	[42]
22	On using planning poker for estimating user stories	[15]
23	Efficiency factor and risk factor based user case point test effort estimation model	[43]
24	Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly Iterative or agile software development processes	[44]
25	Does the use of Fibonacci numbers in planning poker affect effort estimates?	[45]
26	Improving the user story Agile technique using the INVEST criteria	[46]
27	Bayesian network model for task effort estimation in agile software development	[47]

projects, however it can also be verified that Function Points are still widely used, often being combined with other metrics.

**RQ.2:** Function point metrics can be used to make estimates. A dynamic cost estimating model for agile software projects can be used, namely, it has the ability to adapt during the development process and with changes in requirements [36]. This model adopts function points such as estimation metric and a tracking algorithm, the project time estimating the size of the functionalities using function points and calculating the cost to derive the duration of the project.

To derive the project plan, three procedures are performed. 1. The project team calculates the function points of the desired functionalities; 2. The estimation model, which is composed of the remaining function points, is generated; 3. The project team develops a plan for the release, iteration, and delivery using estimated cost metrics, such as: people per month and number of lines of code.

Although some papers suggest adaptations of traditional models for estimating and measuring agile software projects, one of the weaknesses of agile communities may still be the failure to estimate and measure projects using standard metrics such as function points that are largely known and used within the industry, but which cover only the functional requirements (Jones 1998), different from Story Points, for example, that do not correspond to a software size and not even the actual effort, but to estimates (not measurement), and that cover the functional and non-functional requirements.

Some selected studies also state that Point of Function is not suitable for estimating agile projects because of their granularity and insufficient support for feedback and requirements change. Therefore, they firmly support the idea that companies that work in a traditional or agile way collect traditional measures of size (such as Function Points) for portfolio management, project management and benchmarking; and that companies working according to an agile method also do this, in addition to collecting size measures in an agile-size metric (such as Story Points) for estimation purposes.

#### 4. Conclusions and Future Work

Through the execution of this work, it was possible to perceive the relevance of size, effort, cost and time estimates in the context of agile software development, and therefore, methods and estimation metrics have been increasingly discussed in scholarly works that seek the best and most precise metrics used in a given agile context.

Through the Systematic Review of Literature (SLR) it was possible to identify the methods and the main size metrics used in estimations in the context of agile software development. Among the most used techniques are: Planning Poker, Expert Opinion and Function Point Analysis. The most used metrics for estimates are Story Points and Func-

tion Points.

The primary studies identified in the SLR showed that the methods and the metrics for estimates are mostly applied to a given context of agile development with adaptations in order to fit the project in question. Thus, it can also be concluded that the estimation metrics must always be adapted to fit the project context, since each project will have its own characteristics which influence the result of the estimates.

The case study showed that the estimates using the Function Points metric had deviation percentages of the actual values from the estimated values lower than the estimates made using the Story Points metric. This is due to the fact that the team has a lot of experience with the use of estimates with the Function Points metric, so the initial values of the productivity estimates and the hour value of 1 FP were much closer to the actual value, which consequently made the rest of the estimates more precise.

However, it is necessary to perform other experiments using real projects of different characteristics to be able to affirm with certainty that the Function Points metric is more accurate than the Story Points metric in agile development projects.

#### References

- [1] J. Li, "Agile software development," 2010.
- [2] T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review," *Information and software technology*, vol. 50, no. 9-10, pp. 833-859, 2008.
- [3] K. Pulford, A. Kuntzmann-Combelles, and S. Shirlaw, *A quantitative approach to software management: the AMI handbook*. Addison-Wesley Longman Publishing Co., Inc., 1995.
- [4] R. S. Pressman, *Software engineering: a practitioner's approach*. Palgrave Macmillan, 2005.
- [5] R. Dias, "Análise por pontos de função: uma técnica para dimensionamento de sistemas de informação," *Revista Eletrônica de Sistemas de Informação ISSN 1677-3071, volume=2, number=2, year=2003*.
- [6] B. Kitchenham, "Whats up with software metrics?—a preliminary mapping study," *Journal of systems and software*, vol. 83, no. 1, pp. 37-51, 2010.
- [7] R. J. Rubey and R. D. Hartwick, "Quantitative measurement of program quality," in *Proceedings of the 1968 23rd ACM national conference*. ACM, 1968, pp. 671-677.
- [8] C. Hazan, "Análise de pontos de função: Uma aplicação nas estimativas de tamanho de projetos de software," *Engenharia de Software Magazine, Edição*, vol. 2, pp. 25-31, 2008.
- [9] P. Bourque, R. E. Fairley et al., *Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0*. IEEE Computer Society Press, 2014.
- [10] M. Cohn, *Agile estimating and planning*. Pearson Education, 2005.
- [11] J. Highsmith, *Agile project management: creating innovative products*. Pearson Education, 2009.
- [12] K. Beck, *Extreme programming explained: embrace change*. addison-wesley professional, 2000.

- [13] F. Alves and M. Fonseca, "Ideal day e priorizao: Mtodos geis no planejamento," 2008.
- [14] N. C. Haugen, "An empirical study of using planning poker for user story estimation," in *Agile Conference, 2006*. IEEE, 2006, pp. 9–pp.
- [15] V. Mahnič and T. Hovelja, "On using planning poker for estimating user stories," *Journal of Systems and Software*, vol. 85, no. 9, pp. 2086–2095, 2012.
- [16] M. Choetkiertikul, H. K. Dam, T. Tran, T. T. M. Pham, A. Ghose, and T. Menzies, "A deep learning model for estimating story points," *IEEE Transactions on Software Engineering*, 2018.
- [17] M. Usman, "Improving expert estimation of software development effort in agile contexts," Ph.D. dissertation, Blekinge Tekniska Högskola, 2018.
- [18] J. C. C. Martins, "Tcnicas para gerenciamento de projetos de software," 2001.
- [19] B. Kitchenham, "Procedures for performing systematic reviews," *Keele, UK, Keele University*, vol. 33, no. 2004, pp. 1–26, 2004.
- [20] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," *ech. rep. EBSE 2007-001*, 2007.
- [21] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering—a systematic literature review," *Information and software technology*, vol. 51, no. 1, pp. 7–15, 2009.
- [22] K. Petersen, "Measuring and predicting software productivity: A systematic map and review," *Information and Software Technology*, vol. 53, no. 4, pp. 317–343, 2011.
- [23] N. Nunes, L. Constantine, and R. Kazman, "Iucp: Estimating interactive-software project size with enhanced use-case points," *IEEE software*, vol. 28, no. 4, pp. 64–73, 2011.
- [24] A. R. Mukker, A. K. Mishra, and L. Singh, "Enhancing quality in scrum software projects," *International Journal of Science and Research (IJSR)*, vol. 3, no. 4, pp. 682–688, 2014.
- [25] T. Javdani, H. Zulzalil, A. A. A. Ghani, A. B. M. Sultan, and R. M. Parizi, "On the current measurement practices in agile software development," *arXiv preprint arXiv:1301.5964*, 2013.
- [26] S. Misra and M. Omorodion, "Survey on agile metrics and their inter-relationship with other traditional development metrics," *ACM SIGSOFT Software Engineering Notes*, vol. 36, no. 6, pp. 1–3, 2011.
- [27] A. T. Raslan, N. R. Darwish, and H. A. Hefny, "Towards a fuzzy based framework for effort estimation in agile software development," *International Journal of Computer Science and Information Security*, vol. 13, no. 1, p. 37, 2015.
- [28] J. Schofield, A. Armentrout, and R. Trujillo, "Function points, use case points, story points: Observations from a case study," *CrossTalk*, vol. 26, no. 3, pp. 23–27, 2013.
- [29] C. Torrecilla-Salinas, J. Sedeño, M. Escalona, and M. Mejías, "Estimating, planning and managing agile web development projects under a value-based perspective," *Information and Software Technology*, vol. 61, pp. 124–144, 2015.
- [30] M. Owais and R. Ramakishore, "Effort, duration and cost estimation in agile software development," in *Contemporary Computing (IC3), 2016 Ninth International Conference on*. IEEE, 2016, pp. 1–5.
- [31] R. Popli and N. Chauhan, "Cost and effort estimation in agile software development," in *Optimization, Reliability, and Information Technology (ICROIT)*. IEEE, 2014, pp. 57–61.
- [32] T. Schweighofer, A. Kline, L. Pavlic, and M. Hericko, "How is effort estimated in agile software development projects?" in *SQAMIA*, 2016, pp. 73–80.
- [33] F. Raith, I. Richter, R. Lindermeier, and G. Klinker, "Identification of inaccurate effort estimates in agile software development," in *Software Engineering Conference (APSEC), 2013 20th Asia-Pacific*, vol. 2. IEEE, 2013, pp. 67–72.
- [34] M. Usman, E. Mendes, and J. Börstler, "Effort estimation in agile software development: a survey on the state of the practice," in *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering*. ACM, 2015, p. 12.
- [35] M. Usman, E. Mendes, F. Weidt, and R. Britto, "Effort estimation in agile software development: a systematic literature review," in *Proceedings of the 10th International Conference on Predictive Models in Software Engineering*. ACM, 2014, pp. 82–91.
- [36] S. Kang, O. Choi, and J. Baik, "Model-based dynamic cost estimation and tracking method for agile software development," in *Computer and Information Science (ICIS), 2010 IEEE/ACIS 9th International Conference on*. IEEE, 2010, pp. 743–748.
- [37] W. M. Farid and F. J. Mitropoulos, "Norplan: Non-functional requirements planning for agile processes," in *Southeastcon, 2013 Proceedings of IEEE*. IEEE, 2013, pp. 1–8.
- [38] S. Basri, N. Kama, F. Haneem, and S. A. Ismail, "Predicting effort for requirement changes during software development," in *Proceedings of the Seventh Symposium on Information and Communication Technology*. ACM, 2016, pp. 380–387.
- [39] S. Čelar, M. Turić, and L. Vicković, "Method for personal capability assessment in agile teams using personal points," in *Telecommunications Forum Telfor (TELFOR)*. IEEE, 2014.
- [40] B. Tanveer, L. Guzmán, and U. M. Engel, "Understanding and improving effort estimation in agile software development: an industrial case study," in *Software and System Processes (ICSSP), 2016 IEEE/ACM International Conference on*. IEEE, 2016, pp. 41–50.
- [41] C. Matthies, T. Kowark, M. Uflacker, and H. Plattner, "Agile metrics for a university software engineering course," in *Frontiers in Education Conference (FIE), 2016 IEEE*. IEEE, 2016, pp. 1–5.
- [42] M. L. Gamba and A. C. G. Barbosa, "Engenharia de software-aplicação de métricas de software com scrum," *Anais SULCOMP*, vol. 5, 2010.
- [43] A. W. M. M. Parvez, "Efficiency factor and risk factor based user case point test effort estimation model compatible with agile software development," in *Information Technology and Electrical Engineering (ICITEE), 2013 International Conference on*. IEEE, 2013.
- [44] H. M. Olague, L. H. Eitzkorn, S. Gholston, and S. Quattlebaum, "Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly iterative or agile software development processes," *IEEE Transactions on Software Engineering*, vol. 33, no. 6, pp. 402–419, 2007.
- [45] R. Tamrakar and M. Jørgensen, "Does the use of fibonacci numbers in planning poker affect effort estimates?" 2012.
- [46] L. Buglione and A. Abran, "Improving the user story agile technique using the invest criteria," in *Software Measurement and the 2013 Eighth International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2013 Joint Conference of the 23rd International Workshop on*. IEEE, 2013, pp. 49–53.
- [47] S. Dragicevic, S. Celar, and M. Turic, "Bayesian network model for task effort estimation in agile software development," *Journal of Systems and Software*, vol. 127, pp. 109–119, 2017.

# Investigating gaps on Agile Improvement Solutions and their successful adoption in industry projects - A systematic literature review

Arthur Freire\*, André Meireles†, Gleyser Guimarães‡, Mirko Perkusich§,  
Raissa da Silva\*, Kyller Gorgônio\*, Angelo Perkusich\* and Hyggo Almeida\*

\* Intelligent Software Engineering (ISE) Group, Federal University of Campina Grande

Emails: {arthurfreire, raissasilva}@copin.ufcg.edu.br,

{kyller, hyggo}@dsc.ufcg.edu.br, and perkusich@dee.ufcg.edu.br

† Federal University of Ceará — Email: andre@crateus.ufc.br

‡ Federal University of Campina Grande — Email: gleyser@copin.ufcg.edu.br

§ Federal Institute of Paraiba — Email: mirko.perkusich@ifpb.edu.br

## Abstract—

**Background:** The focus of Agile software development (ASD) is different than plan-driven development, requiring new software process improvement (SPI) paradigms.

**Objective:** To identify and synthesize the possible gaps of Agile improvement solutions (AIS) given their focus on people factors, report of successful adoption in industry projects and availability of tool support.

**Method:** We applied a Systematic Literature Review of studies published up to (and including) 2017 through backward and forward snowballing given a start set.

**Results:** In total, we evaluated 55 papers, of which 44 included AIS and the main findings are: 1) 26 consider teamwork factors; 2) 21 were applied on industry; 3) 10 out of these 21 presented evidence of increase in company performance; and 4) 19 of the solutions are for the purpose of adoption, 18 for assessment and 8 are maturity models.

**Conclusion:** The main implication for this research is a need for more and better empirical studies documenting and evaluating AIS. For the industry, the review provides a map of current AIS approaches and can be used as a starting point to adopt agile SPI.

**Index Terms—**Agile, Systematic Literature Review, Maturity Model, Assessment, Adoption, Tailoring

## I. INTRODUCTION

Agile software development (ASD) methods have gained much attention throughout the last years due to the need of adaptability and flexibility in software projects [1]. ASD is considered an alternative approach to plan-driven development because it promises some benefits if compared to these approaches such as delivery of business value in short iterations. Moreover, it focuses more on people and their interactions instead of processes and tools.

According to Salo & Abrahamsson [2], since the focus of agile methods is different than plan-driven development processes, there is a need of new software process improvement (SPI) (i.e., initiatives that can be used in software organizations to mature their operations [3]).

One possibility to use ASD methods is by tailoring a particular method to fit a given context. In some contexts, ASD is implemented along with SPI initiatives based on the ISO/IEC 33001 international standard or the Capability Maturity Model Integration for Development (CMMI-DEV) [4], [5], [6], [7], [8], [9], [10]. However, these models require that processes be formally defined and controlled, which is not usual practice in ASD [11].

Fontana et al. [11] hypothesize that, given the current definition of maturity defined by CMMI-DEV, agile teams could never achieve maturity without shifting their focus from process to people. Given this, they proposed a definition for ASD maturity that includes not only the definition and improvement of processes, but also people factors such as collaboration, communication, commitment, care, sharing and self-organization.

We defined a terminology for some terms that we use throughout the paper to ease the reading:

- Maturity: related to maturity itself, agility or improvement;
- Maturity Model: according to Kohlegger et al. [12], maturity models are instruments used to rate capabilities, and based on this rating, initiatives can be implemented to improve the maturity of an element - a person, an object or a social system. In this paper, maturity model is a SPI solution or model that describes a set of levels or steps to allow maturing in a given software process;
- Adoption: Related to ASD adoption itself, tailoring, customization, adaptation, transition, etc.;
- Agile Improvement Solution (AIS): describes a solution (i.e., model, process, approach, framework, method, etc.) that is focused (i) on the definition of agile maturity levels or a maturity model itself; (ii) on the maturity assessment of a given company, team, etc.; (iii) on the adoption of ASD by a company, team, etc. For a solution to be considered an AIS, it must not be combined with traditional approaches.

Following the definition of ASD maturity stated by Fontana et al. [11], researchers have proposed maturity models and agile maturity assessment solutions. According to Buglione [13], to be suitable for agile environments, agile maturity models should be inexpensive, fast and easy to understand, should produce short management reports, and should provide relevant drivers and best practices for a road map to maturity.

The goal of this study is to report the state of the art of AIS. We focus on the following questions: (i) do they consider agile teamwork factors?; (ii) were they applied to industry projects?; (iii) were they evaluated in terms of benefits of adoption such as increase on productivity, value delivered or cost reduction?; and (iv), do they provide tool support?

For this purpose, we performed a systematic literature reviewed following the guidelines proposed by Kitchenham and Charters [14] and Wohlin [15]. In this paper, we detail our study and also point the gaps and future directions for research in AIS.

The remaining of the paper is structured as follows: in Section II, we discuss previous literature reviews that are related to this study; in Section III, we describe the protocol of our review process; in Section IV, we present our findings; in Section V, we discuss the results; the threats to validity are presented in Section VI; and in Section VII, we present our conclusions and future works

## II. RELATED REVIEWS

Some literature reviews were conducted on topics related to this review. We summarize them as follows.

Henriques & Tanner [16], performed a systematic review of agile and maturity model research. The authors identified 39 relevant papers to their research, which aimed to identify the trends in research concerned with agile methods in the context of agile maturity models. They concluded that agile and CMMI can coexist when agile is introduced into already highly mature environments or when the primary goal is focused solely on the delivery. However, they concluded that if higher levels of CMMI maturity is the goal, agile cannot be used without being supplemented with other non-agile practices.

Silva et al. [17] evaluated, synthesized, and presented results on the use of ASD with CMMI. From 81 included studies, they concluded that agile methodologies can be used by companies to reduce efforts in getting to levels 2 and 3 of CMMI, there even being reports of applying agile practices that led to achieving level 5. However, as Henriques & Tanner [16], they concluded that agile practices alone could not achieve level 5, being necessary to resort to additional practices.

Martinez et al. [18] identified the problems related to agile adoption in a systematic review, and stated that the findings of this review would be the basis to propose a framework to support the agile adoption. 27 papers were selected on their research after being filtered by the selection criteria. The authors categorized the problems in four groups: people, process, project, and company. Considering their findings, the literature reports more problems related to *people*.

Dikert et al. [19] conducted a systematic literature review of industrial large-scale agile transformations, and analyzed 52 papers describing 42 organizations. Part of the results of this review indicated that large-scale agile seems to be harder to implement than people expect, as companies complain about not finding enough guidance in the literature. The most challenge topics are integrating non-development functions, resistance to change, and requirements engineering.

Unlike the previously mentioned works, which cover the coexistence of agile and CMMI and problems related to agile adoption, we focus on the fidelity of AIS to the Agile Manifesto [20] in regards to teamwork factors (six out of the twelve principles are related to the individuals involved in the product development), their validity in real projects, as well as if they provide tool support to facilitate their usage by practitioners.

## III. METHODOLOGY

In this research, we performed a Systematic Literature Review (SLR) following the guidelines presented in [14].

### A. Research Questions

As previously mentioned, the main goal of this research is to evaluate the fidelity of AIS to the Agile Manifesto [20] in regards to teamwork factors, their validity in real projects, as well as if they provide tool support to facilitate their usage by practitioners. Therefore, we formulated the following research questions (RQ):

**RQ1:** What is the percentage of AIS that take teamwork factors in consideration?

**RQ2:** What is the percentage of AIS that were validated in real projects?

**RQ3:** Within the AIS validated in real projects, what is the percentage of them that present traces of increased efficiency/performance/productivity, or even cost reduction?

**RQ4:** What is the percentage of AIS that provide tool support?

### B. Identifying Primary Studies

In order to identify the relevant primary studies for our research questions, we decided to use the snowballing guidelines defined in [15]. The start set of primary studies necessary for the snowballing was defined based on relevant studies identified in [16], and previously known studies by the authors. With these two sources, we had 53 studies - 39 from [16] and 14 previously know - to analyze according to the selection criteria we defined. From these 53 studies, we identified 22 relevant studies that composed the start set of the snowballing.

After having the start set defined, we started performing the snowballing iterations by executing the backward and forward snowballing steps, and applying the inclusion and exclusion criteria, as well as checking for duplicates until the moment that no relevant studies were found.

A paper is considered irrelevant in case it is (i) not related to ASD only, which means that we are not interested



in studies that propose the usage of agile and traditional software development approaches (e.g., coexistence of agile and CMMI); (ii) not written in english; (iii) published in non peer reviewed publication channel such as books, thesis or dissertations, tutorials, keynotes, etc.; (iv) secondary study; and (v) duplicated. On the other hand, a relevant paper must present a solution that fits the description of AIS and is focused on ASD only.

Every paper identified in this review was evaluated based on the sequence of the four steps described below:

- 1) Initial Evaluation: Each paper found in the backward and forward snowballing steps were initially evaluated based on their titles, abstracts, and keywords. The goal of this step is to exclude every paper that does not fit this review context, and to not discard possible relevant papers;
- 2) Check of Duplicates: The goal of this step is to avoid rework. If a given paper is possibly relevant according to the *Initial Evaluation*, the reviewer needs to check in the database if that paper was already evaluated. The paper only goes to the next step of evaluation if it was not already in the database;
- 3) Superficial Evaluation: In this step, the reviewer needs to superficially check the paper in order to identify a solution that fits the review context. In case the paper presents no solution, the paper should be discarded. However, if it is not clear to the reviewer that the paper is relevant, it should not be discarded. Each paper is evaluated by two random reviewers in this step by following the procedure presented by Ali et al. [21];
- 4) Data Extraction: The goal of this step is to minutely evaluate the paper and extract the relevant information to answer the research questions. It is also possible to find irrelevant papers in this step because a deeper analysis is made if compared to the one made in the previous step. Two random reviewers are necessary to execute this step. The first reviewer is called data extractor and the second reviewer is called data checker. The role of the data extractor is to extract the relevant data from the paper if it is considered relevant. The data checker needs to check if every piece of data that was extracted from the paper is correct, and even if the paper is relevant or not depending on the data extractor's judgment. In case they do not reach a common sense, a third reviewer is invited to help them.

According to Ali et al. [21], there are six categories of agreement or disagreement between the reviewers, as shown in Figure 1. These categories were introduced on the third and fourth steps.

Categories A or B mean that at least one reviewer evaluated the paper as relevant and it is included. Category B occurs when one reviewer is uncertain about the relevance of the paper. To minimize the risk of discarding a significant study, the paper is included in the pool for the next step. Afterwards, in the next step all doubts about the paper's relevance are

		Reviewer 2		
		Relevant	Uncertain	Irrelevant
Reviewer 1	Relevant	A	B	D
	Uncertain	B	C	E
	Irrelevant	D	E	F

Fig. 1. Categories of agreement or disagreement.

clarified with a further evaluation. Category C means that no concrete decision was made by any of the two reviewers and further investigation is needed. In this case, a third reviewer needs to evaluate the paper. If the third reviewer evaluates the paper as irrelevant, it is discarded; otherwise, it is included in the pool for the next step

Categories D and E are results from disagreement and the reviewers are asked to discuss what reasons led them to their respective decisions. After that, a consensus is expected and a new category (A, C or F) classification must be done. Papers in category F are excluded, as both reviewers agreed on their irrelevance.

### C. Extracted Data

We used a spreadsheet editor to record information. For each paper, we extracted general information such as year of publication, and type of article, as well as data related to the RQ. The following data were extracted from the papers:

- (i) year of publication;
- (ii) type of article (i.e., journal, conference, or workshop);
- (iii) validation context (i.e., none, academic, industrial, or both academic and industrial);
- (iv) agile method domain;
- (v) tool support;
- (vi) considers teamwork factors
- (vii) category (i.e., assessment, maturity model, adoption);
- (viii) traces of increased efficiency/performance.

Some AIS are approached in more than one paper. For example: an AIS was introduced in a paper, and its validation is described in another paper. For this reason, the extraction of the data regarding the RQ, which is related to the AIS themselves and not just papers, was made by taking in consideration the set of papers that is related to a given AIS. Moreover, if an author uses an existing AIS as basis for another AIS by performing small modifications, it is considered a new AIS.

## IV. RESULTS

In this section, we present the results for the SLR process and for the RQ.

From the 22 papers that composed the start set, we identified 16 relevant papers by executing the snowballing steps. We used these 16 papers as the seed set for a new snowballing iteration and identified 11 papers. For the next iteration, we found 5 papers. During the last iteration, we found an additional

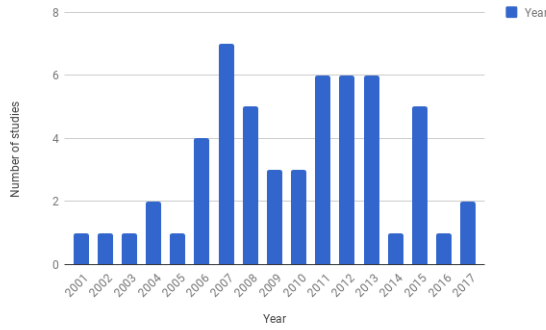


Fig. 2. Number of papers per year.

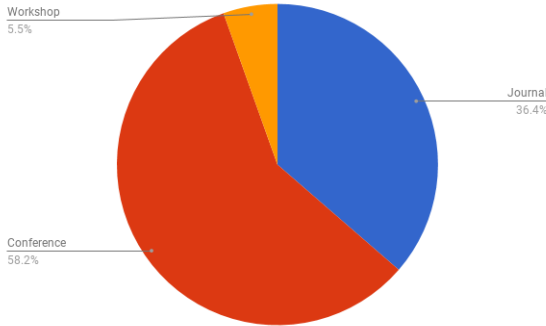


Fig. 3. Percentage of papers per type of publication channel.

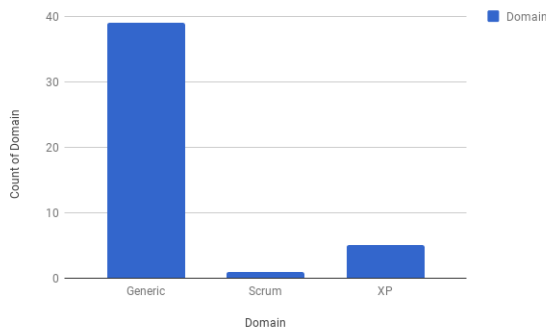


Fig. 4. Percentage of agile methods approached in the AIS.

paper. In total, we selected 55 papers. Due to space limitations, we present data extracted from 18 papers in Table I related to the RQ. The complete list of papers with more detailed information is available at <https://goo.gl/FYoZYg>.

In Figure 2 we present the amount of papers per year. In Figure 3 we show the percentage of papers per type of publication channel.

We identified 45 AIS by analyzing the papers and authors. In Figure 4, we present the distribution of the agile methods domain approached by the AIS. In Figure 5, we present the distribution of the AIS categories.

The percentage of AIS that consider teamwork factors in their approach is 60%, which corresponds to 27 of the 45

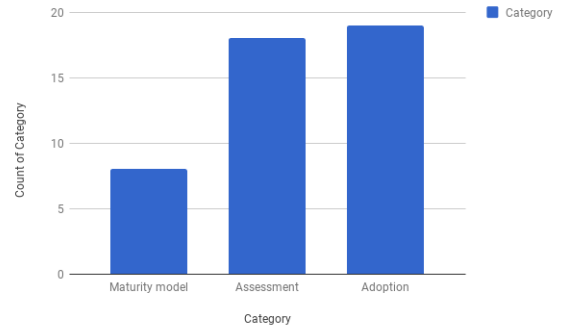


Fig. 5. Distribution of the AIS categories.

identified AIS. 21 solutions were validated in industry environments, which corresponds to 46.7% of the total. From these 21 AIS validated in industrial environments, only 10 presented benefits of their adoption such as increase on productivity, value delivered or cost reduction, which corresponds to 45.5%. Finally, in regards to **RQ4**, only 28.9% of the identified AIS provide tool support, which corresponds to 13 AIS. These percentages, which are related to the RQ, are also presented in Table II.

## V. DISCUSSION

In this section, we discuss the results regarding the four research questions (see Section III-A).

The first paper identified was published in 2001, but we also identified two papers published in 2017. The period with the highest rate of publications was between 2006 and 2013. However, over the last five years, there was a significant number of papers published. We believe that these new solutions are being proposed because there is no consolidated basis to provide a strong theoretical foundation for the AIS, or most researchers do not want to build their solutions on top of such basis.

In regards to **RQ1**, 40% of the solutions identified in this research do not take teamwork factors in consideration. We consider this percentage very high given that the focus of this research is AIS. It is contradictory the fact that some solutions state they are focused on ASD, but do not approach important and valuable principles stated in the Agile Manifesto [20]. The lack of AIS that take teamwork factors in consideration is often identified in solutions that focuses on agile practices and objectives.

**RQ2** is answered according to validation of the AIS on industry environments. The rate of solutions that were not validated in real projects is higher than 50%. This percentage shows that there is a big gap between what the AIS identified promise and their application in real environments. Moreover, as discussed before, it is possible to argue that the lack of validation of these solutions may be a crucial factor contributing to the absence of a consolidated basis. The reason to believe in such a statement is very clear: if there is no data regarding the approval of these solutions in the industry, how would these

TABLE I  
EXTRACTED DATA RELATED TO RQ FROM 18 PAPERS.

Title	Teamwork Factors	Validated in Industry	Evidence of Benefits	Tool Support
Progressive outcomes: A framework for maturing in agile software	X	X	X	
Scrum Maturity Model				
Agile maturity model (AMM): A software process improvement framework for agile software development practices	X			X
Agile Compass: A Tool for Identifying Maturity in Agile Software Development Teams	X			X
Light maturity models (LMM): an Agile application				
An evaluation of the degree of agility in six agile methods and its applicability for method engineering	X			X
Rebalancing Your Organization's Agility and Discipline	X			X
A framework for adapting agile development methodologies	X			
Adopting agile in distributed development		X	X	X
Project agility assessment: An integrated decision analysis approach		X	X	
An Approach for Assessing Suitability of Agile Solutions: A Case Study		X		
A disciplined approach to adopting agile practices: The agile adoption framework				X
An iterative improvement process for agile software development			X	
Adapting the Lean Enterprise Self-Assessment Tool for the Software Development Domain	X	X		X
Tailoring for agile methodologies: a framework for sustaining quality and productivity	X	X		
Agile Transition Model Based on Human Factors	X			
A Mapping Model for Transforming Traditional Software Development Methods to Agile Methodology	X			
AM-QuICk: A Measurement-Based Framework for Agile Methods Customisation	X	X		

TABLE II  
RQ RELATED RESULTS.

Research Question	Percentage	Quantity
Teamwork Factors	60%	27
Validated in Industry	46.7%	21
Evidence of Benefits	45.5%	10
Tool Support	28.9%	13

solutions evolve and serve as basis for future researches that pursue to solve similar problems?

48.9% of the solutions that were validated, but 2.2%, which correspond to 1 within the 45 identified AIS, were validated in Academic context. The percentage of validated AIS correspond to 21 within the 45 identified AIS. From these 21 AIS, 45.5% percentage of them do not present traces of increased efficiency/performance/productivity or cost reduction. In other words, almost half of them do not prove that there is a gain when applying such a solution in an industry environment. This percentage is an important indicator to explicit the lack of contributions these solutions provide to the industry environment. Some researchers may argue that they received positive feedback from the subjects involved on the validation of their AIS. However, these feedback could not provide substantial conclusion as a numeric evidence (e.g., based on the comparison between the effort to implement and use an AIS in a specific context, and the gains it provides). These discussion

regarding **RQ3** explicit the need for a better collaboration and proximity between the academic and industrial environments. This proximity could be crucial to obtain a more consolidated concept of agile maturity.

The percentage of AIS related to **RQ4** indicates that only 23.8% of the included AIS provide tool support. This small percentage is another indicator of the existing gap between the industry and the academic community. Nowadays, there is a problem when trying to implement solutions proposed by the academic community into the industry. We believe it occurs because, usually, researches do not try to address their solutions to a more practical utility instead of solving a small problem with an approach that requires too much effort. Industry practitioners want tools, solutions, and instruments, that can make their work easier without requiring them to put more effort than they actually do.

## VI. THREATS TO VALIDITY

As well as in all SLR studies, a common threat to validity regards to the covering of all relevant studies. Therefore, to mitigate this problem, we executed the snowballing technique, as described in [15], until no more relevant papers were found.

Another threat is related to the researchers' opinions and judgments in regards to the extraction of the data, which can influence the results of the study. To mitigate this problem, each paper was evaluated by two reviewer. Also, depending on the disagreement between two reviewers, a third random

reviewer was invited to the process, as explained in Section III-B.

## VII. CONCLUSION

In this paper, we conducted a SLR to investigate if AIS take in consideration important agile teamwork key factors; if they were applied in real projects; if, after applying them, it was possible to identify increases on efficiency/performance/productivity, or even cost reduction (i.e., something that proves that applying a given AIS brings real earnings besides positive feedback of subjects); and whether they provide tool support for applying them in real projects or not.

To reach this goal, we executed the snowballing procedure [15] on a start set containing 22 papers. From these 22 papers, we were able to identify other 33 relevant papers, resulting in a total of 55 included papers. From these 55 relevant papers, we identified 45 different AIS.

The results of this SLR indicate some of the possible reasons behind the lack of usage of academic proposed solutions in the industry environment. Most of the AIS investigated in this research were not validated in real projects, and do not provide tool support to facilitate their adoption. Moreover, almost half of the investigated that were validated do not present any trace of increased efficiency/performance/productivity, or even cost reduction.

Another issue identified in regards to the investigated AIS is that a high percentage of them do not present fidelity to the Agile Manifesto [20] because they do not take in consideration agile teamwork factors.

For future works, we intend to perform a more detailed analysis of the papers to understand the structure (e.g., levels, steps, stages, components, etc.) of the AIS, and perform a detailed comparison between them by category (i.e., maturity model, assessment, adoption). We also intend to complement the database by executing the forward snowballing on the relevant papers identified so far.

## ACKNOWLEDGMENT

The authors would like to thank CAPES for supporting this research.

## REFERENCES

- [1] W. Alsaqaf, M. Daneva, and R. Wieringa, "Quality requirements in large-scale distributed agile projects – a systematic literature review," in *Requirements Engineering: Foundation for Software Quality*, P. Grünbacher and A. Perini, Eds. Cham: Springer International Publishing, 2017, pp. 219–234.
- [2] O. Salo and P. Abrahamsson, "An iterative improvement process for agile software development," *Software Process: Improvement and Practice*, vol. 12, no. 1, pp. 81–100, 2007. [Online]. Available: <http://dx.doi.org/10.1002/spip.305>
- [3] I. Aaen, J. Arent, L. Mathiassen, and O. Ngwenyama, "A conceptual map of agile software development," *Scandinavian Journal of Information Systems*, vol. 13, pp. 123–146, Jun. 2001. [Online]. Available: <http://dl.acm.org/citation.cfm?id=565431.565437>
- [4] K. Lukasiewicz and J. Miler, "Improving agility and discipline of software development with the scrum and cmmi," *IET Software*, vol. 6, no. 5, pp. 416–422, October 2012.
- [5] S. W. Baker, "Formalizing agility, part 2: how an agile organization embraced the cmmi," in *AGILE 2006 (AGILE'06)*, July 2006, pp. 8 pp.–154.
- [6] C. R. Jakobsen and K. A. Johnson, "Mature agile with a twist of cmmi," in *Proceedings of the Agile 2008*, ser. AGILE '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 212–217. [Online]. Available: <https://doi.org/10.1109/Agile.2008.10>
- [7] J. Sutherland, C. R. Jakobsen, and K. Johnson, "Scrum and cmmi level 5: The magic potion for code warriors," in *Agile 2007 (AGILE 2007)*, Aug 2007, pp. 272–278.
- [8] S. Cohan and H. Glazer, "An agile development team's quest for cmmi maturity level 5," in *2009 Agile Conference*, Aug 2009, pp. 201–206.
- [9] D. J. Anderson, "Stretching agile to fit cmmi level 3 - the story of creating msf for cmmi reg; process improvement at microsoft corporation," in *Agile Development Conference (ADC'05)*, July 2005, pp. 193–201.
- [10] M. C. Paulk, "Extreme programming from a cmm perspective," *IEEE Software*, vol. 18, no. 6, pp. 19–26, Nov 2001.
- [11] R. M. Fontana, I. M. Fontana, P. A. da Rosa Garbuio, S. Reinehr, and A. Malucelli, "Processes versus people: How should agile software development maturity be defined?" *Journal of Systems and Software*, vol. 97, pp. 140 – 155, 2014.
- [12] M. Kohlegger, R. Maier, and S. Thalmann, "Understanding maturity models results of a structured content analysis," in *Proceedings of I-KNOW '09 and I-SEMANTICS '09*, September 2009, pp. 193–201.
- [13] L. Buglione, "Light maturity models (lmm): An agile application," in *Proceedings of the 12th International Conference on Product Focused Software Development and Process Improvement*, ser. Profes '11. New York, NY, USA: ACM, 2011, pp. 57–61. [Online]. Available: <http://doi.acm.org/10.1145/2181101.2181115>
- [14] B. Kitchenham and S. Charters, "Guidelines for performing Systematic Literature Reviews in Software Engineering," Keele University and Durham University Joint Report, Tech. Rep. EBSE 2007-001, 2007.
- [15] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, ser. EASE '14. New York, NY, USA: ACM, 2014, pp. 38:1–38:10. [Online]. Available: <http://doi.acm.org/10.1145/2601248.2601268>
- [16] M. T. Vaughan Henriques, "A systematic literature review of agile and maturity model research," *Interdisciplinary Journal of Information, Knowledge, and Management*, vol. 12, pp. 53–73, 2017. [Online]. Available: <http://www.ijikm.org/Volume12/IJIKMv12p053-073Henriques3025.pdf>
- [17] F. S. Silva, F. S. F. Soares, A. L. Peres, I. M. de Azevedo, A. P. L. Vasconcelos, F. K. Kamei, and S. R. de Lemos Meira, "Using cmmi together with agile software development: A systematic review," *Information and Software Technology*, vol. 58, pp. 20 – 43, 2015. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0950584914002110>
- [18] J. López-Martínez, R. Juárez-Ramírez, C. Huertas, S. Jiménez, and C. Guerra-García, "Problems in the adoption of agile-scrum methodologies: A systematic literature review," in *2016 4th International Conference in Software Engineering Research and Innovation (CONISOFT)*, April 2016, pp. 141–148.
- [19] K. Dikert, M. Paasivaara, and C. Lassenius, "Challenges and success factors for large-scale agile transformations: A systematic literature review," *Journal of Systems and Software*, vol. 119, pp. 87 – 108, 2016. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0164121216300826>
- [20] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas, "Manifesto for agile software development," <http://www.agilemanifesto.org/>, 2001.
- [21] N. B. Ali, K. Petersen, and C. Wohlin, "A systematic literature review on the industrial use of software process simulation," *Journal of Systems and Software*, vol. 97, pp. 65 – 85, 2014. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0164121214001502>

# Towards Cost Effective Privacy Provision for Typed Resources in IoT Environment

Yucong Duan<sup>\*1,2</sup>, Zhengyang Song<sup>1,2</sup>, Xiaoxian Yang<sup>3</sup>, Quan Zou<sup>4</sup>, Xiaobing Sun<sup>5</sup>, Xinyue Zhang<sup>1,2</sup>

<sup>1,2</sup>State Key Laboratory of Marine Resource Utilization in the South China Sea, College of Information Science and Technology, Hainan University, 570228 Haikou, China

<sup>3</sup>School of Computer and Information Engineering, Shanghai Polytechnic University, 201209 Shanghai, P.R. China

<sup>4</sup>College of Computer Science, Tianjin University, China

<sup>5</sup>School of Information Engineering, Yangzhou University, China

Email: duanyucong@hotmail.com, 1464626602@qq.com, xxyang@sspu.edu.cn, zouquan@tju.edu.com, sundomore@163.com, yuexinaai@163.com

**Abstract**—We present privacy resources in IoT as data, information, and knowledge. We construct a privacy protection architecture on our previously proposed DIKW graphs: Data Graph, Information Graph, and Knowledge Graph. On this architecture, we search privacy protection target resources both as they appear explicitly in their original types and as they appear implicitly which means that they are expressed not in their original types. For a single privacy protection target, it may have various concrete compositions in various layers of DIKW Graph. It becomes more complex since the implementation of a privacy target might also be intertwined with the implementation of other privacy targets. We propose to protect target resources according to their types by either isolating the elements comprising an implementation, or weakening relationships among elements comprising an implement. To optimize among several choices of implementing a protection in a business environment, we introduced the tradeoff between customers' expectations/investment and privacy providers' expectation. Thereafter we proposed to prioritize implementation according to their ratio of cost/benefit.

**Keywords**—Internet of Things; typed resources; privacy provision; Knowledge Graph

## I. INTRODUCTION

We classify privacy resources into data privacy, information privacy, and knowledge privacy. Chaim [1] illustrated the concepts of defining data, information and knowledge. Duan et al. [3] clarified the architecture of Knowledge Graph in terms of data, information, knowledge and wisdom. In [4], the authors proposed to designate the form of Knowledge Graph as four basic forms including Data Graph, Information Graph, Knowledge Graph and Wisdom Graph. We have identified there are enormous potentials of security protection according to the difference of resource types in an investment driven manner [2]. We propose to protect privacy resources in a three-tier architecture consisting of Data Graph, Information Graph and Knowledge Graph. For a single privacy protection target, it may have various concrete compositions in various layers of DIKW Graph. For example, a piece of data might exist as a piece of data or a set of data in Data Graph explicitly, or it might take the implicit form of being expressed as a series of relationships in Information Graph. It becomes more complex since the implementation of a privacy target might also be intertwined with the implementation of other privacy targets.

In [5], the authors proposed a cost effective approach to satisfy performance requirements while minimizing dynamic power consumption. In [6], cost-effective data sharing with

forward security improved efficiency is provided through reducing the computation and communication cost. In [7], the authors proposed to provide the I/O resources for specific workloads, while minimizing the total operating cost. We elaborate towards cost effective [8] information privacy provision approach on the basis of Data Graph, Information Graph, and Knowledge Graph.

The rest of this paper is organized as follows. Section II defines typed resources and presents a privacy protection architecture. Section III shows a running example. Section IV presents implementation procedures. Section V shows a simulation. We conclude in Section VI.

## II. PRIVACY PROTECTION ARCHITECTURE

### A. Definitions of Typed Resources

We have reconstructed a DIKW system for modeling and implementation of resource identification and management[2], [3]. Data is not specified for a specific stakeholder or a machine. Data represents directly observed objects as isolation which only contains the shared common meaning of their necessary identifications. Information represents data or information which are observed or interacted directly or indirectly by human. Knowledge represents the abstracted data, information and knowledge which are taken in a limited or unlimited complete manner as a whole. Knowledge here can be roughly mapped to cover what Kant called Categories[9]. Knowledge is exploited to reason and predict unknown resources or not observed but happened relationships in terms of Data, Information and Knowledge. We are actually building "schemas"[9] for DIKW resources for privacy modeling and provision subsequently.

**Definition 1. Typed resources.** We define typed resources as a triad:

$$TR_{DIK} = \langle D_{DIK}, I_{DIK}, K_{DIK} \rangle$$

D represents Data, I represents Information and K represents Knowledge for convenient description.  $D_{DIK}$  is transformed to  $I_{DIK}$  through taking roles in real or imaginary scenarios by connecting to other  $D_{DIK}$  or  $I_{DIK}$  in term of time [9] or order. The associated  $D_{DIK}$  corresponds to  $I_{DIK}$ .

**Definition 2. DIKWGraph.** We specify the usually used concept of Knowledge Graph in three layers of Data Graph ( $DG_{DIK}$ ), Information Graph( $IG_{DIK}$ ), and Knowledge Graph( $KG_{DIK}$ ) [3].

DIKWGraph = < (DG<sub>DIK</sub>), (IG<sub>DIK</sub>), (KG<sub>DIK</sub>)>.

**Definition 3. DG<sub>DIK</sub>.**

DG<sub>DIK</sub>: = collection {array, list, stack, queue, tree, graph}.

DG<sub>DIK</sub> is a collection of discrete elements expressed in the form of various data structures including arrays, lists, stacks, trees, graphs, and so on. DG<sub>DIK</sub> can record basic structures of entities. Also, DG<sub>DIK</sub> can record spacial and topological relationships with frequencies.

**Definition 4. IG<sub>DIK</sub>.**

IG<sub>DIK</sub>: = composition<sub>time</sub> { D<sub>DIK</sub> }.

IG<sub>DIK</sub> comprises of temporal relationships based on D<sub>DIK</sub> with specific scenarios. IG<sub>DIK</sub> expresses the interaction and transformation of I<sub>DIK</sub> between entities in the form of a directed graph. IG<sub>DIK</sub> can record the interactions between entities including direct interaction and indirect interaction.

**Definition 5. KG<sub>DIK</sub>.**

KG<sub>DIK</sub>: = collection<sub>consistent</sub> {Rules<sub>Statistic OR Logical</sub> } category.

KG<sub>DIK</sub> consistently accommodates either empirical statistical experiences expressed in terms of categories which represent the underlying elements as a whole or completely.

*B. Schemas for using DIKW Graphs*

To utilize the graphs in DIKW Graphs, we need to mediate the bidirectional feasible transformations of resources among different types of Data, Information and Knowledge. By restricting the transformation with feasible, we mean that not all bidirectional transformations are deemed to be meaningful and practical.

Schemas[9] are proposed by Kant to cognitively mediate the cognitive objects/experiences mostly through logical reason and concretization in time dimension. We borrowed this term here for specifying the transformation among resources with a focus on the type level implementation.

**Schema “Data-Resource(Data, Information)”**: Data are observed by observers from outside world or from inside categorization on a set of resources, structured or not, which are given the conceptual unity as an entity, or on abstraction of information expressions which are exposed as temporal association among elements. Since resource elements can be abstracted upward or decomposed downward, the expressions of specific DG<sub>DIK</sub> and IG<sub>DIK</sub> are therefore intertwined based on the overlapping of the elements and their relationships. We propose to justify and predict the semantic meaning and semantic associations corresponding to resource element expressions based on the reasoning and calculation in a bottom up manner from composing elements of DG<sub>DIK</sub> and IG<sub>DIK</sub>.

**Schema “Knowledge-Resource(Data, Information, knowledge)”**: Knowledge here is either based on the probabilistic experience or based on reason on categories abstracted from directly observed resources or indirectly reasoned resources. A shared characteristic of both forms of knowledge is that they both demand a semantic identification of completeness regardless of whether the actual target resources which are the basis of

conceptualization of related categories are limited or unlimited. The schema to enact knowledge on resources is either through temporally decomposing the content of the comprising categories in the knowledge expression as elements shared or can be related to elements in target resources, or through logical or probability reasoning first and decomposing and relating subsequently.

For construction of “Wisdom” related schemas, we adopt the intuition from Schopenhauer[10] to take wisdom as the balancing between reasoning and will for optimizing human long run goals. We omit the discussion on the schema of wisdom here.

*C. Privacy Protection Architecture*

I<sub>DIK</sub> expresses interaction and collaboration between entities. Through classifying and abstracting interactive records or behavior records related to the dynamic behavior of entities, we obtain K<sub>DIK</sub> in the form of statistical rules. We infer K<sub>DIK</sub> from known resources and collect necessary I<sub>DIK</sub> in the process of inference through appropriate techniques such as experiments, surveys, and so on. Transformation from I<sub>DIK</sub> to D<sub>DIK</sub> takes place either as a conceptualization process from relationships to an entity or as an abstraction which selectively maps the involved elements comprising the I<sub>DIK</sub> to elements in a structure of a target elements of D<sub>DIK</sub>. We obtain D<sub>DIK</sub> through observing an object at a certain time in a static state. K<sub>DIK</sub> elements are either associated with underlying fine grained instances within their categories such as sub-attributes or sub-operations, or connected through pure logical reason or mathematical computation. The top-down influence from K<sub>DIK</sub> to I<sub>DIK</sub> and D<sub>DIK</sub> is realized through creatively decomposing of the content of K<sub>DIK</sub> to I<sub>DIK</sub> and D<sub>DIK</sub> temporally.

<p><b>Algorithm 1.</b> Process of Privacy Protection Architecture</p> <p><b>Input:</b> Target Privacy Resources  <b>Output:</b> Final Elements  <b>SWITCH</b> ( all target privacy resources)  <b>CASE 1:</b> Data resources  <b>IF</b> (they are explicit)              Isolate or transform them;  <b>ELSE IF</b> (they are implicit)              Find out them on DIKW Graphs;              Isolate or transform them;  <b>CASE2:</b> Information resources  <b>IF</b> (they are explicit)              Isolate or transform them;  <b>ELSE IF</b> (they are implicit)              Find out them on DIKW Graphs;              Isolate or transform them;  <b>Return</b> final elements;</p>
--

Our process of privacy protection architecture is shown as Algorithm 1. When we protect target Data privacy resources, we search for them on the three-layer graphs firstly. There are two scenarios. One is that these resources are explicit, we can succeed in searching them directly. The other is that we fail to search, then we analyze associated relationships between target Data privacy resources and other relative typed resources on DG<sub>DIK</sub>, IG<sub>DIK</sub>, and KG<sub>DIK</sub>. We infer them through three associated relationships as associated Data resources infer target Data resources, associated Information resources infer target Data resources and associated Data and Information resources infer target Data resources.

When we protect target Information privacy resources,

we find them on the three-layer graphs firstly. There are also two scenarios the same as finding Data privacy resources. If these Information resources are explicit, we isolate them or transform them directly. If they are not, we find out them on DIKW Graphs. At last, we isolate these resources, or transform them into other typed resources and store these final elements into a security space, in which the elements will not be used, tampered with, lost and destroyed in unauthorized situations.

### III. RUNNING EXAMPLE

We design a campus monitoring system shown in Figure 1, which consists of geographic location acquisition module, credit card consumption tracking module, video acquisition module, and resource analysis and processing module.

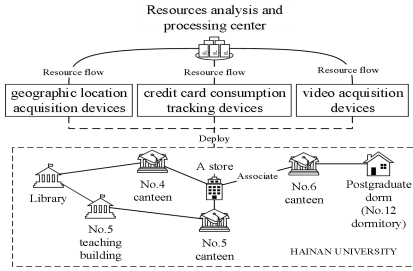


Figure 1. Campus monitoring system

We collected resources and construct  $DG_{DIK}$ ,  $IG_{DIK}$ , and  $KG_{DIK}$ . When target privacy resources are implicit, there is a way that we find out these resources by analyzing associated resources in forms of specific  $ID_{DIK}$ , such as linked and aggregative structure  $ID_{DIK}$ . LSI denotes linked structure  $ID_{DIK}$ . ASI denotes aggregative structure  $ID_{DIK}$ .

Given an  $IG_{DIK}$  denoted as a directed graph  $G=(V, E)$ , where  $V$  is a set of nodes,  $E$  is a set of edges connecting those nodes.  $VoN$  denotes the value of a node which has two attributes of the number and direction of edge connected to this node. We use  $t(f_1(\text{direction}(N)) * f_2(\text{number}(N)))$  to evaluate  $VoN$ .

#### A. Protection strategies of LSI

1) Protection of linked  $ID_{DIK}$  without branched structure in LSI (IUP). We present two strategies to protect IUP privacy which are illustrated as follows.

**Binary Damage Method:** We search the middle node of each IUP. Second, we continue to search two middle nodes of two IUP parts divided by previous middle node, respectively. And so on, we find out  $m$  nodes need protection.

**Middle Centralized Damage Method:** We search nodes of the middle location in IUP. In sequence, we find out  $m$  nodes need protection.

After searching, we isolate or transform these  $m$  nodes and store them into security space.

2) Protection of linked  $ID_{DIK}$  with branched structure in LSI (IBP). We rank nodes of this link according to  $VoN$ . We find out  $n$  nodes in sequence. Then we isolate or transform those  $n$  nodes and store them into a security space.

#### B. Protection strategies of ASI

1) Protection of aggregative nodes with equal value (VEG). We find out  $h$  nodes according to depth-first algorithm or breadth-first algorithm, after isolating or transforming these  $ID_{DIK}$  nodes, we store them in a security space.

2) Protection of aggregation nodes with unequal node value (VUG). Similar to protection of IBP, we determine the order of target privacy resources according to the rank of  $VoN$ . For example, in Figure 1, after analyzing, we know that a student uses his credit card in No.6 canteen frequently. But he rarely uses his credit card in No.4 canteen which is popular to students. Therefore we infer this student lives near No.6 canteen. Since No.6 canteen is near postgraduate dorms, we can infer this student is a postgraduate. Our proposed mechanism isolates or transforms these collected resources of No.6 dormitory according to the rank of  $VoN$  and store them into a given secure space.

### IV. FRAMEWORK OF VALUE DRIVEN PRIVACY PROVISION

To optimize among several choices of implementing a protection in a business environment, we introduced the tradeoff between customers' expectations/investment and privacy providers' expectation.

**Definition 9.** We denote target privacy resource collection as  $TPR$ .  $TRPC$  represents associated privacy resources which is the target resource processing collection. We define  $TRPC$  as a tuple:

$$TRPC: = \langle LSIC, ASIC \rangle$$

$LSIC$  is the set of LSI privacy resources and  $ASIC$  is the set of ASI privacy resources.  $LSIC = \{IUPC, IBPC\}$ , where  $IUPC$  is the set of IUP privacy resources, and  $IBPC$  is the set of IBP privacy resources.  $ASIC = \{VEGC, VUGC\}$ , where  $VEGC$  is the set of VEG privacy resources, and  $VUGC$  is the set of VUG privacy resources.

**Definition 10. Security Space.** We define security space denoted with  $SS$  as a tuple:

$$SS: = \langle SST, SSS \rangle$$

$SST$  is the type set of graph resources denoted with a triad  $\langle sst_d, sst_i, sst_k \rangle$ .  $SSS$  is the scale of different kinds of graph resources represented by a triad  $\langle sss_d, sss_i, sss_k \rangle$ . Each  $sss$  denotes the scale of resource in the form of  $sst$ .

#### A. Calculation of User Investment

1) **Cost of damaging nodes:** We assign that  $1C$  is atomic cost of damaging each node in  $TRPC$ . In fact, it is necessary to analyze importance of relationship between nodes. But at present we only consider the number of nodes when computing cost. Cost of damaging nodes can be illustrated as

$$DeCost = (m + n + h + k) * 1C \quad (1)$$

2) **Cost of transforming  $TRPC$  into  $SS$ :**  $SS$  is a security space. We convert resource types to optimize storage and computation, which makes it hard for unauthorized users to access protected resources. As shown in Table I,  $PUnitCost$  represents atomic cost of transforming unit resource into  $SS$ .

Cost of transforming resources is illustrated as

$$\text{TrCost} = \sum_{i \in \{D, I, K\}} \text{SUnitCost}_i * \text{ss}_i + \text{ss}_i' \quad (2)$$

TABLE I. ATOMIC COST OF CONVERTING UNIT RESOURCE IN SS

	$D_{DIK}$	$I_{DIK}$	$K_{DIK}$
$D_{DIK}$	SUnitCost <sub>D-D</sub>	SUnitCost <sub>D-I</sub>	SUnitCost <sub>D-K</sub>
$I_{DIK}$	SUnitCost <sub>I-D</sub>	SUnitCost <sub>I-I</sub>	SUnitCost <sub>I-K</sub>
$K_{DIK}$	SUnitCost <sub>K-D</sub>	SUnitCost <sub>K-I</sub>	SUnitCost <sub>K-K</sub>

### 3) User investment computaion

Costs of providing protection services for private resources consist of two parts as damaging nodes and transforming nodes and store them into SS. We calculate total cost of protecting target privacy, which is illustrated as

$$\text{TotalCost} = \text{DeCost} + \text{TrCost} \quad (3)$$

$\mu$  denotes the unit investment of TotalCost that obtained through data training. Corresponding user investment can be illustrated as

$$\text{UserCost} = \mu * \text{TotalCost} \quad (4)$$

### B. Privacy Level Computation

Privacy level reflects the ability of protection service. The smaller privacy level is, the better protection ability the service has. We denote the privacy level as PL.  $M_m$  represents the total number of nodes in  $m$ th link. Privacy level of IUP  $I_{DIK}$  denoted with  $LPL_{UP}$  is illustrated as:

$$LPL_{UP} = m / M_m \quad (5)$$

$N_n$  represents the total number of nodes in  $n$ th link.  $\alpha$  represents an adjusted parameter that is obtained through data mining.  $V_i$  represents node  $V_i$  that belongs to  $n$ th link. Privacy level of IBP  $I_{DIK}$  denoted with  $LPL_{BP}$  is illustrated as:

$$LPL_{BP} = (\sum_{i \leq n} \alpha * \text{VoN}(V_i)) / N_n \quad (6)$$

$H_h$  denotes the total number of nodes in  $h$ th aggregation. Privacy level of VEG  $I_{DIK}$  denoted with  $APL_{EG}$  is illustrated as:

$$APL_{EG} = h / H_h \quad (7)$$

$K_k$  denotes the total number of nodes in  $k$ th aggregation.  $\beta$  represents an adjusted parameter that is obtained through data mining.  $V_i$  represents node  $V_i$  that belongs to  $k$ th link. Privacy level of VUG  $I_{DIK}$  denoted with  $APL_{UG}$  is illustrated as

$$APL_{UG} = (\sum_{i \leq k} \beta * \text{VoN}(V_i)) / K_k \quad (8)$$

Algorithm 2 describes a procedure of calculating PL according to user investments.

Algorithm 2. Calculating PL
<b>Input:</b> TRPC, SS, UserCost <sub>0</sub> , PL <sub>0</sub>
<b>Output:</b> The maximum PL <sub>0</sub>
<b>FOR</b> all TRPC <b>DO</b>
Calculate DeCost; Calculate TrCost;
Calculate PL; Calculate UserCost;
<b>IF</b> (UserCost < UserCost <sub>0</sub> & PL > PL <sub>0</sub> )
PL <sub>0</sub> = PL;
<b>ELSE IF</b> (m ≤ M <sub>m</sub> OR n ≤ N <sub>n</sub> OR k ≤ K <sub>k</sub> )
Next step;
<b>Return</b> PL <sub>0</sub> ;

## V. SIMULATION

Following the scenario in section III, we set target privacy collection (TPR) and constructed  $DG_{DIK}$ ,  $IG_{DIK}$ , and  $KG_{DIK}$  according to collected resources. We considered a constructed  $IG_{DIK}$  denoted with a directed graph  $G = (V, E)$ . We assigned that expectant investment is 50 units and expectant privacy level is 0.6. For convenience, we assumed that there are same resource collections in all considered 18 nodes, which is  $\{I, D, I\}$ . Meanwhile, we assigned that  $\text{ss}_i = \{D, I, K\}$ ,  $\text{ss}_i = \{4, 2, 1\}$ ,  $\mu = 0.4$ ,  $1C = 2$  units. Figure 2 shows that the greater the number of nodes which need protection is, the better PL of each  $I_{DIK}$  structure performs.

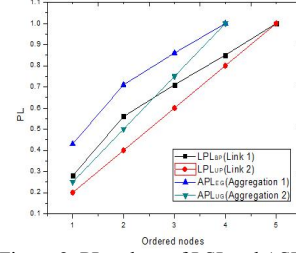


Figure 2. PL values of LSI and ASI.

## VI. CONCLUSION

According to the types of various resources, we provide protection solutions based their state of being explicitly or implicitly expressed. During the implementation, we take into consideration of the tradeoff between the value expectation of customers and the value expectation of privacy providers to prioritize privacy targets and corresponding levels of protection.

## ACKNOWLEDGMENT

We acknowledge Hainan Project No.ZDYF2017128, NSFC under Grant (No.61662021 and No. 61502294). \*refers to corresponding author.

## REFERENCES

- [1] C. Zins, "Conceptual approaches for defining data, information, and knowledge," JASIST, vol. 58, no. 4, pp. 479–493, 2007.
- [2] L. Shao, Y. Duan, L. Cui, Q. Zou, and X. Sun, "A pay as you use resource security provision approach based on data graph, information graph and knowledge graph," IDEAL 2017, pp. 444–451.
- [3] Y. Duan, L. Shao, G. Hu, Z. Zhou, Q. Zou, and Z. Lin, "Specifying architecture of knowledge graph with data graph, information graph, knowledge graph and wisdom graph," SERA 2017, pp. 327–332.
- [4] L. Shao, Y. Duan, X. Sun, Q. Zou, R. Jing, and J. Lin, "Bidirectional value driven design between economical planning and technical implementation based on data graph, information graph and knowledge graph", SERA 2017, pp. 339–344.
- [5] J. Guerra, H. Pucha, J. S. Glider, W. Belluomini, and R. Rangaswami, "Cost effective storage using extent based dynamic tiering," 9th USENIX Conference on File and Storage Technologies, 2011, pp. 273–286.
- [6] X. Huang, J. K. Liu, S. Tang, Y. Xiang, K. Liang, L. Xu, and J. Zhou, "Cost-effective authentic and anonymous data sharing with forward security," IEEE Trans. Computers, vol. 64, no. 4, pp. 971–983, 2015.
- [7] N. Zhang, J. Tatemura, J. M. Patel, and H. Hacig'um'us, "Towards cost-effective storage provisioning for dbmss," CoRR, vol. abs/1201.0226, 2012.
- [8] M. A. Jabbar, G. S. Bopche, B. L. Deekshatulu, and B. M. Mehtre, "Diversity-aware, cost-effective network security hardening using attack graph," SSCC 2017, pp. 1–15.
- [9] Kant I. Critique of pure reason[M]. Cambridge University Press, 1998.
- [10] Schopenhauer, The World as Will and Representation, Vol. I, Appendix, "Criticism of the Kantian Philosophy," p. 449 f.



# Finding Shilling Attack in Recommender System based on Dynamic Feature Selection

Gaofeng Cao, Huan Zhang, Yuyou Fan, Li Kuang\*

School of software  
Central South University  
Changsha, China

{caogaofeng, 3901140112, hatsune, kuangli} @csu.edu.cn

**Abstract**—Recommender system is widely used as an important tool in various fields for effectively dealing with information overload, and collaborative filtering algorithm plays a vital role in the system. However, such system is highly vulnerable to malicious attacks, especially shilling attack because of data openness and independence. Therefore, detecting shilling attack has become an important issue to ensure the security of recommender system. Most of existing methods for detecting shilling attack are based on rating classification features and their limitation is that they are easily to be interfered by obfuscation techniques. Moreover, traditional detection algorithms can not handle multiple types of shilling attack flexibly. In order to solve these problems, in this paper, we propose an outlier degree shilling attack detection algorithm based on dynamic feature selection. By considering the differences of user choosing items and taking user popularity as a detection metric, as well as using information entropy to select detection metrics dynamically, a variety of shilling attack models can be dealt with flexibly. Experiments show that the algorithm has stronger detection performance and interference immunity in shilling attack detection.

**Keyword**—Recommender System; Malicious Attacks; Detection Algorithm; User Selection; Detection Metrics

## I. INTRODUCTION

As an information filtering technology, recommender system plays a role with importance increasing and has become an effective way to deal with information overload. Typical recommendation approaches, including content-based recommendation [1], collaborative filtering recommendation [2], knowledge-based recommendation [3], hybrid recommendation [4], have been widely used in large e-commerce websites such as Taobao, Amazon, Google News, etc. A good recommender system can provide users with relevant interesting items and thus bring more economic benefits to merchants. Not only has now research on recommender system become a popular research field in academia, but also many companies, such as Netflix and Alibaba, have set up their own research teams in order to improve the accuracy of their own recommender system.

At present, recommender system is faced with many problems such as data sparse [7-8], poor scalability, cold-start [9], security [10], etc., and the security will be the focus of this paper. The openness can reflect user's preference through

rating, which provides data foundation for recommendation. However, because of the openness some junk information could be inserted into the system by malicious users and thus influence system's behaviors, like, in recent years, popular network part-time jobs "brush credit" and "brush praise". This phenomenon is called aggression behavior of malicious user [11], profile injection attack [12] or shilling attack [13]. Facing with shilling attack, traditional collaborative filtering recommender system shows their vulnerability that is attackers can change the predictions of some target items when the system has no protection. The inserted junk information causes a decline in accuracy and reliability of the system.

Detecting shilling attack can be regarded as a binary classification problem between normal users and attackers. When it comes to classifying attackers, most current classification features are relative to user ratings, and the corresponding classification features, which can differentiate normal users and fake users, could be found by detecting how they rate certain items. However, there are some problems in classification features based on ratings: (1) Misjudging a user as an attacker easily; (2) When attacker's ratings are camouflaged and not the same as the normal shilling attack models, it will result in low detection accuracy, and the current detection metrics are useless for various changes of shilling attack models.

In order to solve above problem, this paper starts from dealing with the user's selection of rating items. Since normal user has certain needs to choose items -- item popularity is generally follow long tail effects, thus we can use the user's popularity [21] as a metric to detect the shilling attack. And by using information entropy effective detection metrics can be selected dynamically; the most effective metric helps to calculate user's outlier degree [22] and then we can detect attackers. Taking detection mistakes into account, we propose a new method which can get the intent and target items of shilling attack through analyzing suspected users, and remove abnormal users, users always give good reviews or bad reviews, based on the information we get.

The main contributions of this paper are as follows: (1) Combining user's popularity with conventional classification features based on ratings as detection metrics to improve the accuracy of shilling attack detection. (2) Using information entropy to dynamically select metrics to adapt a flexibility in

coping with various attack models. (3) Using metrics selected dynamically to calculate user's outlier degree and detect attacker.

The rest of this paper is structured as follows: In Section 2 we introduce the research background. In Section 3 we propose an outlier degree shilling attack algorithm based on dynamic feature selection, and then we introduce the experiment and analyze experimental results in Section 4. Finally we conclude with a summary and future work in Section 5.

## II. BACKGROUND

Due to that the accuracy of collaborative filtering recommendation depend on a large amount of user data and the open nature of recommender system, so that attackers can inject fake profiles into the system with a little cost and maximize their interests by affecting the prediction results with the attack profiles. Shilling attack contains two intents: (1) increase the recommendation frequency of target items, namely push attack; (2) reduce the recommendation frequency of target items, namely nuke attack.

The research on shilling attack mainly includes attack detection and robust recommendation algorithm of defense. This paper is to analyze algorithms of shilling attack detection, and there are two main categories: based on supervised learning and unsupervised learning.

Research on shilling attack detection has been fully developed. Chirita et al. [14] proposed using statistical metrics, such as the degree of similarity with top neighbors, rating deviation from mean agreement (RDMA), to distinguish genuine profiles and attack profiles. This method performs very well in the detection on the attack profiles of high density filling but not great in low density filling. Mehta et al. [15-16] believe that the information in recommender system mainly depends on genuine profiles and they used principal component analysis technology to filter attack profiles. Then, they proposed a PCA-Var Select detection that can effectively detect multiple attack types. Li Cong et al. [17] constructed a corresponding object function for genetic optimization through qualifying the group effect of attack profiles and combined it with Bayesian inference in the process of genetic optimization, which is an new unsupervised algorithm for detecting shilling attack — IBIGDA. To some extent, IBIGDA reduces the dependence on prior knowledge, but it still assumes that the number of attack profiles is less than genuine profiles and obtain higher precision with sacrificing recall. Chung et al. [18] proposed a detection algorithm based on Beta distribution, namely Beta-Protection, to detect attack profiles. Beta-Protection has better detection performance when it meets certain conditions: the number of ratings is extremely small, the rating value is extremely small or extremely large.

According to the existing researches, the existing unsupervised algorithms for detecting shilling attack only rely on one solid feature and take it as a detection metric of attack profiles. This kind of single detection metric is difficult to ensure the accuracy under different attack scenarios and its inflexibility causes problems when nre attack strategies appear.

In order to improve accuracy and interference immunity of the detection algorithm, first we use rating metrics and popularity-based metrics in the literature [19,23] as a feature candidate set of detecting shilling attack; the second step uses information entropy to dynamically select five features; the third step is to use selected features to calculate user's outlier degree and find out suspected user; the fourth step is to judge the user regarded as a attacker by mistake, analyze suspected users and get the intent and target items of shilling attack. After these steps, we can remove users who do not meet the intent and target items from suspected users so as to determine real attacker. we will illustrate the feasibility and superiority of this algorithm through experimental results.

## III. AN OUTLIER DEGREE SHILLING ATTACK DETECTION ALGORITHM BASED ON DYNAMIC FEATURE SELECTION

### A. Definition

**Item popularity:** the rating frequency of item in the recommender system.  $d_i$  refers to the item popularity of item i.

**User popularity vector:** a vector of the item popularity and the item has been rated by user.

$$V_u = (d_1, d_2, \dots, d_k) \quad (1)$$

Each element in user popularity vector is a user's item popularity, and k refers to item k rated by user.

**Mean of user popularity degree(MUPD):** the mean of elements in user popularity vector. The specific formula as follow:

$$MUPD_u = \frac{1}{n} \sum_{i=1}^n d_i \quad (2)$$

$d_i$  refers to the popularity of item i in user popularity vector.

**Range of user popularity degree(RUPD):** the difference between the maximum and minimum of item popularity in user popularity vector. The specific formula as follow:

$$RUPD_u = d_{max} - d_{min} \quad (3)$$

$d_{max}$  and  $d_{min}$  refer to the maximum of item popularity and the minimum of item popularity in user popularity vector.

**Attack Profiles:** In general, each attacker's rating vector consists of four parts: a set of selected items  $I_S (I_S \subset I)$ , a set of filler items  $I_F (I_F \subset I)$ , a set of target items  $I_T (\{I_T\} \subset I)$  and a set of unrated items  $I_\phi (I_\phi = I - (I_S \cup I_F \cup \{I_T\}))$ .

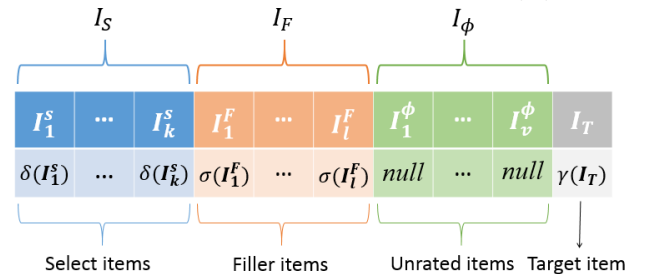


Figure 1. Attack profile vector structure

**Attack size:** the ratio of the number of attack profiles

injected into the system to the total number of user profiles.

**Filler size:** the ratio of the number of items rated by users to the total items in the system.

**Attack model:**  $M = (\chi, \delta, \sigma, \gamma)$ ,  $\chi$  refers to selection function,

$$\chi(I, U, \Phi, I_T) = \langle I_S, I_F, I_\phi, I_T \rangle$$

The function has four parameters: item set( $I$ ), user set( $U$ ), target item set( $I_T$ ), and a set of other parameters( $\Phi$ ).

**Random attacks:** rate a subset of items randomly around the overall mean vote.

**Average attacks:** rate a subset of items randomly around the mean vote of every item.

**Bandwagon attacks:** rate a subset of items randomly around the overall mean vote, and some highly popular items are rated with the maximum vote.

**Segment attacks:** rate target items highest (or lowest) score, the most relevant items with the target items highest (or lowest) score, and items filled lowest (or highest) score.

#### B. Construction of Detection Features of shilling Attacks

Based on the user popularity indicators introduced above, this paper cites the 10 user rating indicators defined in the literature [19,23], including RDMA(Rating Deviation from Mean Agreement)、WDA(Weighted Degree of Agreement)、WDMA(Weighted Deviation from Mean Agreement)、ADegSim(Average Degree of Similarity with Top Neighbors)、LengthVar(Length Variance)、FMTD(Filler Mean Target Difference)、FMV(FillerMeanVariance)、MeanVar(MeanVariance)、TMF(Target Model Focus)、SDUR (Standard Deviation in User's Ratings)、DAOU (Degree of Agreement the Other Users).

The essence of the attack detection problem is a two-class problem, namely classifying normal users and attackers in the user dataset. However, the classifier with the machine learning method have poor flexibility, and they can only work on a particular attack type. And if all the feature attributes are selected to implement the training machine learning model, the classifier will be too complex, which will seriously affect the efficiency of the classifier.

Therefore, in order to improve the flexibility of classifier, especially in the case of dealing with unknown types of attack, this paper proposes a method to dynamically select a set of feature subsets according to the training set, and then perform the attack detection based on this set of feature subsets.

The main idea of the dynamic feature selection method based on information gain is: First, calculate the index values of each feature of each user. Then, calculate the information gain of each feature by dividing the normal user and the attack user in the training set. Finally select the feature with the greatest information gain.

The feature construction algorithm is as follows:

Input: Training Set $D_t$ ,
Output: Best classification feature subset $F'$
1: Calculate the popular features of each user $u$ in the training set $D_t$ : $F_1 = \{MUPD, RUPD, QUPD\}$
2: Calculate the 10 indicators proposed in the literature[19,23] of each user $u$ in the training set $D_t$ : $F_1 = \{f_1, f_2, \dots, f_{10}\}$
3: Calculate the proportion of attacker feature values $p_{i,s}$ , $S$ is attack user set, $U$ is all user set: $p_{i,s} = \frac{\sum_{k=1}^{ S } \varphi_{f_i}(P_k)}{\sum_{j=1}^{ U } \varphi_{f_i}(P_j)}$
Here, $P_u$ denotes the user profile of user $u$ , and $\varphi_{f_i}(\ast)$ denotes calculate the feature value of $f_i$ .
4: Calculate the proportion of normal user feature values $p_{i,r}$ : $p_{i,r} = 1 - p_{i,s}$
5: Calculate the information entropy $H_i$ of the feature index $f_i$ : $H_i = -p_{i,r} * \log(p_{i,r}) - p_{i,s} * \log(p_{i,s})$
6: Compute empirical entropy of Training Set $D_t$ : $H(D_t) = -\frac{ S }{ D_t } \log \frac{ S }{ D_t } - \frac{ N }{ D_t } \log \frac{ N }{ D_t }$
Here, $N$ denotes normal user set
7: calculate empirical gain $H(D_t, f_i)$ : $H(D_t, f_i) = H(D_t) - H_i$
8: sort the features $f_i$ in descending based on the values of information gain and select top-k features: $F_2 = \{f'_1, \dots, f'_k\}$
9: Build a feature subset: $F' = F_1 + F_2$

#### C. An Outlier Degree shilling Attack User Detection Algorithm Base on Feature Vector

we get a subset of features through dynamic feature selection algorithm based on information entropy. In order to detect the attack user, this paper proposes an outlier degree detection algorithm based on feature vectors. According to the subset of features, we can get the feature vector of each user. Moreover, the features in the feature vectors are the attributes that have high classification ability for the attackers in the dataset. There is a difference between the feature profiles of normal users and attackers. Therefore, we can use user's feature vectors to determine whether he is an attacker. In this paper we use the Euclidean distance to measure the outlier degrees of the user's feature vectors, then mark user with outlier degree as a suspected user.

When it comes to the detection of attacker, we can find out attacker who deviates from the normal user's profile by his outlier degree. The specific formula of calculating outlier degree is as follow :

$$d(u) = \sum_{v \in U, v \neq u} \|V_a - V_b\|$$

$$= \sum_{b \in U, b \neq a} \left( \sum_{i=1}^n (V_{a,i} - V_{b,i})^2 \right)^{\frac{1}{2}} \quad (4)$$

$V_a$  refers to the feature vector of user  $a$ , and  $V_b$  refers to the feature vector of user  $b$ . A user whose outlier degree exceeds a certain threshold can be marked as a suspected user.

Input: The dataset of user ratings
Output: The set of attackers
<p>For <math>a</math> in all <math>U</math> do:</p> <p style="padding-left: 20px;">Construct user's feature vector.</p> <p style="padding-left: 40px;"><math>V_a = (V_{a,1}, V_{a,2}, \dots, V_{a,n})</math></p> <p>end for</p> <p>For <math>a</math> in all <math>U</math> do:</p> <p style="padding-left: 20px;">Calculate user's outlier degree:</p> <p style="padding-left: 40px;"><math display="block">\text{outlierD}_a = \sum_{b \in U, b \neq a} \left( \sum_{i=1}^n (V_{a,i} - V_{b,i})^2 \right)^{\frac{1}{2}}</math></p> <p>End for</p> <p>List users in descending order by their outlier degree;</p> <p>Select 20% users with the highest outlier degree to form suspected user set <math>S_h</math>;</p>

#### IV. EXPERIMENT

In this section, we introduce the dataset, the setup and objectives of the experiments, and analyze the experimental results.

##### A. Data description

We use the MovieLens 100K dataset in experiments, which is a popular dataset used by researchers and developers in the field of recommendation. The dataset contains ratings from 943 users on 1,682 movies. Furthermore, we write spider program to get the required data about the introduction information of movies (for item similarity) and the communication messages between users (for trust relationship). The dataset contains 19194 communication messages between 4932 users. The rating records are integers from 1 to 5.

In order to verify the accuracy of the recommendation algorithm, we use 5-fold cross validation. The attacking users in the experimental data set are generated through simulation experiments. According to the principle of the attacking attack model, artificially generated attacking user data is generated in the original data set.

##### B. Evaluation Metric

In our experiments, we first use the following evaluation indicators to determine the parameters of our methods, and then we use the indicators to analyze and compare our proposed methods with other two in literature.

In the experiment, two types of user profiles will be included, one is the real user profile and the other is the profile of attack user. Detect shilling attack users can be seen as a two-class problem. Therefore, the test results can be

represented by the confusion matrix shown in Table 1. Negative represents the real user profile, and Positive represents the profile of the attacked user.

TABLE I. CONFUSION MATRIX TABLE OF CLASSIFICATION RESULTS OF SUPPORT ATTACK DETECTION

The actual situation	The Predicted situation	
	Real user	Attack user
Real user	True Negative(TN)	False Positive(FP)
Attack user	False Negative(FN)	True Positive(TP)

This paper, we use the accuracy to evaluate the performance of shilling attack detection algorithm and accuracy formula is defined as following:

$$\text{Accuracy} = \frac{TP + TN}{TN + FP + TP + FN}$$

##### C. Feasibility analysis based on item popularity characteristics

###### 1) Analysis MUPD

MUPD can effectively partition the type of attack that there is no select item in the user attack profile vector, including the random attack model and the average attack model. Because the filled item is selected randomly in these two types of attacks, the probability of each item selected come to be equal, and the distribution of popularity of the item belongs to the long tail distribution. Therefore, the mean of user popularity vector will be very low in the general appearance of the attack users generated by the random attack model and the average attack model. As shown in Figure 2.

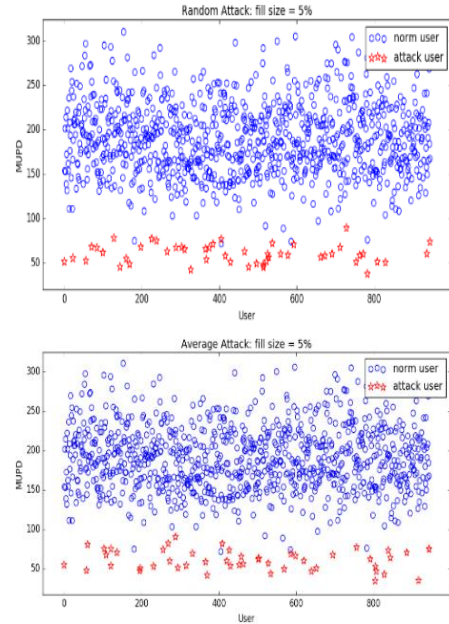


Figure 2. Distinguish the attack Model of Non-select items by MUPD

###### 2) Analysis RUPD

RUPD can effectively distinguish the popular attack model. According to the principle of popular attack model, there exist one most popular items in the attack profile. Therefore, the popularity of the attack user profile will have a great range. In

this case, MUPD can distinguish the popular attack model Invalid, as shown in Figure 3 and Figure 4:

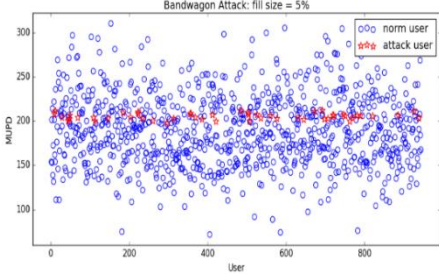


Figure 3. Identification of popular attack users using MUPD

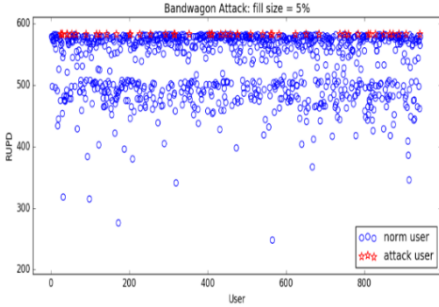


Figure 4. Identify popular attacking users using RUPD

#### D. Experimental parameter settings

TABLE II. PARAMETER SETTINGS

$S_h$	$I_h$	Filler size	Attack size	F size
20%	3	5%,10%,15%	3%,5%,10%	6

As shown in the table 2, the suspect attack user set is set to top 20%. Because of taking the cost of attack into account, the attack size won't exceed 20%. The suspect item set is set to 3, that is, the three items with the highest rating deviation are selected as the suspected attacked item set. Filler size is set to 5% or 10% or 15% respectively and Attack size is set to 3% or 5% or 10% respectively. size of Feature subset  $F'$  is 6, because according to the existing shilling attack detection algorithm, selecting three appropriate rating-based metrics can provide good detection results, so here we choose three rating-based metrics that have best detect ability, and then combine three popularity-based metrics as the result of feature subset.

#### E. Verification the Performance of shilling Attack Detection Algorithm Based on Dynamic Feature Selection and Outlier

In order to verify the performance of our method, experiment compares with the classical feature-based select method PCA-VarSelect algorithm proposed by Mehta et al. In order to distinguish this algorithm from the comparison algorithm, our method named outlier-based method, and the detection algorithm proposed by Mehta is named pca-based method.

TABLE III. ACCURACY OF RANDOM ATTACKS DETECTION RESULTS

Attack size		Filler size	Random attacks		
			5%	10%	15%
3%	outlier-Based		0.9802	0.9723	0.9910
	pca-Based		0.9246	0.9302	0.9372
5%	outlier-Based		0.9846	0.9819	0.9921
	pca-Based		0.9060	0.9390	0.9783
10%	outlier-Based		0.9967	0.9882	0.9977
	pca-Based		0.9811	0.9607	0.9513

TABLE IV. ACCURACY OF AVERAGE ATTACKS DETECTION RESULTS

Attack size		Filler size	Average attacks		
			5%	10%	15%
3%	outlier-Based		0.9853	0.9936	0.9834
	pca-Based		0.9353	0.9464	0.9177
5%	outlier-Based		0.9909	0.9845	0.9857
	pca-Based		0.9372	0.9628	0.9699
10%	outlier-Based		0.9874	0.9748	0.9850
	pca-Based		0.9528	0.9659	0.9408

As shown in Table 2 and Table 3, the outlier-based method and the pca-based method have high accuracy in the detection results of the random attacks and the average attacks. And even if the filler size and attack size is small, both methods can identify the attack user and the accuracy rate is more than 90%. However, the accuracy of outlier-based method proposed in this paper has a slightly higher than the pca-based method. Considering the combination of any attack size and filler size, the average accuracy of pac-based method is 0.9456, while outlier-based method is 0.9864. Though there are a improvement in 5%, but when the filler size is fixed, outlier-based method become relatively stable as attack size increasing, while a accuracy increased in pac-based method. What causes this phenomenon is that pac-based method, when attacker size increases, can do better in identifying the feature between attackers and normal users, but outlier-based can identify the feature very well even if attacker size is small.

TABLE V. ACCURACY OF BANDWAGON ATTACKS DETECTION RESULTS

Attack size		Filler size (selected size 5%)	Bandwagon attacks		
			5%	10%	15%
3%	outlier-Based		0.9937	0.9781	0.9970
	pca-Based		0.7033	0.7226	0.7759
5%	outlier-Based		0.9887	0.9758	0.9869
	pca-Based		0.8517	0.7639	0.8876
10%	outlier-Based		0.99546	0.9658	0.9841
	pca-Based		0.8446	0.8677	0.8701

As shown in Table 4, the pca-Based method has a lower accuracy when the attack size and filler size is smaller. Because the amount of positive and negative sample data in the data set, the effect of the pca-based method only using the user rating index is not good. For small-scale attack, outlier-based methods can be well identified. Moreover, outlier-based methods are significantly more efficient than pca-based methods for Bandwagon attack model and segmentation attack

models, which proves that the proposed attack detection method has a good efficiency. For combination of any attack size and filler size, the average accuracy of pac-based method is 0.9837 and outlier-based method is 0.8905, which there has a improvement in 22%. We conclude that when user popularity is added, outlier-based method can identify a more complex attack model, while pac-based method has no such good performance.

## V. CONCLUSION AND FUTURE WORK

This paper combines traditional score-based attack detection indicators with user popularity-based attack detection indicators to build vectors based on user popularity and average indicators with utilizing feature subsets selected by PCA based on user's average indicators; Vectors are used to calculate the degree of user's outliers, and the degree helps us mark the outlier users as suspects. Considering odd-looking users in system, we can find out the real attacker by analyzing the score of suspects, judging the intent of attacker and removing the users who dissatisfy the intent.

Through comparing experiment results, we can see the outlier-based attack detection algorithm based on dynamic feature selection has high accuracy and can be adapted to the different attack models flexibly in the system.

In the future work we will: (1) Finding the attack detection features from other perspectives, (2) Integrating existing feature indicators more effectively to find out more feature indicators, (3) building the attack defense from two levels by combining the attack detection method and attack defense robustness algorithm

## ACKNOWLEDGMENT

The research is supported by "National Natural Science Foundation of China" (No. 61772560) and the scientific research "Innovation Project for Graduate Students in Central South University" (No. 1053320170318).

## REFERENCES

[1] Cerqueira, Thaciana, L. Marinho, and F. Ramalho. "A Content-Based Approach for Recommending UML Sequence Diagrams." SEKE 2016.

[2] J. Ben Schafer, Dan Frankowski, Jon Herlocker, et al. Collaborative Filtering Recommender Systems[J]. *Acm Transactions on Information Systems*, 2007, 22(1):5-53.

[3] Felfernig A, Gula B, Leitner G, et al. Persuasion in Knowledge-Based Recommendation[C]// International Conference on Persuasive Technology. Springer-Verlag, 2008:71-82.

[4] Zhang, Chi, G. Chen, and H. M. Wang. "Recommendation Model Based on Blending Recommendation Technology." *Computer Engineering* 36.22(2010):248-250.

[5] Bartolini I, Zhang Z, Papadias D. Collaborative filtering with personalized skylines[J]. *Knowledge and Data Engineering, IEEE Transactions on*, 2011, 23(2): 190-203.

[6] Barragáns-Martínez B, Costa-Montenegro E, Juncal-Martínez J. Developing a recommender system in a consumer electronic device[J]. *Expert Systems with Applications*, 2015, 42(9):4216-4228.

[7] Yan, W. U., et al. "Algorithm for Sparse Problem in Collaborative Filtering." *Application Research of Computers* 24.6(2007):94-97.

[8] CHEN Zong-yan, Yan jun. "Collaborative Filtering Recommendation Algorithm Based on Sparse Data Pre-processing". *The computer technology and development*, 2016, 26(7):59-64.

[9] Yang, Yu, et al. "Cold-Start Developer Recommendation in Software Crowdsourcing: A Topic Sampling Approach." *The, International Conference on Software Engineering and Knowledge Engineering 2017*:376-381.

[10] Zhang, Fu Guo, and X. U. Sheng-Hua. "Review of key security threats and countermeasures in recommender systems." *Application Research of Computers* 25.3(2008):656-659.

[11] Xiang, X. U. "Analysis of shilling attacks on SVD-based collaborative filtering algorithm." *Computer Engineering & Applications* 45.20(2009):92-95.

[12] Huang, Sheng, M. Shang, and S. Cai. "A Hybrid Decision Approach to Detect Profile Injection Attacks in Collaborative Recommender Systems." *International Symposium on Methodologies for Intelligent Systems* Springer, Berlin, Heidelberg, 2012:377-386.

[13] Zhi-Ang, W. U., et al. "Shilling Attack Detection Based on Feature Selection for Recommendation Systems." *Acta Electronica Sinica* 40.8(2012):1687-1693.

[14] Chirita P A, Nejd W, Zamfir C. Preventing Shilling Attacks in Online Recommender Systems[C]//Proceedings of the 7th Annual ACM International Workshop on Web Information and Data Management. New York: ACM, 2005: 67-74.

[15] Mehta B. Unsupervised Shilling Detection for Collaborative Filtering[C]//Proceedings of the 22nd National Conference on Artificial intelligence. Menlo Park, London: AAAI, 2007: 1402.

[16] Mehta B, Hofmann T, Fankhauser P. Lies and Propaganda: Detecting Spam Users in Collaborative Filtering[C]//Proceedings of the 12th International Conference on Intelligent User Interfaces. New York: ACM, 2007: 14-21.

[17] Li, Cong, Z. G. Luo, and J. L. Shi. "An Unsupervised Algorithm for Detecting Shilling Attacks on Recommender Systems." *Acta Automatica Sinica* 37.2(2011):160-167.

[18] Chung C Y, Hsu P Y, Huang S H. βP: A Novel Approach to Filter out Malicious Rating Profiles from Recommender Systems[J]. *Decision Support Systems*, 2013, 55(1): 314-325.

[19] Burke R, Mobasher B, Williams C, et al. Classification features for attack detection in collaborative recommender systems[C]// ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2006:542-547.

[20] Wang J K, JIANG Y C, Sun J S, SUN C H." Item-based Collaborative Filtering Algorithm Integrating User Activity and Item Popularity". *Computer science*. Vol. 43. No.12.pp. 158-162.

[21] Li, W. T., et al. "An shilling attack detection algorithm based on popularity degree features." *Zidonghua Xuebao/acta Automatica Sinica* 41.9(2011):1563-1576.

[22] Chengshu, L. "SHILLING ATTACK DETECTION BASED ON FEATURE SELECTION AND SVM." *Computer Applications & Software* (2015).

[23] Williams C A, Research Advisor, Mobasher B. Thesis: Profile Injection Attack Detection for Securing Collaborative Recommender Svstems[J]. *Service Oriented Computing & Applications*, 2012, 1(3):157-170.

# Service Language Model: New Ecology for Service Development

Ying Li

College of Computer Science and  
Technology  
Zhejiang University  
Hangzhou, China  
cnliying@zju.edu.cn

Meng Xi

College of Computer Science and  
Technology  
Zhejiang University  
Hangzhou, China  
ximeng@zju.edu.cn

Hui Chen

College of Computer Science and  
Technology  
Zhejiang University  
Hangzhou, China  
chchenhui@zju.edu.cn

\*Jianwei Yin

College of Computer Science and  
Technology  
Zhejiang University  
Hangzhou, China  
zjuyjw@zju.edu.cn

**Abstract**—With rapid development of the Internet, all walks of life are engaged in the tide of Internet. In the era of "Internet+", traditional industries are widely developed and expand plenty of emerging business, such as online transactions, Internet finance and so on. However, various problems arise at the same time in this revolution. On one hand, business processes become increasingly intricate, and different fields may have difficulty in communication. On the other hand, developers are hard to understand the real demands from users and the rate of code reuse is not high. In order to solve these problems, we propose a middle-end and project manager (PM) oriented service language model, which could help decouple software development and user requirements, improve work efficiency and reduce development costs.

**Keywords**—service language model, ontology, business process

## I. INTRODUCTION

Nowadays, modern service industry has grown up rapidly and play an important role in our daily life. As service scenarios become increasingly complicated and user demands change quickly, traditional business process modeling methods can hardly meet the current business requirements.

The problem could be serious in the field of E-commerce. Different from offline sales, business processes in E-commerce develop rapidly. To cope with this situation, enterprises need to invest a lot of resources to improve their original business processes. Therefore, it is necessary to design a process modeling method that can quickly and accurately respond to requirements. And in recent years, data-centric Artifact model has emerged. Different from traditional activity-centric modeling approaches, the construction of Artifact model begins with data, and then business processes are built around the data entities. By using data-centric approach, the business process could be flexible and adaptable. However, the data-centric life cycle could be difficult to reuse and easily lead to code redundancy. On the other hand, the existing demand analysis method is usually activity-centered, which makes it difficult to put Artifact modeling into practice.

In this paper, we design a service language model (SLM). SLM consists of Service Concept Model (SCM) that can describe the requirements of services and Service Design Model (SDM) that can construct and manage service processes. Within SDM, there is an entity feature model for data, an

ability feature model for functions and interfaces and a refined life cycle model for business processes.

The main contributions of SLM are as follows. First, a concept model of service is designed to conceptualize requirements and describe them in a formal way. Secondly, a method was provided to manage and deploy data and data entities in services. Thirdly, Business abilities can be reused and their range can be cleared easily by using SLM. Finally, a process model which could help stratify the business and clarify the business structure was proposed.

The rest of this article is organized as follows. Section Two is the motivation case. Section Three introduces SLM systematically. Section Four illustrates an application instance where the model is verified. Finally, the related works and conclusions are given.

## II. MOTIVATION CASE

In order to have more in-depth research and better understanding of the problem, we have worked with Alibaba company. Here, we take buying basketball shoes and virtual coupons as an example. One is physical goods trading which is one of the earliest services of Alibaba, and the other is virtual goods trading which is emerging these years. The former one involves goods delivery, while the latter only needs to send some verification codes. Since the physical properties of these two commodities are different, their service processes should be differentiated. Moreover, when modeling, we may encounter the following questions.

Problem 1. Business logic is coupled with platform logic, and process activity code is difficult to strip and reuse. This makes the platform code difficult to analyze and organize. With the business becoming more complex, platform codes may have redundancy;

Problem 2. Long implementation cycle. A simple need also requires steps such as program evaluation, requirement analysis, process design, process development, and regression testing;

Problem 3. Internal business units lack knowledge of business uniformity, communication between departments can cost a lot of time.

---

\*: corresponding author

DOI reference number: 10.18293/SEKE2018-214

### III. SERVICE LANGUAGE MODEL

In this chapter, we will introduce our service language model systematically. The main concepts in service concept model include entity rules and routing rules, and the main concepts in service design model include entity, ability, essence process, and service process. For a better understanding of the definitions below, we expect the existence of the following pairwise disjoint countably infinite sets:  $T_p$  of primitive types,  $C$  of (artifact) classes (names),  $A$  of attributes (names),  $S$  of artifact states, and  $ID_C$  of (artifact) identifiers for each class  $C$ . A type is an element in the union  $T = T_p \cup C$ .

#### A. Service Concept Model

As we all know, requirement analysis has always been extremely important in software engineering, as its quality directly affects the effectiveness of the later system design. In order to make requirement analysis more concise and efficient, we build the concept of service model.

**Definition 1.** An service concept model instance is a triple  $(sc, Rk, Rr)$ , where  $sc$  is the identifier,  $Rk$  is the entity concept model involved and  $Rr$  is the routing concept model.

The entity concept model is mainly used to specify the terms involved in the service. It can express the domain terminology, attributes of each term, and links among different terms. In addition, the entity concept model can generate the entity feature model by semi-automatic methods.

**Definition 2.** An entity concept class is a four-tuple  $(rkc, k, d, re)$ , where  $rkc$  is the identifier,  $k$  is the set of term names,  $d$  is the set of term descriptions and  $re$  is the set of relationships among terms.

**Definition 3.** An instance of an entity concept is a 3-tuple  $(rk, kd, krk)$  where  $rk$  is an identifier,  $kd$  is a partial mapping from  $k$  to  $d$ , and  $krk$  is a set of relations mapping between terms by means of "k-re-k".

**Example 1.** In the motivation case, business side needs to provide the requirement of data that may be used in their service, like receiving information which consists of name, phone and address. They could describe this requirement in the format of service language model as shown in Table I.

TABLE I. ENTITY CONCEPT EXAMPLE: RECEIVING INFORMATION ENTITY CONCEPT

ID	Entity & Description	Relation
721019	receiving information: data set name: string, length<10 phone: number, length=11 address: string	basic(receiving information, name) basic(receiving information, phone) basic(receiving information, address)

On the other hand, routing rules are mainly used to express the system through a simple logic involved in the activities of the process. The conceptual model of routing mainly guides the construction of the refinement process model, and gives guidance on the construction of the life cycle model.

**Definition 4.** A routing concept class is a four-tuple  $(rrc, S, c, E)$ , where  $rrc$  is the identifier,  $S$  is the state in the route,  $C$  is the set of judgement conditions, and  $E$  is the set of posterior effects.

**Definition 5.** A routing concept instance is a pair  $(rr, sce)$ , where  $rr$  is the identifier, and  $sce$  indicates that the entity  $e$  satisfies the condition  $c$  under state  $s$ .

**Example 2.** In the motivation case, there are requirements about process itself like how to place an order. The business side needs to arrange the process and describe it in the format of routing concept like Table II.

TABLE II. ROUTING CONCEPT EXAMPLE: PLACE AN ORDER ROUTING CONCEPT

ID	sce
217210	(no_order, place an order success, state=to_pay) (to_pay, payoff, balance reduce & state=to_deliver) ...

#### B. Entity Feature Model

The entity feature model is mainly used to model the data entities in the service. A feature model is composed of a set of characteristics and their relationships, and it also has certain constraints.

**Definition 6.** An entity feature class is a 7-tuple  $(C_e, AT, \tau, Q, s, F, Opt)$ , where  $C_e$  is the identifier,  $AT$  is the attribute in the entity or the characteristic in the entity,  $\tau$  is the type of the attribute or the constraint relation of the class or feature,  $Q$  is the set of states of the entity,  $s$  is the initial state of the entity,  $F$  is the end state, and  $Opt$  is the optionality of the state.

**Definition 7.** An entity feature instance is a triple  $(e, \mu, q)$  where  $e$  is the identifier,  $\mu$  is partial mapping that assigns each  $AT$  in  $T$  (mentioned at the begging of section three),  $q$  is current state when  $e$  is an full entity or  $Opt$  when  $e$  is a feature of an entity.

**Example 3.** In the motivation case, order is an important entity. The process of a transaction is the lifecycle of an order which is from generation to extinction. Its instance is presented in Table III.

TABLE III. ENTITY CONCEPT EXAMPLE: RECEIVING INFORMATION ENTITY CONCEPT

ID	$\mu$	state
171701	receiving information: data set name: string, length<10 phone: number, length=11 address: string	basic(receiving information, name) basic(receiving information, phone) basic(receiving information, address)

When analyzing the transaction process, we find that there are entities involve in a process with no state, which are called participants.

**Definition 8.** A participant class is a triplet  $(C_p, RE, \tau)$  where  $C_p$  is the identifier,  $RE$  is the resource of the participant, and  $\tau$  is the main type of the resource.



**Definition 9.** A participant instance is a binary  $(p, \mu)$ ,  $P$  is the identifier,  $\mu$  is the partial mapping that assigns each RE in  $T$ .

### C. Ability Feature Model

The modeling method of ability feature model is the same as that of the entity model, which is used to describe the configurable services and functions of Artifact.

Through the construction of the ability model, we can use functional units in the system as configurable atomic services. Through the call to the service, data in entities can be processed and their state may transfer as well.

**Definition 10.** The instance of the ability is a six-tuple  $(ab, D, VEr, VEw, VPr, VPw)$ ,  $ab$  is the identifier,  $d$  is description of this ability,  $VEr$  is variable of entity to read,  $VEw$  is variable of entity to write,  $VPr$  is variable of participant to read,  $VPw$  is variable of participant to write.

**Example 4.** In the motivation case, we need an ability of creating an order. This ability need to generate an order and transfer the customer's money to third party platform. We need the order's id and whether the transfer succeed. The ability is presented in Table IV.

TABLE IV. ABILITY EXAMPLE: ABILITY OF CREATING AN ORDER

order_create	
id	344323
description	create an order and transfer the customer's money
entity	order
participant	customer
entity read	order.merchandise, order.destination, order.message, order.seller
entity write	order.id
participant read	customer.user_id, customer.cash, customer.method
participant write	customer.payment_result

### D. Refined Lifecycle Model

The refined lifecycle model is mainly used to represent the changes in state and the activities used during the execution of the service. Refinement lifecycle model can stratify the processes in the lifecycle process, decouple the processes in different areas and reduce the complexity of each process while improving the process complexity.

Condition is essentially a discriminant expression, and finally return Boolean values, which could complete some simple calculations and judgments.

**Definition 11.** A condition is a Boolean expression, which could be connected and calculated by logical operators.

L1 process describes the core functions of the business and is used for service classification and location.

**Definition 12.** An L1 process is a triple  $(m, d, RP2)$ ,  $m$  is the identifier,  $d$  is the model description and  $RP2$  is the L2 process inherited from this L1.

L2 inherits from the L1 process and represents all the states that an entity will experience in a life cycle.

**Definition 13.** An L2 and a process is a five-tuple  $(tr, d, q, r, RP3)$ ,  $tr$  is identifier of transition,  $d$  is description,  $q$  is states and  $r$  is relation between states.

The L3 process inherits from L2 and adds activities and gateway nodes on the basis of L2, which can represent all the activities and states that an entity experiences in a process. At the same time in the L3 process we will complete the ability to assemble. We specify each activity to assemble an ability.

**Definition 14.** An L3 process is a six-tuple  $(re, d, q, a, g, r)$ ,  $re$  is identifier,  $d$  is description,  $q$  is the states,  $a$  is activities,  $g$  is gateways and  $r$  is relations connecting  $q, a$  and  $g$ .

**Definition 15.** Activity instance is a four-tuple  $(a, ab, P, E)$ ,  $a$  is identifier,  $ab$  is identifier of ability,  $P$  is pre-condition (atom), and  $E$  is effect.

**Definition 16.** Gateway instance is pair  $(g, type)$ ,  $g$  is identifier of gateway and  $type$  could be one of the four: Exclusive, Inclusive, Complex and Parallel.

**Definition 17.** relation is a four-tuple  $(r, from, to, c)$ ,  $r$  is identifier of relation,  $from / to$  is state / activity / gateway, and  $c$  is a condition.

Then we can add the activities of the preconditions and configuration items to generate L4 process on the basic of L3 process. And a runnable process is complete.

## IV. APPLICATION INSTANCE

Here we take the physical transaction service as an example to explain how to use the service model to construct it. Due to the space limitation, we only make a brief explanation.

### A. Constructing service concept model

The physical transaction process is mainly around the order, which has a clear state in each stage of the process. The data in the entity can be described as "entity data name + relationship + limit value". For instance, if the entity "order" contains "delivery method", its value should be one of "general delivery", "free shipping", "cash on delivery" and "self-pickup". All the requirements for the entities in the process can be described by the entity concept model.

The process may involve other participating entities as well, like goods and members. In this process, the participating entities do not own state, but the data involved in the implementation of the process. Then we could describe the business process and routing rules through a set of combination of state, condition and effect, such as "unpaid, payment executed, state change to unshipped".

### B. Constructing service design model

The service design model is a system implementation of the service concept model. From the service concept model to the service design model there is a mapping relationship, and these two models can manually transform into each other semi-automatically.

By reorganizing and rearranging entity concept models, we can construct entity feature models. That is, modeling the

entity through feature modeling. We modeled three entities in the physical transaction process respectively including "Order", "Goods" and "Customer".

Then, we encapsulate functions and interfaces that may be used in the process and list the abilities. According to the classification criteria of the e-commerce domain characteristics, the classification of abilities could be performed. An ability feature model is similar to entity feature model. In this way, we can quickly query the ability to operate on the entity data involved in the process and assemble it.

The refined process model is used finally. Firstly, we need to determine which trading domain the business belongs to, and that is L1 process. Then we further determine the states of the order, which could be "unpaid", "unshipped", "assessed" and etc. L2 process is constructed by these states and their transition relationship. L3 need to add activities and gateways which may cause the transition of states. For instance, the transition from "unpaid" to "unshipped" needs to go through activity "payment". The L4 level process is the configuration of the preconditions, transition conditions between activities and other configuration items. Through these operations, a physical transaction business is completed.

## V. RELATED WORK

The service language model is based on previous research. We mainly refer to research results of ontology, domain-specific language (DSL), variability modeling and business process modeling (BPM).

The concept of ontology originated from the field of philosophy and later has been given a new definition with the development of computer science. Ontology web language (OWL) is a representative application of ontology in computer science. Different from traditional knowledge-based methods that are represented by formal normative language, OWL is more suitable for regulating demands and domain knowledge [1]. In addition, it has the ability of automatic verification and consistency checks [2]. Nevertheless, OWL only stays in the concept level, which means it is unable to implement a system.

DSL has been recognized as a normative executable language that is highly expressive in specific areas, where domain contents are abstracted [3]. Domain experts can easily design models with the help of DSL. Thus, it is applied to a number of fields. For example, Matlab [4] in the mathematical field can help people deal with vector, matrix and other mathematical operations conveniently. On the other hand, the IBM Sharable Code project also uses DSL to prepare services [5]. However, DSL is only applicable to certain areas, which means it has difficulty in solving cross-domain problems.

Variability modeling is the key technology for variability management in software product lines. In our project, we mainly refer to the feature-based variability modeling method to construct our entity feature model. This method was first introduced in 1990 by Kang KC et al. [6]. In recent years, since feature modeling technique can greatly improve the flexibility and reusability of system, it has been widely used to describe relevant characteristics of software product lines and to manage reusable assets in systems.

There are many business process modeling methods. Business process modeling notation (BPMN) is a basic and practical standard in web service field[7] since it can intuitively express business processes. It defines a variety of elements that may be used in business process modeling, including activities, gateways, events and so on. Business process execution language (BPEL) is an abstract high-level language that can clearly describe functions and services provided by each web, as well as protocols among different activities [8]. However, these methods cannot eliminate the redundancy of large complex systems, since they always focus on business process implementation level.

## VI. CONCLUSION

On the basis of artifact centric process modeling, this research improves the modeling method of data entities to make it more adaptable to the operating environment of large and complex systems. We construct a hierarchical lifecycle model that can help organize and manage the process. Ability feature model is proposed to manage and reuse the code resources more effectively and efficiently. We also modeled requirements into SCM which could help build a domain knowledge and describe requirements in a formal way. At present, we have been able to realize semi-automatic conversion between requirements and design under artificial operation. As for future work, we will introduce natural language processing and machine learning methods to achieve automatic conversion from requirements to code.

## ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China under Grant No.61772459, the National Key Research and Development Program of China under Grant No.2017YFB1401202 and the Key Research and Development Program of Zhejiang Province under Grant No.2017C01013.

## REFERENCES

- [1] Wouters B, Deridder D, Van Paesschen E. The use of ontologies as a backbone for use case management[C]//European Conference on Object-Oriented Programming (ECOOP 2000), Workshop: Objects and Classifications, a natural convergence. 2000, 182.
- [2] Happel H J, Seedorf S. Applications of ontologies in software engineering[C]//Proc. of Workshop on Semantic Web Enabled Software Engineering"(SWESE) on the ISWC. 2006: 5-9.
- [3] Van Deursen A, Klint P, Visser J. Domain-specific languages: An annotated bibliography[J]. ACM Sigplan Notices, 2000, 35(6): 26-36.
- [4] Hanselman D, Littlefield B. Mastering MATLAB 6: a comprehensive tutorial and reference[M]. Pearson, 2001.
- [5] Maximilien E M, Ranabahu A, Gomadam K. An online platform for web apis and service mashups[J]. IEEE Internet Computing, 2008, 12(5).
- [6] Kang K C, Cohen S G, Hess J A, et al. Feature-oriented domain analysis (FODA) feasibility study[R]. Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, 1990.
- [7] White S A. Introduction to BPMN[J]. IBM Cooperation, 2004, 2(0): 0.
- [8] Fu X, Bultan T, Su J. Analysis of interacting BPEL web services[C]//Proceedings of the 13th international conference on World Wide Web. ACM, 2004: 621-630.

# A Knowledge Engineering Approach to UML Modeling

Bingyang Wei

Department of Computer Science  
Texas Christian University  
Fort Worth, Texas 76129  
b.wei@tcu.edu

Jing Sun

Department of Computer Science  
The University of Auckland  
Auckland 1142, New Zealand  
j.sun@cs.auckland.ac.nz

Yi Wang

Department of Electrical  
and Computer Engineering  
Manhattan College  
Bronx, NY, 10471  
yi.wang@manhattan.edu

**Abstract**—Multiple-viewed requirements modeling allows requirement engineers to acquire the requirements of a system from different perspectives. Requirements are then specified in various UML models. Maintaining the requirements knowledge encoded in UML notations is tedious and error-prone, since most UML CASE tools provide poor support for reasoning and query. Ontology is a formal notation for describing concepts and their relations in a domain. Since requirement is a kind of knowledge, we propose to use knowledge engineering approach for managing the consistency and completeness of UML models. In this paper, an ontology for UML diagrams is coded in a semantic web language, OWL (Web Ontology Language). The transformation of UML Class Diagram, Sequence Diagram and State Diagram to OWL knowledge base is presented. In the end, a semantic query language, SPARQL, is used to query the knowledge base. We demonstrate the feasibility of this approach through an example software system.

## I. INTRODUCTION

Requirements engineers usually views the system under development from different perspectives. The structure, behavior and interaction aspects of the system are among the most commonly considered perspectives. UML (Unified Modeling Language) makes this multiple-viewed modeling possible by providing different types of models, e.g., class, state machine and interaction models. As a result, each UML model holds partial requirements from a particular view in the form of UML diagrams, and all the models together constitute the overall description of the system. Since software requirement about a domain is indeed a kind of knowledge, the Requirements Engineering process is in fact a kind of Knowledge Engineering process. An important concern of requirements engineers during multiple-viewed modeling is the management of requirements knowledge: consistency among different models, model query, acquisition of enough useful requirements to make a model more complete. However, it is difficult for a requirements engineer to know whether a model is consistent or complete and what requirements are missing in the current model [4][5].

The application of ontology in Requirements Engineering to support elicitation, analysis, specification, validation and management of requirements is one of the solutions to improve the quality of requirements [2]. To be more specific,

knowledge representations based on formal logic is considered as an effective way to represent and manage requirements knowledge [3] [7] [10]. In this paper, we are combining the popularity of the semi-formal UML notations and the Description Logic based OWL (Web Ontology Language), so that UML diagrams' semantics can be represented in OWL. Our main objective is to add a semantic layer on top of different types of UML diagrams. Different UML diagrams, although contain different kinds of requirements, can be queried and reasoned together for different purposes.

The remainder of the paper is organized as follows. In Section II, UML metaclasses are organized in a taxonomy and encoded in OWL. Section III to V present the way we transform an example system's UML models to OWL individuals and relations based on the UML Ontology. With the UML diagrams represented in OWL, query and analysis are conducted in Section VI. Section VII discusses the related work and current limitations, and Section VIII concludes the paper.

## II. THE UML ONTOLOGY

The Object Management Group published the latest UML Specification, *OMG<sup>®</sup> Unified Modeling Language<sup>®</sup> version 2.5.1* [6] in December, 2017. We extracted the taxonomy of UML model elements according to *Chapter 7 Common Structure, Chapter 8 Values, Chapter 9 Classification, Chapter 10 Simple Classifiers, Chapter 11 Structured Classifiers, Chapter 13 Common Behavior, Chapter 14 StateMachines, Chapter 15 Activities, Chapter 16 Actions and Chapter 17 Interactions*. 127 metaclasses and their inheritance relationships are organized in the class hierarchy in Protégé [8]. The complete ontology in RDF/XML syntax is available at [github.com/Washingtonwei/uml-ontology](https://github.com/Washingtonwei/uml-ontology).

## III. UML CLASS DIAGRAM IN OWL

In order to demonstrate the idea of transforming UML models to OWL notations, we choose an example software system, the University Information System (UnivSys). In this example, UML models for a university seminar enrollment service system are created: Seminars can be created, scheduled, opened, enrolled, and closed; students can enroll in or

drop a seminar if a certain requirement is met. All three UML diagrams come from Ambler’s book [1] with modifications.

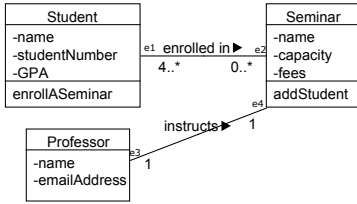


Fig. 1. Class diagram of UnivSys.

A UML model consists of a number of model elements, each of which may be used to make statements about different kinds of individual things within the system being modeled [6]. In Protégé, individuals are created and linked together to represent the semantics of a UML model. Since all the metaclasses of model elements belonging to UML class model are considered in the UML Ontology defined in the previous section, the UnivSys class diagram in Figure 1 can be easily represented as OWL individuals of various UML metaclass types. Those OWL individuals are then connected through predefined relations to form the meaning of the class diagram. Table I and II list the OWL individuals and object properties (relations between two OWL classes) extracted from the UnivSys class diagram. Here is a brief summary: each **Class** individual owns **Property** individuals as attributes and **Operation** individuals as operations. An **Association** individual owns two **Property** individuals as member ends. Each of them has a **Type** individual.

TABLE I  
INDIVIDUALS CREATED FOR UML CLASS DIAGRAM

Individual(s)	UML Metaclass
StudentClass, SeminarClass, ProfessorClass	<b>Class</b>
studentName, studentNumber, GPA, seminarName, capacity, fees, professorName, emailAddress, associationEnd1, ..., associationEnd4	<b>Property</b>
enrolledInAssociation, instructsAssociation	<b>Association</b>
enrollASeminarOperation, addStudent	<b>Operation</b>

TABLE II  
RELATIONS FOR UML CLASS DIAGRAM

Relation	Domains	Ranges
ownedAttribute	<b>Class</b>	<b>Property</b>
ownedOperation	<b>Class</b>	<b>Operation</b>
memberEnd	<b>Association</b>	<b>Property</b>
type	<b>TypedElement</b>	<b>Type</b>

#### IV. UML SEQUENCE DIAGRAM IN OWL

In this section, we focus on transforming UML Sequence Diagram (Figure 2) to OWL.

OWL individuals identified in the UnivSys sequence diagram, their corresponding UML metaclasses, and predefined relations are listed in Table III and Table IV, respectively. Each **Lifeline** individual represents a **ConnectableElement**

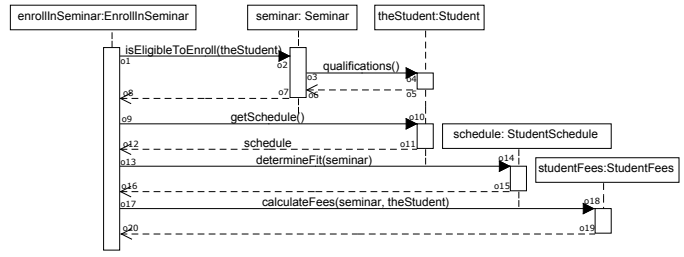


Fig. 2. A snippet of sequence diagram of UnivSys.

TABLE III  
INDIVIDUALS CREATED FOR UML SEQUENCE DIAGRAM

Individual(s)	UML Metaclass
enrollInSeminarLifeline, seminarLifeline, theStudentLifeline, scheduleLifeline, studentFeesLifeline	<b>Lifeline</b>
connectableElement1, connectableElement2, connectableElement3, connectableElement4, connectableElement5	<b>ConnectableElement</b>
EnrollInSeminar, Seminar, Student, StudentSchedule, StudentFees	<b>Type</b>
o1, o2, o3, o4, ..., o19, o20	<b>MessageOccurrenceSpecification</b>
isEligibleToEnrollMessage, qualificationsMessage, qualificationsReply, isEligibleToEnrollReply, getScheduleMessage, getScheduleReply, determineFitMessage, determineFitReply, calculateFeesMessage, calculateFeesReply	<b>Message</b>
theStudentInstanceValue, seminarInstanceValue	<b>InstanceValue</b>
isEligibleToEnrollOperation, qualificationOperation, getScheduleOperation, determineFitOperation, calculateFeesOperation	<b>Operation</b>
theStudentInstance, seminarInstance	<b>InstanceSpecification</b>

individual whose **Type** individual is specified by relation type. Each **Lifeline** individual owns a number of **MessageOccurrenceSpecification** individuals through relation events. Those occurrences represent the send or receive event occurrences, which are then linked to **Message** individuals. A **Message** individual specifies the content through signature and argument. The signature can be either **Operation** or **Signal**. Both of them may include **ValueSpecification** individuals as arguments. In this particular sequence diagram, each **ValueSpecification** individual is related to a **InstanceSpecification**, for example, theStudentInstance individual and seminarInstance individual. Besides the relations between OWL individuals, two data properties for **Message** individuals are considered: messageKind (“complete”, “found”, “lost”, “unknown”) and messageSort (“asynchCall”, “asynchSignal”, “createMessage”, “deleteMessage”, “reply”, “synchCall”).

TABLE IV  
RELATIONS FOR UML SEQUENCE DIAGRAM

Relation	Domains	Ranges
events	<b>Lifeline</b>	<b>Occurrence-Specification</b>
represents	<b>Lifeline</b>	<b>ConnectableElement</b>
signature	<b>Message</b>	<b>NamedElement</b>
argument	<b>Message</b>	<b>ValueSpecification</b>
sendEvent	<b>Message</b>	<b>MessageEnd</b>
receiveEvent	<b>Message</b>	<b>MessageEnd</b>
type	<b>TypedElement</b>	<b>Type</b>
instance	<b>InstanceValue</b>	<b>Instance-Specification</b>
classifier	<b>Instance-Specification</b>	<b>Classifier</b>

V. UML STATE DIAGRAM IN OWL

Figure 3 illustrates the behavior of a seminar during its lifetime. Based on the UML Specification, the semantics of the state diagram is represented in Protégé in the form of individuals (Table V) and their relations (Table VI). **Transaction** plays an important role in capturing the meaning of a state machine diagram. Each **Transaction** individual connects to two **State** individuals by sourceVertex and targetVertex. A transaction owns a **Trigger** individual that is related to an **Event** individual, a **Constraint** individual as guard and a **Behavior** individual as effect. Every constraint is related to a **ValueSpecification** individual by constrainedElement relation, and has a specification that evaluates to a Boolean value. Each **State** individual may own behaviors as its entry, exit and doActivity. Besides the relations between OWL individuals, one data property for **PseudoState** instances is considered: pseudostateKind (“choice”, “deepHistory”, “entryPoint”, “exitPoint”, “fork”, “initial”, “join”, “junction”, “shallowHistory”, “terminate”); another data property for **Transition** instances is considered: transitionKind(“external”, “internal”, “local”).

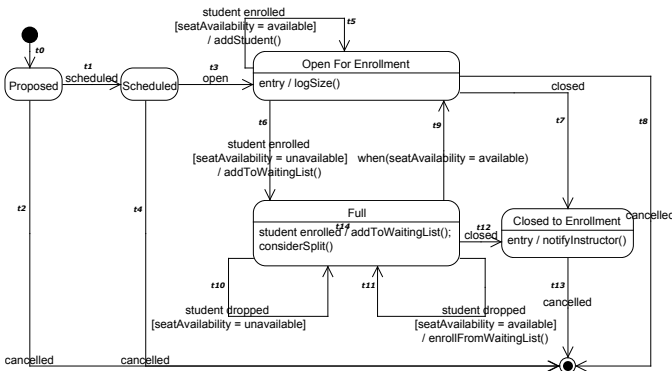


Fig. 3. State diagram of a seminar.

VI. QUERYING UML MODELS USING SPARQL

Section III to V convert University Information System’s three UML models to OWL individuals. This process results in 127 distinct OWL individuals. Interested readers can find

TABLE V  
INDIVIDUALS CREATED FOR UML STATE DIAGRAM

Individual(s)	UML Metaclass
InitialState, ProposedState, ScheduledState, OpenForEnrollmentState, ClosedToEnrollmentState, FinalState	<b>State</b>
transition0, transition1, transition2, ..., transition13, transition14	<b>Transition</b>
completionTrigger, scheduledTrigger, openTrigger, studentEnrolledTrigger, studentDroppedTrigger, closedTrigger, cancelledTrigger, whenSeatAvailableTrigger	<b>Trigger</b>
completionEvent, scheduledEvent, openEvent, studentEnrolledEvent, studentDroppedEvent, closedEvent, cancelledEvent, whenSeatAvailableEvent	<b>CallEvent</b>
seatAvailableConstraint, seatUnavailableConstraint	<b>Constraint</b>
seatAvailableExpression, seatUnavailableExpression	<b>OpaqueExpression</b>
addStudent, logSize, addToWaitingList, notifyInstructor, enrollFromWaitingList, considerSplit	<b>OpaqueBehavior</b>
addStudentOperation, logSizeOperation, addToWaitingListOperation, notifyInstructorOperation, enrollFromWaitingListOperation, considerSplitOperation	<b>Operation</b>
seatAvailability	<b>Element</b>

TABLE VI  
RELATIONS FOR UML STATE DIAGRAM

Relation	Domains	Ranges
sourceVertex	<b>Transition</b>	<b>Vertex</b>
targetVertex	<b>Transition</b>	<b>Vertex</b>
trigger	<b>Transition</b>	<b>Trigger</b>
guard	<b>Transition</b>	<b>Constraint</b>
effect	<b>Transition</b>	<b>Behavior</b>
entry	<b>State</b>	<b>Behavior</b>
exit	<b>State</b>	<b>Behavior</b>
doActivity	<b>State</b>	<b>Behavior</b>
event	<b>Trigger</b>	<b>Event</b>
changeExpression	<b>ChangeEvent</b>	<b>ValueSpecification</b>
specification	<b>Constraint</b>	<b>ValueSpecification</b>
constrainedElement	<b>Constraint</b>	<b>ValueSpecification</b>
method	<b>BehavioralFeature</b>	<b>Behavior</b>

those individuals at [github.com/Washingtonwei/uml-ontology](https://github.com/Washingtonwei/uml-ontology). With the three UML diagrams properly represented in OWL, query can be conducted using SPARQL. SPARQL (SPARQL Protocol and RDF Query Language) is a semantic query language. It is an effective way to retrieve and manipulate data stored in RDF format. The underlying structure of an OWL knowledge base is a collection of triples, each consisting of a subject, a predicate and an object. For example, the description of “a class A owns a property B as an attribute” is stored in OWL as a triple: *A ownedAttribute B*. In a SPARQL query, there are several important parts: a SPARQL variable starts with a question mark and can match any individual in the OWL knowledge base. In the WHERE clause, triple patterns

are defined in which any parts can be replaced with a SPARQL variable. The SELECT result clause returns a table of variables and values that satisfy the query. Protégé has built-in support for SPARQL. One sample query is presented here.

- “What are the associated classes of SeminarClass?”

```
PREFIX
:<http://www.semanticweb.org/uml-ontology#>
SELECT ?associatedClass
WHERE {
  ?end1 :type :SeminarClass .
  ?association :memberEnd ?end1 .
  ?association :memberEnd ?end2 .
  ?end2 :type ?associatedClass .
  FILTER(?associatedClass!=:SeminarClass)
}
```

Explanation: Select any individual such that it is the type of an end (:end2), which is a member end (:memberEnd) of an association (?association), whose the other end has type SeminarClass (:SeminarClass). The FILTER makes sure that we are looking for a class other than SeminarClass.

Answers from Protégé:

- <http://www.semanticweb.org/uml-ontology#StudentClass>
- <http://www.semanticweb.org/uml-ontology#ProfessorClass>

## VII. RELATED WORK AND DISCUSSION

Since UML Class Diagram can also be used to represent ontology of a domain, most work related to UML and OWL emphasizes the transformation between UML Class Diagram and OWL ontology [3][14]. Furthermore, OMG’s ODM(Ontology Definition Metamodel) includes UML profiles for RDF and OWL, which provide a standard graphical notation for RDF vocabulary and OWL ontology development using UML tools. Our work is inspired by the work of Van Der Straeten [9] and Wei [12][13][11]. Van Der Straeten and colleagues used Description Logic to check the inconsistency between different versions of UML models. Wei and colleagues specified overlaps of heterogeneous UML models in Conceptual Graphs and used the overlap for identifying missing requirements. However, few work aims at providing a comprehensive and complete UML Ontology that covers all UML Specification diagrams. Furthermore, our work takes advantage of SPARQL to query and check consistency of different UML models.

One important contribution of this work is the construction of a comprehensive UML ontology in the RDF/XML syntax which can be queried and reasoned in Protégé and other Semantic Web tools like Apache Jena. By organizing the latest UML Specification metaclasses in OWL, we invite the community to contribute to its further refinement. For research purposes, we restrict ourselves to a subset of the UML metamodel. The current UML ontology includes 127 most commonly used UML metaclasses and 25 metarelations, which can support defining the semantics of UML Class, Sequence and State Diagrams in OWL notation. Future work

will include all the metaclasses and metarelations defined in the UML Specification, so that more UML diagrams can be converted to OWL for analysis. Another current limitation is the manual conversion from a given UML diagram to OWL. Future work will focus on automation of transforming UML models to OWL and direct invocation of the OWL API from within a UML CASE tool. Another future work would be designing a user friendly querying interface based on natural language query, so that users don’t need to master SPARQL language.

## VIII. CONCLUSION

In this paper, an approach to manage UML model knowledge is proposed and validated through a simple illustrative example software system. The formalism used is W3C standardized OWL, which is a Description Logic based formalism. Three types of UML diagrams have been converted to OWL knowledge base. Queries and reasoning can be conducted towards the resulting knowledge base. Future work will include more OWL reasoning in terms of consistency, knowledge acquisition and inference on the UML knowledge base. The results can help requirements engineers better understand their models and provide possible inconsistencies suggestions.

## REFERENCES

- [1] S. W. Ambler. *The object primer: Agile model-driven development with UML 2.0*. Cambridge University Press, 2004.
- [2] D. Dermeval, J. Vilela, I. I. Bittencourt, J. Castro, S. Isotani, P. Brito, and A. Silva. Applications of ontologies in requirements engineering: a systematic review of the literature. *Requirements Engineering*, 21(4):405–437, 2016.
- [3] D. Djurić, D. Gašević, and V. Devedžić. Ontology modeling and mda. *Journal of Object technology*, 4(1):109–128.
- [4] D. Firesmith. Are your requirements complete? *Journal of Object Technology*, 4(1):27–44, 2005.
- [5] F. J. Lucas, F. Molina, and A. Toval. A systematic review of uml model consistency management. *Information and Software Technology*, 51(12):1631–1645, 2009.
- [6] O. M. G. (OMG). Unified modeling language, version 2.5.1. OMG Document Number formal/17-12-05 (<https://www.omg.org/spec/UML/2.5.1>), 2017.
- [7] L.-j. SHAN and H. ZHU. A formal descriptive semantics of uml. *Computer Engineering & Science*, 3:026, 2010.
- [8] S. U. Stanford Center for Biomedical Informatics Research (BMIR). Protégé, a free, open-source ontology editor and framework for building intelligent systems. <https://protege.stanford.edu/>, 2018.
- [9] R. Van Der Straeten, T. Mens, J. Simmonds, and V. Jonckers. Using description logic to maintain consistency between uml models. In *«UML» 2003-The Unified Modeling Language. Modeling Languages and Applications*, pages 326–340. Springer, 2003.
- [10] B. Wei. *A comparison of two frameworks for multiple-viewed software requirements acquisition*. PhD thesis, Ph. D. thesis, University of Alabama in Huntsville, 2015.
- [11] B. Wei and H. S. Delugach. A framework for requirements knowledge acquisition using uml and conceptual graphs. In *Software Engineering Research, Management and Applications*, pages 49–63. Springer, 2016.
- [12] B. Wei and H. S. Delugach. Transforming uml models to and from conceptual graphs to identify missing requirements. In *International Conference on Conceptual Structures*, pages 72–79. Springer, 2016.
- [13] B. Wei, H. S. Delugach, E. Colmenares, and C. Stringfellow. A conceptual graphs framework for teaching uml model-based requirements acquisition. In *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET)*, pages 71–75. IEEE, 2016.
- [14] J. Zedlitz, J. Jörke, and N. Luttenberger. From uml to owl 2. In *Knowledge Technology*, pages 154–163. Springer, 2012.

# An Ontology-Based Modelling of Vietnamese Traditional Dances

Truong-Thanh Ma<sup>1</sup>, Salem Benferhat<sup>2</sup>, Zied Bouraoui<sup>2</sup>, Karim Tabia<sup>2</sup>, Thanh-Nghi Do<sup>1</sup>, Huu-Hoa Nguyen<sup>1</sup>

<sup>1</sup> CICT, Can Tho University, Vietnam  
Email: truongthanh1511@gmail.com  
{dtngchi, nhhoa}@cit.ctu.edu.vn

<sup>2</sup> CRIL, Artois University  
CRIL CNRS & Univ Artois, France  
Email:{benferhat, bouraoui, tabia}@cril.fr

**Abstract**—Ontology is an essential resource to enhance the performance of information processing system as well as is an intelligent storage area served for management of large-scale heterogeneous digital contents resulting. In this paper, we propose the initial steps for reconstructing a significant schema of Vietnamese traditional dances. Most of the typical dances of Vietnamese community are recorded in multimedia format, in raw videos. Accordingly, we concentrated on analyzing and collecting knowledge of the dance experts at art schools in Vietnam to classify and to determine the primary features that would be stored in the ontology. We propose an ontology-based modelling for the cultural heritage domain of Vietnamese traditional dance.

**Keywords:** *Vietnamese Traditional Dance, Ontology, Knowledge Representation, Modelling.*

## I. INTRODUCTION

Southeast Asia is one of the most assertive growing regions in the world with natural and cultural resources. Specially, the cultural foundation regarding dance domain plays an important role in community life, it always brings the historical and cultural knowledge to the adjacent generation [1] [11]. The intangible culture heritages (ICHs) of the ethnic group dance are quite difficult to identify the exact values and great significance. This paper focuses on Vietnamese traditional dances (VTDs) which are truthfully assertive and original.

VTDs principally concentrate on four primary aspects: [4][?]: firstly, depicting straightforward the scene of animated daily activities as well as redrawing a picture about traditional careers, particular symbols and outstanding stories in history through truthful dance movements; Secondly, utilizing conventional and ordinary props combined with gentle music. The selected props for performing depend on the song lyric contents in order to express characteristic emotions. The third point is the dance must approach so-called "the true, the good and the beautiful". Vietnamese typical stories and scenes always reflect comprehensively and evidently the daily life of each ethnic group through distinctive dances. The final aspect is the dance message. Most of the VTDs are not only creating aestheticism on the stage as well as serve the entertainment but also transshipping many meaningful messages to audiences.

One of the main contribution of this paper is to propose an ontology-based modelling of VTD. Our ontology is modeled using from the books [1] [4] [10] [11] [2] and also from

the dance experts at the art schools in Vietnam. Ontology is a principle of any system to represent knowledge for a given domain [12]. It represents information from multiple heterogeneous sources in concepts and semantic relations of the concepts. Ontology plays an important role in the representation of information processing system and also is one of the formalism of describing entities, the properties and relationship. It also offers to managing and sharing information. Especially, it brings a mission of transshipping information from multimedia data (such as raw video/images) to computer. Correspondingly, utilizing ontology for preserving ICH is a completely appropriate selection because the traditional dances is recorded in raw videos which are complicated and enigmatic for storing heterogeneous information blocks. The combination between artificial intelligence and expert knowledge regarding dance domain for preserving ICH is definitely expected.

Maintaining the intangible cultural properties as well as the traditional dance is extremely difficult. Indeed, the dance documents in Vietnam is extremely limited. Most of the minority ethnic group dances are not written in any book or any document. They retransmitted through "word of mouth" form, particularly, the VTDs of minority ethnic groups.

Our VTD approach is decomposed in three primary stages: the first aspect is panorama overview of dances (1); the second angle is region-zone of dances (2), considering this aspect because most of the VTDs depart from distinguishable ethnic groups living region-zone in Vietnam territory; the last approach is the fundamental movements of ethnic groups (3). In this paper, we concentrate on presenting the first stage to build VTD ontology. Our primary challenge is to determine primary concepts combined with essential properties based on expert knowledge and restructuring the ambiguity in VTDs.

The remainder of this paper is structured as follows. In the next section we give an overview and related works. In section 3, we present expert knowledge to develop ontology for VTDs. In section 4, we discuss and illustrate building an ontology. Finally, section 5 concludes the paper.

## II. OVERVIEW AND RELATED WORKS

### A. Vietnamese Traditional Dance Overview

Vietnam has a rich cultural heritage, where music and dance has been interwoven with the social fabric. Vietnamese classi-

cal music and dance portray human emotions, love and devotion, narrate stories from life activities, history and religious and are the integral parts of the festivals and celebrations. List of ICHs compiled by UNESCO includes many dances, instruments, theater and music forms, namely "Đờn ca tài tử" of the Vietnam's South, "Nhã nhạc cung đình" of Huế, "Ca trù" of the Vietnam's North, "Cồng chiêng Tây Nguyên" of Western Highlands of Vietnam and so on. Although the VTDs of ethnic groups have not been written into the UNESCO list yet, nevertheless it brings many local traditional cultural values in each particular dance and acknowledged the ICHs in the heart of Vietnamese community.

There are three basic characteristics that the choreographer as well as performer must be paid special attention to: (1) ethnicity, (2) transshipping meaningful message, and (3) "chân-thiện-mỹ" (*the true, the good and the beautiful*). The explicit explanation as follows: firstly, regarding the ethnicity, most dances of the minority ethnic groups is to predominantly concentrate on the ethnicity, in spite of performing with simple movements, they always bring private ethnic features for each performance, it can be local particular props or ethnic costumes. Secondly, considering dance messages, the VTDs is kept in a significant mission to be transshipped at least one message to receivers who is not only spectator but also anything and even be the god. Hence, the VTDs is a steady bridge in educating about human dignity, morality and even historical knowledge. Instead of learning the historical lesson in regular classes as well as participating in the training course for life skills, the dances has become the digital channel for efficient educating personality, knowledge and even ethnicity to the generations. Thirdly, the explanation as follows "chân (*the true*)" means the performer must express their emotions truthfully inside each movements and face's expression; "thiện (*the good*)" means the dances approach the best of the activities in life, it brings an aspiration for the bright future of their ethnic group; "mỹ (*the beautiful*)" means the dances and dancers must be interested in aestheticism.

In addition, Vietnam is a multi-ethnic country bringing many different cultures [11], fifty-four-ethnic groups are living in a territory with many typical multiform dances. The most remarkable feature to easily distinguish the dances of each ethnic group is the ethnic props as well as ethnic traditional costumes. Normally, the ethnic people would wear the private costumes and typical props of their ethnic group for performing in the particular festivals or traditional celebrations. Props as bamboo, hat, towel, soft silk and even daily tools for working as shovel, "lưỡi hái" and baskets are utilized in VTD. Besides we are also seen some ethnic props as "Đàn tính", "khèn" (music instrument), or "Khăn piêu" (handkerchief of Thai people). Almost all ethnic groups possess the private specific costumes of their ethnic group, the selection of the traditional costumes is always the first precedence. Additionally, we are also effortless to catch sight of pastoral pictures through common props as banana tree, bamboo tree, stack of straw for Vietnamese countryside. Generally, most of the VTDs concentrates on rural area than urban region,

therefore the scenes and props are utilized to intimately related to countryside or village.

### B. Related Works

The author in [6] illustrates preserving and promoting of intangible cultural through Benesh notation to define the movements, after that using OWL (Ontology web language) and the semantics of Benesh Movement notation (BMN) to build Ontology for video movement. In [5], dances were described and stored taking advantage of expressivity of description logics. Dance choreographies was built in OWL to represent and archive. Besides using SPARQL queries for searching within the ontology based on steps and movements of dances. Another approach for automatic annotation and retrieval of video content, which is based on ontologies, has been presented by Ballan et al [9]. They build an ontology schema based on abstract concepts and relations, after that they described a web video search engine that based on ontologies. In [7], the author group presented a method to encode and structure metadata of folk song collections. This method uses and extends the CIDOC (Conceptual Reference Model) CRM, the research result is to determine the roles and classes provided by CIDOC CRM have been useful in dealing with the particularities of Basque folk song collections.

## III. VTD EXPERT KNOWLEDGE

### A. Expert Knowledge Congregation

VTD is the art form reflecting the human life through its formal form, most of the gestures and movements intimately related to processing manual labor as well as the life activities in ethnic community. Each dance performance session lasts from 4 minutes to 10 minutes, including average about 5-10 dancers (normally) [11]. In VTD, it splits two types: "múa dân gian" (*folk dance or community dance*) and "múa cung đình" (*court dance or dance in theatre*). They are known as "Múa dân tộc", we could call traditional dance in English. In this section, we concentrate primarily to explicitly model the knowledge on VTD, which could be split in five categories: non-story dances (dư-hứng), bare-handed traditional dances of Vietnam, historical dances, dances about traditional manual labor of ethnic groups, dances in festival and daily life.

The following details each of these categories:

1) *Non-story dances - (dư-hứng)*: In Vietnam, non-story dance is one of the most prevalent dance forms that the ethnic group utilizes to perform in life as well as in the significant events. "Dư Hứng dance" is the common name of this dance form which is called by experts of the dance domain. These dances are the performances without story. The remarkable feature of these dances is the ethnicity and community, because the dance performance brings a private ethnic color in each particular movement. The performers could play creatively in anywhere and any kind of music. Non-story dances are divided in two main kinds as in sub-schema of Figure 1 do that: utilizing props for performing and dances without props (bare-handed dance).



In this subsection, we concentrate on analyzing the dances utilizing props without plot. Normally, the props intimately related to life activities and typical ethnic group symbols. Considering about props of life, there are four main props: (1) flowers, (2) manual labor tools, (3) instruments and (4) symbols. Firstly, most of the flowers would be one of the first selection in VTDs, there are many kinds of flower for performing as lotus dance, sunflower dance, rose dance and so on. In which, lotus and sunflower is performed the most because they had been acknowledged as national flower by Vietnamese community. Secondly, the manual labor tools is also performed in dances, it is the tools of the life activities, for example: "pick tea leaves" dance of Thai ethnic group with familiar prop "gùi - similar with papoose" or "Lên nương dance" with hoe tool and leaf-hat. Thirdly, the ethnic instruments are also the dance name and they would be primary props to perform in those dances, namely "Đàn tính" dance (a kind of instrument of Thái ethnic group), "Trống Cơm" dance (a kind of drum), "gong" dance (percussion instrument) and so on. Finally, the local and national symbols also performed in the major festival of ethnic groups, including lotus dance, "trống đồng" dance.

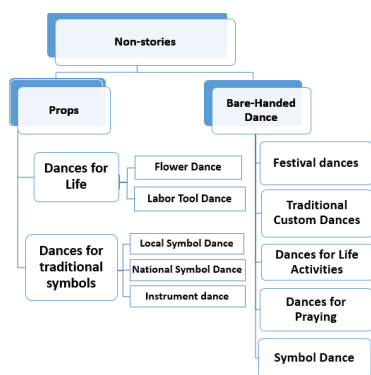


Figure 1. Sub-schema of non-story dances

2) *Bare-handed dances*: This is the dances that performers just utilize the hands and body language without any prop in each dance performance. Creating the distinct hand gestures combine the fundamental movements is one of the private features of these dances. Normally, the dance performances are organized from two to eight people for a dance formation, sometimes even there are the large number of dancers participated in this dance group. There are five main subjects in these dances: (1) symbol dances without props, (2) traditional custom dances, (3) dances for life activities, (4) festival dances, (5) dances for praying. The following details each of the five subjects in bare-handed dances: Firstly, this is dance that performer using body-language to depict ethnic symbols. Instead of utilizing props for performing, these dance performances would be imaginableness through postures and movements, evidence as lotus dance, peafowl dance (note that do not must forms of shadow dance). Secondly, the dances about traditional custom is performed in remarkable events of ethnic groups or anniversaries, such as "Xòe-thương-nhau"

dance, HMông dance. Thirdly, the dances for life activities is performed at parties (wedding, birthday) in ethnic community (note that it is not as formal as traditional custom dances), example as Lăm-leo dance, dù-kê dance of Khmer ethnic group. Fourthly, the remarkable feature of festival dances is heightened the ethnicity in each dance performances. Normally, the circle formation in festival with simple movements to create funny atmosphere is always selected to perform, such as rom-vong dance, saravan dance and so on. Finally, let us consider [10] an example regarding "Tắc-xình" dance of "Sán-Chay" ethnic group is an evidence for praying in dances. This dance is in "cầu mùa" (pray for harvest) festival, it is not only bringing wishes of the good harvest but also an aspiration for connecting the communities together.

3) *Historical dances*: The historical dance performance depicted the scenes of the outstanding events of Vietnamese history as well as the legends. We could split historical dances on three primary aspects: Firstly, the scene of remarkable war events (1), such as "The great victory on the spring in 1975", "The Dien Bien Phu victory"; Secondly, the scene of an illustrious national hero (2) had batted unyieldingly in antitrespassing wars as well as the prominent background of national hero, such as sister "Võ Thị Sáu", Uncle Hồ, Vương king; and the third aspect (3) is the legends of Vietnam, as "Lac Long Quan" king an AuCo Queen, "Thánh gióng".

Generally, most of the dance performances primarily concentrates on the historical message transshipping, the noticeable point is the selected props for performance to be inherent in the historical figure, example: dragon boat and magic sword associate with "Lê Lợi" King; Rằn handkerchief and Áo-bà-ba (Skirt) associate with "Võ Thị Sáu". The remarkable feature of these dances is the personality description as well as transshipping the content of story. The choreographers must select the cardinal features of story to build dance lesson, through the meaningful stories will be worth lessons to recommend and to educate the present generation regarding moral person, history and ethnicity.

4) *Dances regarding traditional manual labor*: The dance performances regarding traditional labor is necessary with supporting of props and bringing a plot. The dances depict truthfully the scenes of traditional manual works. It could be the steps as well as the stages on the process of traditional manual labor in their ethnic group. Normally, the choreographer extracts the scene from the traditional works to build the dance performance, they would retrace step-by-step of the scenes in their traditional career to aim at honoring the labor values. For examples: the salt village dance - they would perform their traditional work with four segments (getting water from sea, containing water, drying-water, crystallizing).

5) *Dances in festival and daily life*: In Vietnam, the dances regarding daily life is split in two main kinds: the dances in rural (1) and the dances in urban (2). Dances for rural region depicts the scene of countryside, works on the fields and even the scenes from daily life activities with pasturing buffaloes, fishing and cultivation. In contract, considering



Figure 2. The sub-schema of festival dances and daily life dances

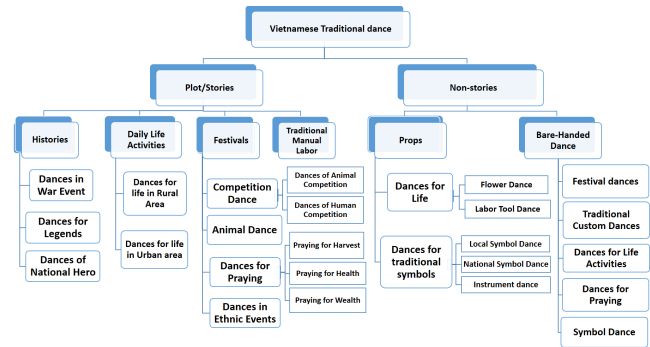


Figure 3. The overview schema for VTD

dances for urban areas is the scenes of the modern life to reflect the bad activities from the young generation. It extracts the dark background in the city to admonish their children (the adjacent generations). More importantly, almost all the dances regarding life of urban area is belonged to Kinh ethnic group.

On the other hand, considering the dances in festival, it splitted on four particular aspects as in Figure 2 the first aspect is animal dance (1). The second one is competition dances between human and animal (2). Thirdly is the dances in significant event (3); and last aspect is the dances for praying (4). All dances in this section always bring a plot in each dance performance. Firstly, considering animal dances, normally, this animal is treasured in the ethnic community, such as: "tortoise dance" of Dao ethnic group. These animal dances would like to receive lucky, happiness and prosperity from sacred animal. Secondly, regarding competition dance, this is the dance performance relevant to competition between animals together or between animal and human. Generally, these dances focus on taming the bad character to return with good lifestyle. Thirdly, the dances in event, it is the special traditional dances to welcome visitor in significant events. These events could be the annual events, the new outstanding events and ethnic community events without bringing the historical elements. In last aspect, regarding praying in dance [10], these dances praying about harvesting the good crops in the whole year or praying for sufficient money, food and happiness or praying a good health for all members in their family and village.

### B. Structure of Vietnamese Traditional Dances

This subsection provides a brief overview schema for VTD. We recall that this schema is based on [1] [4] [10] [11] but also benefits from the dance experts (*consist of dance choreographers, dancers and their teacher*) at art school in Vietnam, we have selected the significant features to reconstruct the schema for VTD regarding case of the panorama overview as Figure 3. There are three main features that the dance experts have suggested to be interested in researching: Firstly, ethnicity in dances to answer for question about "ethnic groups" because most of the traditional dances depart from the customs of ethnic groups, interested in props and costumes. Secondly,

transshipping the meaningful messages to spectator, interested in plot. Thirdly, festivals or events of ethnic groups.

### IV. CONCLUSION AND FUTURE WORKS

The preservation and promotion of ICH is one of the most interest and worried problems in Vietnamese community, particularly, in case of traditional dances. For preservative purpose, we collected significant features needed to build an ontology for VTD. The work presented in this paper is the first stage regarding preserving and promotion of VTDs based upon the background of artificial intelligent. Our next plans include enhancement and extension of the VTD ontology following different aspects and concentrate on developing applications automatically based on ontology.

### V. ACKNOWLEDGEMENTS

This work has received support from the European Project H2020 Marie Sklodowska-Curie Actions (MSCA), Research and Innovation Staff Exchange (RISE): Aniage project (High Dimensional Heterogeneous Data based Animation Techniques for Southeast Asian ICH Digital Content), No: 691215.

### REFERENCES

- [1] L.T.Loc, "Mua dan gian cac dan toc Viet Nam", in *Thoi-dai Publishing house*, 1994.
- [2] T.V. Son, Đ. T. Hoàn, N. T. M. Hương, "Mua dan gian mot so dan toc vung Tay Bac", in *Culture and Nation Publishing House*, 2003.
- [3] J. Davies, A. Duke, Y. Sure, "OntoShare - An Ontology-based Knowledge Sharing System for virtual Communities of Practice", in *proceedings of the 28th International Florida Artificial Intelligence Research Society Conference (FLAIRS)*, 2005.
- [4] L.N. Canh, "Nghệ thuật múa Hà Nội truyền thống và hiện đại", in *Hà Nội Publishing House*, 2011.
- [5] E.R. Katerina, I. Yannis, "A labanotation based ontology for representing dance movement", in *Proceedings of the 9th Int Gesture Workshop*, 2011.
- [6] S. Sawzan, D.B. Dominique, M. Said, M.Pierre, "An ontology for video human movement representation based on Benesh notation", in *Proceedings of Multimedia Computing and Systems (ICMCS)*, 2012.
- [7] S. Debastian, "Ontologies for representation of folk song metadata", in *Technical report, EHKZAARE201201*, 2012.
- [8] S.Renzo and A. Hernan, "Ontology and semantic wiki for an ICH inventory", in *Proceedings of Computing Conference (CLEI)*, 2013.
- [9] L. Ballan, A. Del Bimbo and G. Serra, "Learning Ontology Rules for Semantic Video Annotation", in *Proceedings of the 2nd ACM workshop on Multi semantics*, 2008
- [10] L.N.Canh. "Múa tín ngưỡng dân gian Việt Nam", in *Social Science Publishing house*, 1998.
- [11] L.N.Canh. "Đại cương nghệ thuật múa", in *Culture and information publishing house*, 2003.

# Influence Factors in Software Productivity

## A Tertiary Literature Review

Edson Oliveira, Tayana Conte, Marco Cristo  
Institute of Computing  
Federal University of Amazonas, UFAM  
Manaus, Brazil  
{edson.cesar,tayana,marco.cristo}@icomp.ufam.edu.br

Natasha Malveira Costa Valentim  
Department of Informatics  
Federal University of Paraná, UFPR  
Curitiba, Brazil  
natasha@inf.ufpr.br

**Abstract** — Software organizations need to increase their productivity to stay competitive. Although there is a lot of research on productivity in software development, software organizations still do not know what are the most significant productivity factors in which they should invest. This paper presents a Tertiary Literature Review (TLR) that aimed to identify and analyze Systematic Literature Reviews (SLR) on the influence factors of software productivity reported in the scientific literature. We extracted and classified the influence factors into organizational factors (organizational dependent factors) and human factors (people dependent factors). Using this information, software organizations can improve the productivity of their projects by evaluating the influence factors that best fit their context.

**Keywords** – Tertiary Literature Review; Productivity Influence Factors; Software Productivity.

### I. INTRODUCTION

The competitive environment in software market today requires organizations to increase their quality level and reduce their production costs. The best way to reduce costs in software development is by increasing productivity [1]. According to Aquino and Meira [1], to reduce production costs by improving productivity, the organization needs to select and implement effective practices towards better productivity. These practices, in turn, should be based on the most relevant productivity factors for improving the organization's productivity [2].

Organizations' managers are more aware of the importance of factors that influence the productivity of the team involved in software projects [3]. The problem is that software productivity is influenced by many factors and organizations often do not know what these factors are and neither where to start [4]. Moreover, the impact of these factors on software productivity may be different according to the context and characteristics of the team, the developer, the project and the entire organization [4].

According to Hernández-López et al. [5], many of the factors that influence software productivity are known and used in estimation models. However, it is not clear whether the importance of the identified factors has changed over time, given that the processes and tools have evolved considerably since the initial studies [5]. Another problem is that there are

many factors that influence productivity so that taking all of them into consideration in an analysis would not be economically viable [4]. Therefore, it is best to focus on a limited number of factors that have a greater impact on the productivity of organizations.

This has motivated us to review the factors identified in the literature. As there is already extensive research in the area, with the existence of some Systematic Reviews on the subject, we decided to conduct a Tertiary Literature Review (TLR) to identify and classify the influence factors of software productivity reported in the scientific literature. TLRs are Systematic Literature Reviews (SLR) of secondary studies, which are also SLRs [6]. This paper describes the tertiary review we carried out, presenting the results obtained, classifying the factors of influence found in human and organizational factors.

Section 2 reports our background. Section 3 reports the planning and execution of our TLR. In Section 4, we present the data extracted from each SLR and answer our research questions. We report some discussions of our TLR in Section 5 and our conclusions and future work in Section 6.

### II. BACKGROUND

The first scientific researches involving the concept of productivity in Software Engineering were published around the beginning of the 80's [7, 8]. The subjects of these studies involved the measurement of productivity and the search for factors that influence productivity in software projects. During the 90's, there was a significant increase in the amount of research on software productivity. Research on productivity measurement continued, as well as the study of several factors influencing productivity. These studies include, for instance, factors influencing the productivity of software maintenance [9] and the influence of software reuse on productivity [10].

From the year 2000, research involving productivity explored more influence factors on productivity, such as factors related to new methods, programming in pairs [11], and software development techniques and refactoring in agile teams [12]. Other research investigated productivity factors in different contexts, such as in Enterprise Resource Planning (ERP) [13], in open-source project development, and with teams working continuously in different time zones [14].

In the literature, there are some secondary studies on the factors studied here [4][15][16]. However, we did not identify TLRs that add knowledge of secondary studies. This scenario motivated us to carry out a tertiary study in order to capture the current results of SLRs in software productivity.

### III. TERTIARY LITERATURE REVIEW

In this section we describe in detail the protocol used to conduct this tertiary literature review.

#### A. Goal and research questions

The Evidence-Based Software Engineering aims to apply an evidence-based approach to both research and practice in Software Engineering [17]. Evidence means the synthesis of scientific studies related to a research theme or question. The most reliable evidence comes from aggregating all empirical studies on a particular topic [6]. The recommended method for aggregating evidence is the Systematic Literature Review (SLR), which is characterized as a secondary study. SLR aims to establish a formal process for conducting a literature review, avoiding the introduction of eventual biases. SLRs allow the identification, evaluation, and interpretation of all available and relevant research regarding a research question [18].

In order to evaluate the current state of the research based on software productivity evidence, we conducted a tertiary study. This study is a systematic review of secondary studies and uses the same methodology of a SLR [18].

The main research question of this TLR is "*What are the productivity factors found by existing secondary studies on productivity factors in software development?*" This main question serves as the basis for the following sub-questions:

**SQ1** – *What was the classification used to organize the influence factors we have found?*

**SQ2** – *What were the influence factors found by the secondary studies?*

The *SQ1* sub-question aims to identify the classification employed in the secondary studies to organize the factors influencing productivity. The influence factors on productivity are the focus of sub-question *SQ2*.

#### B. Search strategy

The search strategy of this TLR included the items listed below.

**Search sources:** the digital libraries ACM Digital Library, Engineering Village, IEEE Xplore Digital Library, Scopus and Web of Science. These libraries were chosen due to the experience reported by Dybå et al. [19].

**Document type:** for this tertiary review, we considered only literature reviews published in scientific venues, such as conference proceedings and journals, since these publications have their content reviewed by other independent researchers (peer review).

**Search language:** only papers in English, due to its adoption by most of international Software Engineering conferences and journals.

#### C. Search string

The search string was based on terms selected from a reference list composed by four secondary studies on productivity factors identified in an earlier exploratory literature review [4][20][15][16] and also on terms used in a tertiary review carried out by Kitchenham et al. [6]. We classified these terms into three groups: (i) terms associated with software development, (ii) terms associated with productivity factors of software development, and (iii) terms associated with secondary studies. The first group relates to the context of this tertiary review, based on the words described in the title and abstract of the reference list used. The second bases on the search strings used by the studies of the reference list. Finally, the third relates to the search for secondary studies from the tertiary review by Kitchenham et al. [6]. The search string was defined as shown in TABLE 1.

TABLE 1. SEARCH STRING

Group	Search String
<b>Software Development</b>	("software development" OR "software engineering") AND
<b>Productivity Factors</b>	("factor" OR "indicator" OR "driver") AND ("productivity" OR "development efficiency" OR "development effectiveness" OR "development performance") AND
<b>Secondary Studies</b>	("review" OR "overview" OR "literature" OR "meta-analysis" OR "past studies" OR "in-depth survey" OR "subject matter expert" OR "analysis of research" OR "empirical body of knowledge" OR "overview of existing research" OR "body of published research")

We carried out the analysis of the data extracted in this tertiary review by using the content analysis technique, which is used to categorize and determine the frequency of these categories, facilitating the analysis of the evidence [21].

#### D. Criteria for studies selection

The inclusion criterion (IC) for the 1st filter is: "the publication describes a literature review on productivity factors in software development". The exclusion criterion (EC) is a negation of the inclusion criteria. We used the criteria of the 1st filter to select the publications by reading the title and the abstract. The criteria adopted in the 2nd filter are presented in TABLE 2.

TABLE 2. INCLUSION (IC) AND EXCLUSION (EC) CRITERIA FOR THE 2<sup>ND</sup> FILTER

Criteria	Description
<b>IC.1</b>	The publication is a literature systematic review, with a defined search process on productivity factors in software development.
<b>EC.1</b>	The publication is not a secondary study or does not have a defined search process.
<b>EC.2</b>	The publication is not a secondary study on productivity factors in software development.
<b>EC.3</b>	The publication does not have a list of extracted productivity factors.
<b>EC.4</b>	The publication is not a scientific paper, e.g., it is a chapter of a book. Thus, we are not sure that it was reviewed by another researcher (peer review).
<b>EC.5</b>	The publication is not in English or is not available.

In the 2nd filter, we applied the inclusion and exclusion criteria based on the complete reading of the selected publications after the 1st filter. As an example, criterion EC.1 excludes publications that did not present the description of a defined search process, as they are not characterized as a systematic literature review, following the same criterion also adopted by Kitchenham et al. [6].

*E. Data extraction strategy*

We extracted the data from the selected publications for analysis and interpretation in order to answer each of the research sub-questions. We classified the data extracted in this tertiary review as *productivity factor classification data* and *productivity factor specific data*.

Productivity factor classification data, which includes the category names and descriptions in the taxonomy used to classify the factors, is important to answer the SQ1 sub-question. This data may indicate if there is a common classification adopted by the researchers. The productivity factors specific data, their names and descriptions, are important in order to respond the SQ2 sub-question.

*F. Studies selected after performing the tertiary review*

The first two authors of this work carried out the search and selection strategies defined in this tertiary review, while the others reviewed all the work. We found 353 publications after searching in the selected digital libraries. After removing duplicates, 240 publications were selected for filtering. Among them, 221 were excluded because they did not meet the inclusion criteria for the first filter. We read the remaining 19 publications thoroughly and, at the end of the selection, only 4 publications met the criteria of the second filter (TABLE 3).

To assess the reliability of applying the defined criteria [27], the two researchers applied the selection criteria independently in a random sample of 30 publications using the Kappa statistical test [22] to assess agreement. The result of this agreement evaluation, using data from the first filter, was significant ( $\kappa = 0.783$ ), according to the suggestion proposed by Landis and Koch [23] for interpreting this value.

At the end of the process, we selected 4 publications describing secondary studies on factors influencing productivity, containing a total of 139 different factors. Each publication presented a classification of factors found in different groups. The period of these publications is recent (2008 to 2015) and reflects the increasing interest of researchers in factors influencing productivity. In this paper, we will refer to the reviews carried out by Wagner and Ruhe [16], Trendowicz and Münch [4], Paiva et al. [15] and Dutra et al. [20] as R1, R2, R3, and R4, respectively.

IV. RESULTS

In this section, we present the results obtained for the two research questions proposed for this study. For the first question, we present and analyze the classification adopted by each secondary study selected. For the second research question, we present and classify the factors extracted from each of the secondary studies selected.

*A. SQ1. What was the classification used to organize the factors we have found?*

To facilitate analysis, we grouped together the factors, extracted from primary studies, with similar meaning. In two reviews, the resulting category groups were grouped again, yielding a hierarchical classification of influence factors. Only the review by Paiva et al. [15] (R3) did not adopt any productivity factor classification. Among the adopted classifications, the R1 review defined and used its own classification, while the other two (R2 and R4) based their classification on other studies. As can be seen in TABLE 3, no common classification of factors of productivity influence can be found.

Wagner and Ruhe [16] classified the factors found in two major groups: (i) technical factors, which are factors related to the product, the process, and the development environment; and (ii) non-technical factors, i.e., factors present intangibly in the development team and in the work environment. The authors also considered these non-technical factors as human factors.

TABLE 3. CLASSIFICATIONS OF FACTORS IN THE LITERATURE REVIEWS

Review	Classification
<b>R1</b> Wagner and Ruhe [16]	Technical Factors: <ul style="list-style-type: none"> <li>▪ Product Factors</li> <li>▪ Process Factors</li> <li>▪ Development Environment Factors</li> </ul> Non-technical Factors: <ul style="list-style-type: none"> <li>▪ Project Factors</li> <li>▪ Organizational Culture Factors</li> <li>▪ Team Culture Factors</li> <li>▪ Capability and Experience Factors</li> <li>▪ Work Environment Factors</li> </ul>
<b>R2</b> Trendowicz and Münch [4]	Influence Factors: <ul style="list-style-type: none"> <li>▪ Product Factors</li> <li>▪ Process Factors</li> <li>▪ Project Factors</li> <li>▪ Personnel Factors</li> </ul> Context Factors
<b>R3</b> Paiva et al. [15]	<i>No classification defined</i>
<b>R4</b> Dutra et al. [20]	Team Emergent States Factors Individual Characteristics Factors Support Tasks Factors

Trendowicz and Münch [4] (R2 review) first divided the extracted factors into context and influence factors. According to the authors, given a productivity model, the factors considered by the model are the influence factors; while the factors absent from the model are the context factors, i.e., the ones present in the context and considered constant for the defined model. The authors further subdivided the influence factors into four groups: product, personnel, project and process factors. This classification was based on the ones by Fenton and Pfleeger [24], Jones [25] and Ruhe et al. [26].

Dutra et al. [20] (R4 review) categorized the extracted factors according to the unit of analysis indicated in the primary study. That resulted in three groups: (i) team emergent states factors, (ii) individual characteristics factors and (iii) support tasks factors. This classification was based on the work

by Marks et al. [27], which studied the factors that influence software development in high-performance teams.

Finally, we answer SQ1 research question by noting that there is no single common classification, but there are similarities among the adopted taxonomies. Factors related to the product, the process, the project and the people (team) were common categories found in the classifications.

*B. SQ2. What are the influence factors on productivity found by the secondary studies?*

The description of the factors in primary studies is often incomplete and limited only to the name of the factor, according to Trendowicz and Münch [4] (revision R2). Thus, for an analysis of the factors found in literature, the authors of the selected publications used integration strategies to group factors with the same meaning or names. After that, the factors were grouped according to the hierarchical classification adopted in each systematic review.

Wagner and Ruhe [16] extracted 51 factors, integrating them through the use of similar terms. Trendowicz and Münch [4] extracted 246 factors, integrating them according to the name and description used by the primary studies. Paiva et al. [15] performed the extraction of 32 factors, but without explaining the process used. Dutra et al. [20] obtained 15 factors and integrated them according to their semantic similarity. As in the selected reviews, it was also necessary to integrate the factors extracted from these secondary studies. The strategy used was to integrate factors with a similar name and/or description. Then, factors not integrated with any other were grouped under the generic name "other characteristics". The results obtained after this process are presented in TABLE 4 and TABLE 5.

TABLE 4. HUMAN FACTORS EXTRACTED FROM THE SLRS

Factor	Extracted Factor
Capability and Experience	Experience (R3), Programming language experience (R2), Teamwork capabilities (R2), Project manager experience & skills (R2), Application Experience & Familiarity (R2), Overall personnel experience (R2), Tool experience (R2), Applications Experience (R1), Language and Tool Experience (R1), Manager Application Experience (R1), Platform Experience (R1), Analyst Capability (R1), Manager Capability (R1), Programmer Capability (R1)
Knowledge	Knowledge (R4), Domain of the Application (R3), Task-specific expertise (R2)
Clear Goal	Clear Goals (R1), Goal Setting (R4)
Diversity	Diversity (R4), Developer Temperaments (R1)
Motivation	Motivation (R3), Motivation (R4)
Cohesion and Team Communication	Communication (R4), Cohesion (R4), Communication (R3), Interpersonal Relationship (R3), Team Cohesion/communication (R2), Communication (R1), Team Cohesion (R1)
Other Individual Characteristics	Attitudes (R4), Intelligence (R4), Learning ability (R4), Personality (R4), Emotional Intelligence (R4), Empathy (R4), Leadership Style (R4), Work satisfaction (R4), Commitment (R3)
Other Team Characteristics	Mutual respect (R4), Self-efficacy (R4), Trust (R4), Autonomy (R4), Sense of Eliteness (R1), Team Identity (R1), Fairness (R1)

TABLE 5. ORGANIZATIONAL FACTORS EXTRACTED FROM THE SLRS

Factor	Extracted Factor
Architecture	Architecture (R3), Architecture Complexity (R2), Architecture Risk Resolution (R1)
Complexity	Code complexity (R2), Complexity of interface to other systems (R2), Product Complexity (R1), User Interface (R1), Required Software Reliability (R1)
Consistent Requirements	Consistent Requirements (R3), Requirements Management (R2), Requirements Stability (R1)
Complexity and Database Size	Database Size & Complexity (R2), Database Size (R1)
Decentralized Development	Decentralized development (R2), Physical Separation (R1)
Development Constraints	Development Flexibility (R1), Execution Time Constraints (R1), Main Storage Constraint (R1)
Development Tools	Development Tool (R3), CASE tools (R2), Testing tools (R2), Use of Software Tools (R1)
Development Type	Agile Methodology (R3), Type of Project (R3), Methodology (R3), Development Type (R2), Life cycle model (R2), Domain (R2)
Documentation	Documentation (R3), Documentation match to life-cycle needs (R1)
Knowledge Management	Shared Information (R4), Knowledge Management (R3)
Modernity	Modernity (R3), Technological Gap (R3), Use of Modern Development Practices (R3)
Process Maturity	Maturity Level (R4), Process maturity & stability (R2), Process Maturity (R1)
Programming Language	Programming Language (R3), Programming Language (R2), Programming Language (R1)
Project Management	Managerial Involvement (R4), Project Management (R3)
Project Size	Project Size (R3), Project Duration (R1), Software Size (R1)
Prototyping	Prototyping (R3), Early Prototyping (R1)
Code Reuse	Code Reuse (R3), Quality of reused assets (R2), Reuse level (R2), Developed for Reusability (R1), Reuse (R1)
Schedule Pressure	Schedule pressure (R2), Schedule (R1)
Team Size	Team Size (R4), Team Size (R3), Team Size (R2), Average Team Size (R1)
Telecommunication Facilities	Home Office (R3), Telecommunication Facilities (R1)
Testing	Test (R3), Testing (R2), Effective and Efficient V&V (R1)
Time Fragmentation	E-Factor (R1), Time Fragmentation (R1)
Training	Training (R3), Training level (R2)
Staff Turnover	Turnover (R4), Staff turnover (R2), Turnover (R1)
Work Environment	Work Environment (R4), Workstation (R4), Proper Workplace (R1), Camaraderie (R1)
Other Project Factors	Guard Activities (R4), Work breakdown (R4), Target platform (R2), Reviews & inspections (R2), Team structure (R2), Precedentedness (R1), Completeness of Design (R1), Platform Volatility (R1), Hardware Concurrent Development (R1), Product Quality (R1)
Other Factors of the Organization	Organizational Commitment (R4), Benefits (R3), Internet Access (R3), Physical Location (R3), Salary (R3), Credibility (R1), Respect (R1), Support for Innovation (R1)

As we did not find a common classification (SQ1), we adopted one based on factor similarities, organizing them into human factors (TABLE 4) and organizational factors (TABLE 5). The former is directly controlled by the software

organization (product, process, project, work environment, and development environment). The latter depends on the people involved in the organization's projects (culture, capabilities, and experience).

We now answer the sub-question SQ2 by stating that there are at least 35 influence factors in software productivity, extracted from four secondary studies existing in the scientific literature.

## V. DISCUSSION

In this study, we identified 35 influence factors on productivity from four secondary studies on the same subject in recent years. These factors possibly represent the most significant to be considered by software organizations. We did not find any common classification. However, we found some similarities between the categories. We used these similarities to create the classification we adopted in this work: human and organizational factors.

Wagner and Ruhe [16] (R1) reinforced the importance of the existence of a list of productivity factors to assist software organizations, in which we agree. Having a list of factors, it helps software organizations where to begin their control and analysis of productivity factors in their context. In this way, they find out what factors have the most significant impact on their software projects, what work and what do not work. Trendowicz and Münch [4] (R2) concluded that their biggest result is to observe that the success of the software project still depends on the people involved. Paiva et al. [15] (R3) observed that only experience and consistent requirements were considered as important by both researchers and developers. Dutra et al. [20] (R4) observed that team communication and individual motivation were the most researched factors within the context of high-performance teams.

It is clear, from the conclusions of these secondary studies, the importance of human factors for software development. An evidence of the importance of people in software development is noted in the number of factors related to the capability and experience of individuals in the various roles existing in software projects. Nevertheless, this TLR clearly indicates the large interest in studying organizational factors. There are far more studies on organizational factors than on human factors. This contradiction has already been noted by Meyer et al. [28] and explored by Lenberg et al. [29] in their work advocating for Behavioral Software Engineering (BSE).

For software organizations to improve the productivity of their software projects, they need to intervene in factors that can actually influence productivity in their projects. The classification adopted in this TLR indicates the point at which the intervention should occur: in people or in the organization itself. Intervening in the organization itself, through methods, processes, and tools, is much simpler than intervening with people [30]. This may explain a large number of organizational factors researched. However, it is the people who perform the software development process and, therefore, ignore the human factors may explain the dissatisfaction with some development methodologies: they do not consider real organizations [30]. Therefore, it is important for organizations to balance their productivity improvement actions by considering a

combination of human and organizational factors that are compatible with their organizational context.

## VI. CONCLUSIONS AND FUTURE WORK

This Tertiary Literature Review aimed to identify factors influencing productivity. We identified and extracted 35 factors from four systematic reviews on factors influencing productivity. We did not observe any common classification adopted in these studies. This is due to the fact that the classifications adopted depend a lot on the focus that one wishes to investigate. In this study, we classified the extracted factors in organizational and human factors. We also note that organizational factors were more investigated than human factors.

Every study has threats that may affect the validity of its results [31]. The main threat to the validity of the conclusion of this tertiary review is how general are the observed results, since the search strategy may not have collected some relevant papers. To mitigate this threat, we used five different digital libraries, based on the experience reported by Dybå et al. [19]. Other threat to validity is the classification adopted for the factors we have found. This threat was mitigated by the participation of other researchers who also have reviewed our classification. Another threat to validity is the extra layer of abstraction added to integrate factors. This threat was mitigated by adopting similar integration strategies of the selected studies, reducing the side effects caused by this extra layer. Finally, another threat to the validity of the results is the possibility that the author of this study has introduced his bias during the execution of the review protocol. To mitigate this threat, another more experienced researcher reviewed the process of implementing this systematic review.

The next step of this research is to investigate *in vivo*, in software organizations, what are the influence factors observed in their developers. Comparing the results of this TLR with *in vivo* observations may clarify the importance of influencing factors within the context of productivity within organizations.

## ACKNOWLEDGMENT

We would like to thank the financial support granted by SEFAZ, UFAM, CNPq through processes numbers 423149/2016-4 and 311494/2017-0, and CAPES through process number 175956/2013. Finally, we also thank the researchers of USES group for their support during this study.

## REFERENCES

- [1] G. S. Aquino Junior, and S. R. L. Meira, 2009. Towards effective productivity measurement in software projects. In *Proceedings of the 4th International Conference on Software Engineering Advances (ICSEA 2009)* (Porto, Portugal, 2009). Proc. 4th Int. Conf. Softw. Eng. Adv. (ICSEA 2009). IEEE. 241–249. DOI= <http://dx.doi.org/10.1109/ICSEA.2009.44>.
- [2] S. C. de B. Sampaio, E. A. Barros, G. S. de Aquino Junior, M. J. C. Silva, and S. R. de L. Meira, 2010. A Review of Productivity Factors and Strategies on Software Development. In *Proceedings of the 5th International Conference on Software Engineering Advances (ICSEA 2010)* (Nice, França, 2010). Proc. 5th Int. Conf. Softw. Eng. Adv. (ICSEA 2010). IEEE. 196–204. DOI= <http://dx.doi.org/10.1109/ICSEA.2010.37>.

- [3] C. Melo, D. S. Cruzes, F. Kon, and R. Conradi, 2011. Agile Team Perceptions of Productivity Factors. In *Proceedings of the 2011 Agile Conference* (Salt Lake City, UT, Aug. 2011). Proc. 2011 Agil. Conf. IEEE. 57–66. DOI= <http://dx.doi.org/10.1109/AGILE.2011.35>.
- [4] A. Trendowicz, and J. Münch, 2009. Factors Influencing Software Development Productivity – State-of-the-Art and Industrial Experiences. *Advances in Computers*. 77, (2009), 185–241. DOI= [http://dx.doi.org/10.1016/S0065-2458\(09\)01206-6](http://dx.doi.org/10.1016/S0065-2458(09)01206-6).
- [5] A. Hernández-López, R. Colomo-Palacios, and A. García-Crespo, 2013. Software Engineering Job Productivity – A Systematic Review. *International Journal of Software Engineering and Knowledge Engineering*. 23, 3 (Apr. 2013), 387–406. DOI= <http://dx.doi.org/10.1142/S0218194013500125>.
- [6] B. Kitchenham, R. Pretorius, D. Budgen, O. Pearl Brereton, M. Turner, M. Niazi, and S. Linkman, 2010. Systematic literature reviews in software engineering – A tertiary study. *Information and Software Technology*. 52, 8 (Aug. 2010), 792–805. DOI= <http://dx.doi.org/10.1016/j.infsof.2010.03.006>.
- [7] A. J. Albrecht, 1979. Measuring application development productivity. In *Proceedings of the Joint SHARE, GUIDE, and IBM application development symposium* (Monterey, California, 1979). Proc. Jt. SHARE, Guid. IBM Appl. Dev. Symp. IBM Corporation. 83–92.
- [8] E. Chrysler, 1978. Some basic determinants of computer programming productivity. *Communications of the ACM*. 21, 6 (Jun. 1978), 472–483. DOI= <http://dx.doi.org/10.1145/359511.359523>.
- [9] G. K. Gill, and C. F. Kemerer, 1991. Cyclomatic complexity density and software maintenance productivity. *IEEE Transactions on Software Engineering*. 17, 12 (1991), 1284–1288. DOI= <http://dx.doi.org/10.1109/32.106988>.
- [10] C. Deng-Jyi, and P. J. Lee, 1993. On the study of software reuse using reusable C++ components. *Journal of Systems and Software*. 20, 1 (1993), 19–36. DOI= [http://dx.doi.org/10.1016/0164-1212\(93\)90046-Z](http://dx.doi.org/10.1016/0164-1212(93)90046-Z).
- [11] K. M. Lui, and K. C. C. Chan, 2006. Pair programming productivity: Novice–novice vs. expert–expert. *International Journal of Human-Computer Studies*. 64, 9 (Sep. 2006), 915–925. DOI= <http://dx.doi.org/10.1016/j.ijhcs.2006.04.010>.
- [12] R. Moser, P. Abrahamsson, W. Pedrycz, A. Sillitti, and G. Succi, 2008. A case study on the impact of refactoring on quality and productivity in an agile team. *Lecture Notes in Computer Science*. 5082, (2008), 252–266.
- [13] E. Stensrud, and I. Myrvtveit, 2003. Identifying high performance ERP projects. *IEEE Transactions on Software Engineering*. 29, 5 (May 2003), 398–416. DOI= <http://dx.doi.org/10.1109/TSE.2003.1199070>.
- [14] J. A. Colazo, 2008. Following the sun: Exploring productivity in temporally dispersed teams. *14th Americas Conference on Information Systems, AMCIS 2008*. 3, (2008), 1833–1839.
- [15] E. Paiva, D. Barbosa, R. Lima, and A. Albuquerque, 2010. Factors that Influence the Productivity of Software Developers in a Developer View. In *Innovations in Computing Sciences and Software Engineering*, T. Sobh and K. Elleithy, eds. Springer Netherlands. 99–104. DOI= [http://dx.doi.org/10.1007/978-90-481-9112-3\\_17](http://dx.doi.org/10.1007/978-90-481-9112-3_17).
- [16] S. Wagner, and M. Ruhe, 2008. A Systematic Review of Productivity Factors in Software Development. In *Proceedings of the 2nd International Software Productivity Analysis and Cost Estimation (SPACE 2008)* (Beijing, China, 2008). Proc. 2nd Int. Softw. Product. Anal. Cost Estim. (sp. 2008).
- [17] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, 2009. Systematic literature reviews in software engineering – A systematic literature review. *Information and Software Technology*. 51, 1 (Jan. 2009), 7–15. DOI= <http://dx.doi.org/10.1016/j.infsof.2008.09.009>.
- [18] B. Kitchenham, and S. Charters, 2007. *Guidelines for performing Systematic Literature Reviews in Software Engineering*.
- [19] T. Dybå, T. Dingsøy, and G. K. Hanssen, 2007. Applying systematic reviews to diverse study types: An experience report. *Proceedings - 1st International Symposium on Empirical Software Engineering and Measurement, ESEM 2007*. 7465 (2007), 225–234. DOI= <http://dx.doi.org/10.1109/ESEM.2007.21>.
- [20] A. C. S. Dutra, R. Prikladnicki, and C. Franca, 2015. What Do We Know about High Performance Teams in Software Engineering? Results from a Systematic Literature Review. In *2015 41st Euromicro Conference on Software Engineering and Advanced Applications* (Aug. 2015). 2015 41st Euromicro Conf. Softw. Eng. Adv. Appl. IEEE. 183–190. DOI= <http://dx.doi.org/10.1109/SEAA.2015.24>.
- [21] M. Dixon-Woods, S. Agarwal, D. Jones, B. Young, and A. Sutton, 2005. Synthesising qualitative and quantitative evidence: a review of possible methods. *Journal of Health Services Research and Policy*. 10, 1 (Jan. 2005), 45–53. DOI= <http://dx.doi.org/10.1258/1355819052801804>.
- [22] J. Cohen, 1960. A coefficient of agreement of nominal scales. *Educational and Psychological Measurement*. 20, 1 (1960), 37–46. DOI= <http://dx.doi.org/10.1177/001316446002000104>.
- [23] J. R. Landis, and G. G. Koch, 1977. The Measurement of Observer Agreement for Categorical Data. *Biometrics*. 33, 1 (Mar. 1977), 159. DOI= <http://dx.doi.org/10.2307/2529310>.
- [24] N. E. Fenton, and S. L. Pfleeger, 1998. *Software Metrics: A Rigorous and Practical Approach*. PWS Publishing Co., Boston, MA, USA.
- [25] C. Jones, 2005. Software Cost Estimating Methods for Large Projects. *CrossTalk, The Journal of Defense Software Engineering*. 18, 4 (2005), 8–12.
- [26] M. Ruhe, R. Jeffery, and I. Wiczorek, 2003. Cost estimation for web applications. *25th International Conference on Software Engineering, 2003. Proceedings*. 6, (2003), 285–294. DOI= <http://dx.doi.org/10.1109/ICSE.2003.1201208>.
- [27] M. A. Marks, J. E. Mathieu, and S. J. Zaccaro, 2001. A temporally based framework and taxonomy of team processes. *Academy of management review*. 26, 3 (2001), 356–376.
- [28] A. N. Meyer, T. Fritz, G. C. Murphy, and T. Zimmermann, 2014. Software developers’ perceptions of productivity. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering* (New York, New York, USA, 2014). Proc. 22nd ACM SIGSOFT Int. Symp. Found. Softw. Eng. ACM Press. 19–29. DOI= <http://dx.doi.org/10.1145/2635868.2635892>.
- [29] P. Lenberg, R. Feldt, and L. G. G. Wallgren, 2015. Behavioral software engineering: A definition and systematic literature review. *Journal of Systems and Software*. 107, (Sep. 2015), 15–37. DOI= <http://dx.doi.org/10.1016/j.jss.2015.04.084>.
- [30] D. E. Avison, F. Lau, M. D. Myers, and P. A. Nielsen, 1999. Action research. *Communications of the ACM*. 42, 1 (1999), 94–97. DOI= <http://dx.doi.org/10.1145/291469.291479>.
- [31] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, 2012. *Experimentation in Software Engineering*. Springer Berlin Heidelberg, Berlin, Heidelberg.



# Explanation Templates for Case-based Reasoning in Collaborative Risk Management

Nielsen L. R. Machado<sup>1</sup>, Lisandra M. Fontoura<sup>2</sup>, Rafael H. Bordini<sup>1</sup>, Luis A. L. Silva<sup>2</sup>

<sup>1</sup> Programa de Pós-Graduação em Ciência da Computação, PUCRS. Porto Alegre, RS, Brazil  
nielsen.machado@acad.pucrs.br, rafael.bordini@pucrs.br

<sup>2</sup> Programa de Pós-Graduação em Ciência da Computação, UFSM. Santa Maria, RS, Brazil  
{lisandramf, silva.luisalvaro}@gmail.com

**Abstract** - We have put forward an approach to online collaborative discussion of software development problems based on Argumentation theory. Having records of past discussions can significantly help solve problems in new projects, and CBR techniques are used to retrieve the most similar cases. However, long discussions on past projects still contain too much information to provide support in new discussions. To address this problem, in this paper we introduce the idea of explanation templates that are able to summarize past experiences, particularly for risk management discussions. We formalize this notion of template, introduce the main templates we have developed to support explanation of past experience with risk management, and report the results of a case study on a real-world software project to assess the usefulness of those templates.

**Keywords** – Explanation, Argumentation, Case-Based Reasoning, Risk Management

## I. INTRODUCTION<sup>1</sup>

Explanation and argumentation are intertwined activities in case scenarios of problem-solving. In many applications, argumentation-based dialogues [1, 2] contain rich information in the *locutions* exchanged by users, which in turn allow explanations to be more thorough. In this context, this work investigates a dialogue-based explanation approach [1] to select and display prominent argumentation characteristics recorded in *cases* for Case-Based Reasoning (CBR) systems [3]. In our project, we are particularly interested in risk management discussions conducted by stakeholders of software projects, where *facts and arguments* presented in discussion cases capture these risk management experiences [4-6]. As a result of queries posed by users in such CBR systems, these cases are retrieved from risk management case bases allowing such facts and arguments recorded in the most similar cases to be reused in the analysis of new problems.

To assess the content of cases in order to construct well-grounded experience-based solutions for new problems, different explanation techniques have been discussed in the CBR literature [7-11]. By themselves, *argumentation trees* recording user arguments in the structure of cases can be assessed by users as concrete narrative explanations of problem-solving situations. In a scenario in which CBR becomes a form of explanation-based reasoning, we show that *explanation templates* can be explored to draw users' attention to the most relevant aspects of the cases of interest, such as

“highly discussed and questioned arguments”, and “the balance of pro and con arguments”, for instance. In our project, these argumentation characteristics are directed to the analysis of debate tasks for the identification, analysis, and response planning of risks in software projects [12, 13], in a scenario in which CBR supports the development of experience-based collaborative risk management tasks [4-6]. In effect, the proposed templates allow users to focus on meaningful combinations of project stakeholder moves of argumentation, as for instance, the identification of pros and cons of successful risk proposals, while other user arguments posed in these debates are temporarily omitted when cases retrieved from case bases are inspected. As a way of promoting the reuse of problem-solving information recorded in cases, these templates capture customized forms of selecting and displaying locutions defined in a *dialogue protocol* [5, 14] which is used in the collection and organization of risk management debates. As implemented in the RD System v3.0 [4], the usefulness of the templates was evaluated in two different test scenarios: a case study and a set of experiments involving different participants. The overall results show positive evidence for the usefulness of the templates proposed in the analysis and reuse of argumentation information recorded in the solution of risk management problem situations.

The paper is organized as follows. Section 2 reviews basic concepts of CBR, Argumentation, and explanation in AI. Section 3 then formalizes and informally explains what templates are and how they are used to filter the most relevant information for the users. The proposed templates along with experimental results are discussed in sections 4 and 5, respectively. Finally, final remarks are presented.

## II. BACKGROUND TO THIS WORK

The most common form of explanation in CBR systems is related to the fact that users using these systems are able to find out the most similar cases to a current problem situation. Due to these similarity computations, as reviewed in [7], the case retrieved can offer a concrete explanation about how to make a decision on a new problem. In effect, as a past case was decided in such a way, the current problem also should be decided in this way. That is because those case situations have relevant similarities between them. In practice, past cases reflect real problem-solving situations, often providing convincing support to the conclusions that CBR systems achieve. Precedent-based explanations are also crucial in legal applications [15]. As analyzed in [8], these kinds of

<sup>1</sup> DOI reference number: 10.18293/SEKE2018-098

explanations are frequently attractive to users. Among other reasons, explanations grounded in past experiences are likely to be more convincing for users than explanations based on standardized explanation rules. In CBR, such explanation capabilities can be achieved when filtering and ranking methods are explored. In recommendation problems [10], for instance, besides retrieving a customized list of items, some systems also show reasons to support such recommendations.

Decision-making is subject to discussion in many CBR applications. To reach decisions, case problems can be analyzed through different user arguments. As described in [5, 14], customized dialogue protocols (or sets of *locutions* along with their *interaction rules*) can mediate risk management discussions. As organized in cases for CBR, large numbers of user arguments can be recorded in *argumentation trees* in the body of cases. There, the tree nodes capture the textual content of the user arguments (e.g. Fig. 1), which are indexed through the use of general kinds of locutions such as *ask*, *inform*, *argument\_pro*, *argument\_con*, *withdraw*, *summarize*, etc. Risk management tasks also involve discussions which are particularly directed to the identification, analysis and response planning of project risks [12, 13]. Due to this fact, a dialogue protocol to organize the development of collaborative risk management tasks can also be defined by domain-specific locutions, such as: *propose\_risk* for supporting users to pose risk statements in risk identification tasks; *propose\_probability*, *propose\_impact* to deal with risk analysis tasks; *prioritize\_risk* for the identification of the most important risks; and *propose\_plan* for the statement of risk treatment plans to deal with the prioritized risks.

### III. FORMALIZATION OF TEMPLATES

Explanation templates support project stakeholders by constructing summaries of risk management discussions. Such discussions are most commonly retrieved by users when different kinds of queries are posed in CBR systems (see examples of queries in [4]). Based on standard filtering and tree-traverse techniques applied to the examination of argumentation trees, where the nodes of these trees represent user arguments advanced according to a dialogue protocol for risk management, template-based explanations can be computed. So, templates are detailed in terms of locutions used in the organization of the discussions. When templates are applied on argumentation trees represented in retrieved cases, *selected* parts of those discussions are separately *displayed*, whereas other arguments recorded are temporarily hidden. Note the importance of this combined approach for users: besides using CBR to retrieve and rank the most similar cases, users can then use templates to further examine the retrieval results so that only the most relevant arguments need to be read. We now formalize the idea of templates.

Let  $\mathcal{L}$  be the set of all locutions [5] used in the argumentation-based risk management discussions stored in cases. On top of this, we describe simple expressions (very similar to regular expressions) to define a *constraint* over the stored arguments. We later use these constraints to formalize templates and after that we explain the semantics of each type of expression. Syntactically, the set of all constraints  $C$  is inductively defined as follows:

1. if  $l \in \mathcal{L}$ , then  $l \in C$ ; also  $\_ \in C$ , where  $\_$  is a special symbol used to refer to any  $l \in \mathcal{L}$ .
2. if  $c_1, c_2 \in C$ , then  $(c_1 \cdot c_2) \in C$ ; this is similar to the usual *concatenation* operation;
3. if  $c \in C$ , then  $(c)? \in C$ ; this is the *option* operation, as usual (i.e., it denotes 0 or 1 occurrence of expression  $c$ );
4. if  $c \in C$ , then  $(c)^* \in C$ ; this is the *repetition* operation, as usual (it denotes 0 or more occurrences of expression  $c$ );
5. if  $l \in \mathcal{L}$ , then  $(l, n) \in C$ , where  $n \in \mathbb{N}$ ; this is called the *threshold* operation (we only display arguments within retrieved cases with at least  $n$  occurrences of a locution satisfying  $c$ );
6. if  $l_1, l_2 \in \mathcal{L}$ , then  $(l_1, l_2) \in C$ ; we call this the *balance* operation (the locution of interest is more frequent than its complement);
7. nothing else is in  $C$ .

Then, formally, a template  $t$  is a tuple  $(i, S, D)$  where:

- $i$  is the template ID (a unique name used to refer to that template in the template base);
- $S$  is a set of *selection constraints* for selecting arguments within retrieved cases; a particular argument is only selected for display if it satisfies all constraints  $s \in S$ ;
- $D$  is a set of *display constraints*: arguments within retrieved cases satisfying the constraints in  $S$  are selected to be show to the user, but only the dialogue portions satisfying all constraints  $d \in D$  are displayed.

The semantics of each kind of expression is as follows. A concatenation, such as  $l_1 \cdot l_2$  with  $l_1, l_2 \in \mathcal{L}$ , is used to refer to a dialogue excerpt where a locution  $l_2$  is used within a dialogue context started with locution  $l_1$ . The option and repetition expressions are exactly as in regular expressions. Note that using the  $\_$  operator, concatenation, and repetition we can easily define a constraint saying a particular locution may appear arbitrarily nested after another (e.g.,  $l_1 \cdot \_ * \cdot l_2$ ). These expressions allow for arbitrarily long but finite concatenations of any locutions, although in practice some combinations of locutions are not allowed (see [5]). However, such details are not particularly relevant to our formalization here.

The threshold operation  $(l, n)$  is used to constrain the selection of arguments to those which have at least  $n$  occurrences of locution  $l$ . In practice,  $n$  is implicitly set to be the average number of occurrences of that locution in the retrieved cases. This allows, for example, expressing that only the proposal arguments with most asked questions should be selected (where by “most” we mean above average). Finally, a balance operation  $(l_1, l_2) \in S$  works as follows. We require that either  $l_1 \in D$  or  $l_2 \in D$ , but not both. If  $l_1 \in D$ , a particular proposal argument is only *selected* for display if the number of occurrences of  $l_1$  is greater than that of  $l_2$  (and similarly if  $l_2 \in D$  instead). This is useful for opposing locutions such as, for example, accepting or rejecting an argument. As an example, consider a template to select only the most questioned risk proposals, those which had the most occurrences of *ask*

locutions, assuming the average number of questions typically asked is  $n$ . For the sake of explaining the formalization, assume we want to examine debates where the *ask* locution is above average but we only want the user to see the *ask* questions. Such a simple template could be formalized as simply  $\langle id_i, \{propose\_risk \cdot * \cdot (ask, n)\}, \{propose\_risk \cdot \_ * \cdot (ask, n)\} \rangle$ .

#### IV. EXPLANATION TEMPLATES IN CBR SYSTEMS

Having explained how templates work and formalized them, we now proceed to present (informally only, due to space) various examples of the templates we have developed to support collaborative argumentation-based debate of ongoing risk management situations. In that context, three general types of templates can be identified, as discussed below.

The **argumentation-based explanation templates** select and display general purpose argumentation characteristics in argumentation trees. The template goal is detailed in terms of different arguments that appear when these dialogue protocols are used by users. To capture how and why decisions were taken, the template view of an argumentation tree is focused on properties such as highly discussed and questioned arguments and the balance of pro and con arguments. In dialogue protocols [14], for example, *ask*, *inform*, *argument pro* and *argument con* are standard locutions used in the identification of such argumentation characteristics.

The **domain-specific explanation templates** select and display domain argumentation characteristics in argumentation trees where these moves of dialogue are directly linked to the development of problem-solving tasks in certain application domains. In risk management, prominent debate tasks are the identification, analysis and response planning of risks [12, 13]. In our dialogue protocol for risk management [5], these tasks are mainly identified when project stakeholders use *propose\_risk*, *propose\_impact*, *propose\_probability* and *propose\_plan* locutions, which are used in the identification of domain argumentation characteristics.

The **domain and argumentation-based explanation templates** select and display both domain-specific and general-purpose argumentation characteristics in argumentation trees. In risk management, the use of such templates allows users to understand why decisions were made based on arguments. For example, debate participants can start discussions by advancing risk proposals for targeted projects. In the argumentation subtree which is rooted on such *propose\_risk* locutions, participants can advance arguments not only to ask further information about the proposals posed originally, but also to advance pro and con arguments regarding the relevance of the risks. Having such debates retrieved from CBR queries, a template to select and display identified risks can be combined with a template to highlight proposals that were discussed by a large number of user arguments. So, the resulting template focus on *propose\_risk* that have the largest number of *ask*, *inform*, *argument\_pro* and *argument\_con* locutions.

##### A. Argumentation-based explanation templates

Explanatory argumentation characteristics have an important role in the analysis of debates. Prominent characteristics are the balance of pros and cons for debate proposals, the fact that certain proposals may be subjected to

more lengthy discussions and more heavily questioned than others, etc. Based on such properties, argumentation-based templates are proposed:

(A1) The **template for the most discussed proposals** selects and displays proposals because they are characterized as the most discussed in available cases. To do so, this template analyzes argumentation sub-trees rooted on *propose* arguments, and it checks which ones contain the highest number of user arguments. To show the list of *propose* arguments to users, the decision on whether such proposals are considered as highly discussed is detailed by a threshold value in the template specification. Such decision is taken in the argumentation context of the case retrieved from a given query, since numerical criteria for deciding whether proposals are the most discussed may be different among cases.

(A2) The **template for proposals with the highest number of questions asked** selects and displays proposals because they are highly questioned by users. To do so, argumentation sub-trees rooted on *propose* arguments are identified, and the number of *ask* arguments in these sub-trees is considered. Then, a decision whether proposals are considered as highly questioned is determined by using a threshold value. Using an argumentation tree as input, the template displays to users only the most questioned proposals in the context of a retrieved case, where this argumentation characteristic is then displayed to users.

(A3) The **template for pros and cons of proposals** selects and displays proposals because their analysis is subject to *argument\_pro* and *argument\_con* arguments. Similarly, (A3.1) the **template for proposals with the highest number of pro arguments** and (A3.2) the **template for proposals with the highest number of con arguments** selects and displays proposals because they can be characterized as having the highest number of either pros or cons posed by users. These templates identify more arguable proposals in which users have mostly either agreed with them through the explicit use of pro arguments or disagreed with them through con arguments. To compute the outcome of such templates, argumentation sub-trees rooted on *propose* arguments are identified, and the analysis of user arguments is focused on *argument\_pro* and *argument\_con* locutions only. Moreover, a threshold value is used to assess whether proposals are considered as having high numbers of pros and cons. In the end, these templates display the content of proposals along with their pros and cons.

##### B. Domain-specific explanation templates

Explanation templates can be related to alternative needs of explanation in targeted application domains. In risk management, these needs are directed to the identification, analysis and response planning of risks. To promote the reuse of risk information stored in cases, domain-specific explanation templates are proposed, such as the ones below.

(B1) The **template for identified risk proposals** selects and displays *propose\_risk* arguments in argumentation trees because the identification of such risks is a key task for risk management activities. To do so, the content of *propose\_risk* locutions is displayed to users. When the template is used, it hides other risk management kinds of arguments that may have

been posed by users when dealing with risk analysis and risk response planning tasks as recorded in cases.

(B2) The **template for analyzed risk proposals** selects and displays *propose\_risk* arguments along with their impact and probability analysis. To do so, it considers the *propose\_probability* and *propose\_impact* locutions within argumentation sub-trees with a *propose\_risk* root. That is because probability and impact are commonly analyzed when determining whether a risk is prioritized in a project or not. Instead of displaying only the prioritized risks, this template selects and displays the full set of risks proposed and analyzed in debates. To target this prioritization aspect, (B2.1) the **template for prioritized risk proposals** selects and displays *propose\_risk* arguments along with their impact and probability analysis provided these risks are prioritized in the project. To risk proposals in which a *prioritize\_risk* locution is used by users, (B2.1) selects and displays such most important risks, temporally hiding other non-prioritized risks.

(B3) The **template for risk response plan proposals** selects and displays *propose\_risk* arguments that have been prioritized along with their *propose\_plan* arguments, where the specification of such kinds of (mitigation, transfer, etc.) response plans are fundamental in risk management. To do so, the template displays the content of *propose\_plan* locutions for prioritized risks as recorded in cases.

(B4) The **template for key risk management tasks** selects and displays *propose\_risk* arguments that have been prioritized as a result of debates. Moreover, it selects and displays *propose\_probability* and *propose\_impact* arguments recorded for those prioritized proposals. It also displays *propose\_plan* arguments available in the argumentation sub-trees. In effect, a debate summary is presented to users when this template is used, displaying the main arguments posed by users while identifying, analyzing and planning how to respond to risks.

### C. Domain and argumentation-based explanation templates

To allow project stakeholders to identify risk arguments that led to more critical discussions, we created templates for selecting and displaying the most discussed, the most questioned, and the pros and cons advanced in debates regarding prioritized risks and their response plan proposals.

(C1) The **template for the most discussed prioritized risk proposals** selects and displays prioritized risks along with the impact and probability analysis developed by users to decide on such prioritization. It shows *propose\_impact* and *propose\_probability* arguments, where they are recorded in argumentation sub-trees rooted on risk proposals. To measure the length of prioritized risk discussions, the number of *ask*, *inform*, *argument\_pro* and *argument\_con* kinds of arguments advanced by users is determined. The identification of the most discussed prioritized risks relies on those numerical estimates and defined threshold values. (C1.1) The goal of the **template for the most discussed risk response plan proposals** is similar to the template (C1), but (C1.1) is focused on risks along with their treatment plan proposals. To compute the length of risk response plan debates, the template determines the number of *ask*, *inform*, *argument\_pro* and *argument\_con* kinds of arguments advanced by users in the debate of

*propose\_plan* arguments. The determination of the most discussed plans is computed from argumentation sub-trees that are rooted on each *propose\_plan*. In the end, arguments directly related to risk and plan proposals are the only ones displayed when these templates are used.

(C2) The **template for prioritized risk proposals with the highest number of questions asked** selects and displays prioritized risk proposals along with question-like arguments presented by users in the analysis of these proposals. To identify the prioritized risks with the highest number of questions, the number of *ask* arguments advanced by users is considered. As multiple risks can be prioritized in risk management discussions, this numerical estimate is developed in the context of argumentation sub-trees directly rooted on each *propose\_risk* argument recorded in a case. (C2.1) The **template for risk response plan proposals with the highest number of questions asked** selects and displays prioritized risk proposals along with their risk response plan proposals. In argumentation sub-trees rooted on each *propose\_plan*, this template counts the number of *ask* arguments, while this numerical estimate is considered in each one of the *propose\_plan* locutions advanced in the debates. Considering the most questioned debates, the template selects and displays the prioritized risk proposal arguments along with their risk response plan arguments. In the end, templates focusing on questions asked only display to users the content of these proposals along with questions advanced in their discussions.

(C3) The **template for pro and con arguments for prioritized risk proposals** selects and displays the *argument\_pro* and *argument\_con* arguments regarding prioritized risk proposals. When examining argumentation sub-trees rooted on *propose\_risk*, this template focuses on pros and cons arguments only. Although these pros and cons can be advanced in the debate of different risk management issues, such as in the analysis of probability and impact proposals, for example, this template only selects and displays the pros and cons that are directly related to *propose\_risk*. Similar to (C3), (C3.1) the **template for pro and con arguments for risk response plan proposals** selects and displays *argument\_pro* and *argument\_con* arguments regarding plan proposals. These risk treatment plan arguments are captured by *propose\_plan* locutions recorded in argumentation sub-trees of prioritized risk proposals. Other templates aiming to select and display pros and cons of risk and plan proposals are: (C3.2) the **template for prioritized risk proposals with the highest number of pro arguments**, (C3.3) the **template for risk response plan proposals with the highest number of pro arguments**, (3.4) the **templates for prioritized risk proposals with the highest number of con arguments** and (3.5) the **template for risk response plan proposals with the highest number of con arguments**. In them, similar counting procedures as before are used, except now they are used to determine whether the balance between pros and cons is in favor of either pros or cons. To do so, *argument\_pro* and *argument\_con* arguments recorded on argumentation sub-trees rooted on *propose\_risk* and *propose\_plan* arguments are analyzed, respectively. As a result, only the risk and plan proposals with the highest number of pro or con arguments are displayed to users.

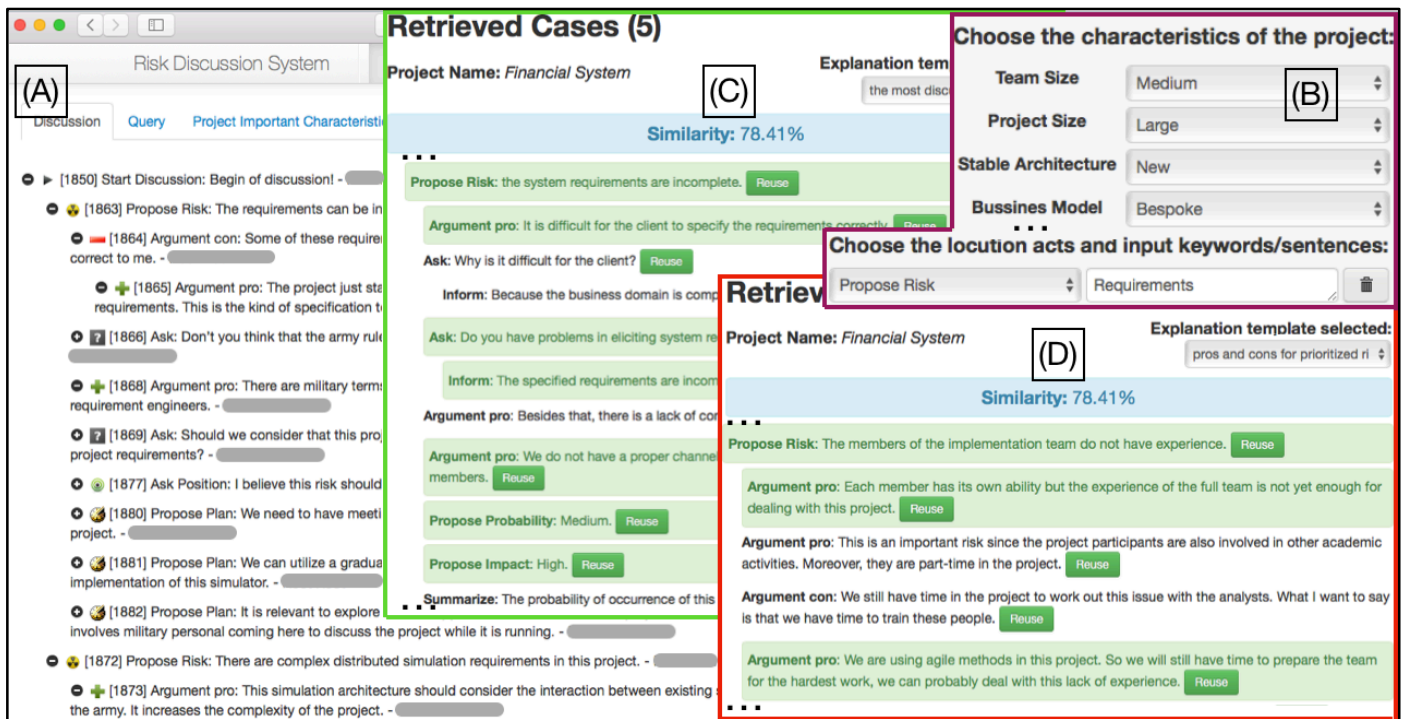


Figure 1 – A print screen of the CBR and explanation templates resources of the RD System

## V. A CASE FOR EXPLANATION TEMPLATES IN EXPERIENCE-BASED COLLABORATIVE RISK MANAGEMENT

The usefulness of the templates proposed was assessed in a case study executed in a R&D project aiming to build a simulation system for the Brazilian Army (e.g. [16]), where key project participants involved were a small group of researchers who were concerned with the management of risks in this project. Based on these participants' point-of-view and experience, the study goal was to examine the usefulness of the explanation-based technique detailed in this paper. Using results of queries posed by the participants in the RD System, templates were used in the analysis of the argumentation content of cases retrieved from a risk management case base.

The evaluation hypothesis (H) and (P) procedures were detailed in this study, where H1 - is the use of queries along with templates helpful in the examination of risk information in past risk management case discussions?; and P1 - to execute different queries and, for each query either use or not templates in the analysis of risk information recorded in the retrieved cases. The procedures were developed in the execution phase of this study having as a result the discussion shown in Fig. 1 (A). To H1, CBR queries based on factual information and pairs (locution, keywords) of current project were executed as in Fig. 1 (B) (see query details in [4]). Then, most of the templates were used in the filtering of the debate details recorded in the retrieved cases. First, the participants explored templates more focused on direct risk information, such as the content of prioritized risk proposals. Later, they explored templates more focused on information collected as a result of longer debates, such as the content of questions and answers (Fig. 1 (C)) and the analysis of pros and cons (Fig. 1 (D)). After that, participants analyzed the risk information that the

templates were selecting and displaying in order to understand why some risks were more heavily discussed than others, for example, using the *template for the most discussed prioritized risk proposals*. In this scenario, some participants listed one prioritized risk from a past case retrieved, stating that such risk would be relevant in the current project. Then, they reused this risk adapting it to the current discussion (Fig. 1 (A) - argument 1863). As far as reuse of past risk information was concerned, most of the time the participants just reused the content of risk proposals. However, they also reused pro arguments to argue in the current risk management situation in favor of the occurrence of past risks examined. For that, the participants used the *template for pro and con arguments for prioritized risk proposals*. In this case, another risk, similar to a possible current situation was listed. Besides that, such risk contained more pro than con arguments on the past discussion. In this sense, the risk and some *argument pro* were adapted on the current discussion. Finally, the participants stated that the reuse of arguments to propose and justify current risk proposals is relevant to help them to overcome debate difficulties they had, bringing more fluency to their risk management analyzes. In the evaluation phase in this case study, for these participants the use of templates allowed them to filter the kind of arguments that they wanted to examine. For them, the use of templates helped to form a quicker understanding of the debates retrieved. So, the summarization of the debates retrieved also had a positive impact on the reuse of past arguments. Finally, the participants stated that the information reviewed and examined via templates allowed them to make more informed decisions in the current problem.

To further evaluate the usefulness of the techniques proposed, 41 CS students developed risk management tasks

using the RD System. In these experiments, a software project was presented to the participants (divided in 9 debate groups). To assess the project risks, they formulated and executed CBR queries in the RD System. As requested, each query was repeated multiple times, allowing them to examine the content of the retrieved cases with and without the use of templates. Among a larger set of questions, they were asked to present their opinion about the following statements: (S1) The way (in particular, without using any kind of template) that past problem-solving experiences are retrieved supports (helps) the tasks of reusing such past information in the solution of the current problem. (S2) The way (in particular, now using the templates available in the system) that past problem-solving experiences are retrieved supports (helps) the tasks of reusing such past information in the solution of the current problem.

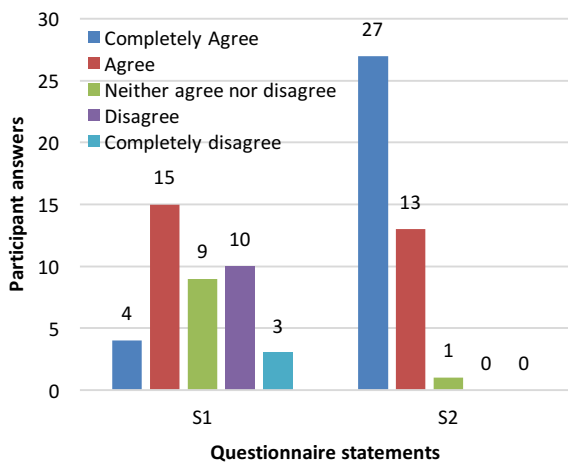


Figure 2 – Participant answers in the proposed questionnaire

According to the questionnaire results (Fig. 2), the use of templates did have a major impact on the participants’ opinion with respect to the analysis and reuse of argumentation information recorded in past risk management experiences. In fact, none of the participants stated that the templates were not helpful in the reuse of past risk information recorded in the retrieved cases, even considering that they had issues when they were asked to analyze the full content of argumentation trees (see the various kinds of answers for S1).

## VI. CONCLUDING REMARKS

Remembering and explaining past risk management experiences is fundamental to the critical analysis of risks in software projects. As a way of avoiding the repetition of past risk management problems, the proposed templates allow users to examine why and how past problems were approached helping the construction of solutions for new risk management problems. In this paper we introduced a new set of explanation templates for CBR systems. We also explore the templates in the analysis and reuse of argumentation information that is recorded in cases that are not just formed by a list of facts as in standard CBR applications. Furthermore, we show the usefulness of the templates in the scenario of experience-based collaborative risk management applications. Future work involves the development of new kinds of experiments to further assess the usefulness of the explanation techniques

proposed and to seek further connections with formal approaches to Argumentation in AI.

## ACKNOWLEDGMENT

We gratefully acknowledge financial support from CAPES-Brazil, and the Brazilian Army through the SIS-ASTROS Project (813782/2014), developed in the context of the PEE-ASTROS 2020.

## REFERENCES

- [1] B. Moulin *et al.*, “Explanation and Argumentation Capabilities: Towards the Creation of More Persuasive Agents,” *Artificial Intelligence Review*, vol. 17, no. 3, pp. 169-222, 2002.
- [2] T. J. Bench-Capon, and P. E. Dunne, “Argumentation in artificial intelligence,” *Artificial intelligence*, vol. 171, no. 10, pp. 619-641, 2007.
- [3] R. Lopez De Mantaras *et al.*, “Retrieval, reuse, revision and retention in case-based reasoning,” *The Knowledge Engineering Review*, vol. 20, no. 03, pp. 215-240, 2005.
- [4] N. L. R. Machado *et al.*, “Case-based Reasoning for Experience-based Collaborative Risk Management,” in *The 26th Int. Conf. on Software Eng. and Knowledge Eng. (SEKE 2014)*, Vancouver, Canada, 2014, pp. 262-267.
- [5] F. S. Severo, L. M. Fontoura, and L. A. L. Silva, “A Dialogue Game Approach to Collaborative Risk Management,” in *The 25th Int. Conf. on Software Eng. & Knowledge Eng. (SEKE 2013)*, Boston, USA, 2013, pp. 548-551.
- [6] D. L. Siqueira *et al.*, “Argumentation Schemes for Collaborative Debate of Requirement Risks in Software Projects,” *Int. Journal of Software Eng. and Knowledge Eng. (IJSEKE)*, vol. 27, no. 9-10, pp. 1613-1635, 2017.
- [7] F. Sormo, J. Cassens, and A. Aamodt, “Explanation in case-based reasoning—perspectives and goals,” *Artificial Intelligence Review*, vol. 24, no. 2, pp. 109-143, 2005.
- [8] P. Cunningham, D. Doyle, and J. Loughrey, “An Evaluation of the Usefulness of Case-Based Explanation,” in *The 5th Int. Conf. on Case-Based Reasoning (ICCBR 2003)*, Trondheim, Norway, 2003, pp. 122-130.
- [9] T. R. Roth-Berghofer, “Explanations and Case-Based Reasoning: Foundational Issues,” in *The 7th European Conf. on Case-Based Reasoning (ECCBR 2004)*, Madrid, Spain, 2004, pp. 389-403.
- [10] D. McSherry, “Explanation in Recommender Systems,” *Artificial Intelligence Review*, vol. 24, no. 2, pp. 179-197, 2005.
- [11] R. C. Schank, *Explanation Patterns: Understanding Mechanically and Creatively*, Mahwah, NJ, USA: Lawrence Erlbaum Associates, Inc., 1986.
- [12] A. Guide, “PROJECT MANAGEMENT BODY OF KNOWLEDGE (PMBOK® GUIDE).”
- [13] B. Bohem, “Software Risk Management: Principles and Practices,” *Software, IEEE*, 1991.
- [14] P. McBurney, and S. Parsons, “Dialogue Games for Agent Argumentation,” *Argumentation in Artificial Intelligence*, I. Rahwan and G. R. Simari, eds., 2009.
- [15] K. D. Ashley, and E. L. Rissland, “Law, Learning and Representation,” *Artificial Intelligence*, vol. 150, no. 1-2, pp. 17-58, 2003.
- [16] J. R. Brondani, E. P. Freitas, and L. A. L. Silva, “A task-oriented and parameterized (semi) autonomous navigation framework for the development of simulation systems,” in *The 28st Int. Conf. on Knowledge Based and Intelligent Information and Engineering Systems (KES 2017)*, Marseille, France, 2017, pp. 534-543.

# Keywords Extraction based on Sentence-Ranking from Chinese Patents

Zhihong Wang\*, Yi Guo\*<sup>†‡</sup>, Tianmei Qi\*

\*Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai, China

<sup>†</sup>School of Information Science and Technology, Shihezi University, Xinjiang, China

**Abstract**—Patent, an important scientific literature, records a large amount of innovative and practical research. The patent keywords also provide a high-level topic description of a patent document and hold an important position in classic NLP tasks, such as patent classification or clustering. However, there are few research works on keywords extraction covering the Chinese patents in current stage. In this paper, we propose a novel algorithm to extract keywords from Chinese patents. A sentence-ranking model, based on a sentence embedding graph and heuristic rules, is constructed to select the top- $K_S$  percent of the sentences. At the same time, the semantic-ranking weights of sentences are also transmitted to keywords extraction. The experimental results on our Chinese patents datasets testifies that the sentence-ranking based keywords extraction algorithm improves the performance by 6% to 13% in F-score. In summary, the new idea of selecting key sentences from original documents can effectively filter out noisy sentences and leverage the performance of keywords extraction.

**Index Terms**—Chinese patents, key sentences, sentence-ranking, keywords extraction

## I. INTRODUCTION

Chinese patent documents has reached 40,673,532 till October 2017 according to the latest statistics of State Intellectual Property Office (SIPO), near a third of global patent documents. Meanwhile, it keeps high growth rates every year, such as the growth rate reached 9% in 2017. Patent is a kind of important scientific literature, which records a large amount of innovative and practical research productions in industry and academia. And we can also speculate on the direction of new technologies and even develop new application areas by analyzing patent bibliographic, changes in patent legal status or citation relations [1]. All in all, it is of great importance to analyze and mine valuable information from mass Chinese patent documents.

Patent keywords provide a high-level topic description of a patent document and play an important role in many applications or tasks of patents. For example, Fujii [2] stated that keywords play a key factors' role in patent translation. Thus, patent keywords get more and more attentions and are widely applied. However, all Chinese patent documents have no author-assigned keywords, which make manually assigning keywords for each patent document very laborious jobs. Therefore, it is highly desirable to extract keywords automatically.

<sup>‡</sup>Corresponding author: guoyi@ecust.edu.cn  
DOI reference number: 10.18293/SEKE2018-034

In this paper, we address the task of automatic keywords extraction from Chinese patents and propose an automatic keywords extraction system for this end and our contributions are as following:

- Construct a sentence-ranking model based on a sentence embedding graph and heuristic rules.
- Optimize the state-of-the-art keywords extraction system (TF-IDF) for Chinese patents based on the sentence-ranking model. The keywords extraction system is named SR based TF-IDF, short for Sentence-Ranking based Term Frequency and Inverse Document Frequency.

The rest of this paper is organized as follows: Section II describes the closely related work; Section III details the architecture of our keywords extraction system; Section IV evaluate our models with dedicated experiments and Section V concludes this paper.

## II. RELATED WORK

Quite a few research works have published about keywords extraction. In general, keywords extraction of Chinese text usually proceeds in four steps: pre-processing, candidate selection, keywords extraction, and post-processing.

In the first pre-processing step, the title and text will be extracted based on specified heuristics rules or extraction algorithms of content main body. At the same time, long texts should be segmented into several paragraphs with paragraph marks (carriage return character, line feeds etc.), and paragraphs sometimes need to be segmented into several sentences with punctuation [3]. In addition, Chinese does not have a clear demarcation between words like English. Some basic operations therefore are needed in the pre-processing step, such as word segmentation, part of speech tagging, new word detection [4] and so on.

The second phase is to determine the keywords candidate collection. So that the remaining steps of keywords extraction is no longer necessary to consider the features of non-candidate words, which will improve the efficiency of keywords extraction. In practice, there are some efficient optimization for candidate extraction. In the KEA algorithm, the candidate keywords are obtained by several basic rules such as the length of keywords, proper nouns and the characteristics of keywords. Csomai et al. [5] experimented with stop-word-filtered n-grams and named entities as potential keywords. However, candidate keywords are still with a wide range and

contain many non-grammatical phrases after selecting with the above rules. So that W. You and D. Fontaine et al. [6] proposed a method to reduce the range or noise of the candidate words. In this method, the top- $k$  words with highest frequencies are defined as the core words, and then the associated words are added into the candidates by word co-occurrence with core words.

The third step - keywords extraction - is quite complicated, because it is not obvious to choose which extraction algorithms (ranking or classification). The most known keywords extraction algorithms are graph-ranking-based algorithms [7] which are derived from PageRank, such as TextRank. Based on the traditional TextRank algorithm, Nan et al. [8] proposed an eccentricity and degree centrality based complex network for keywords extraction, and Li et al. [9] used  $K$ -proximity coupled graph to transfer patents into complex graph model and a patent comprehensive correlation calculation method for quantitative analysis of keyword importance is proposed. [10] pointed out that the external knowledge base can be used to enrich the information to assist in keywords extraction for essay texts in TextRank algorithm.

Besides, the classification algorithms are also efficient in keywords extraction even better have better performance in some fields. In a study by Hasan and Ng [11], TF-IDF was shown to be a surprisingly robust candidate and beaten other more complex ranking strategies. Other important features for keywords classification include TF-IDF, first occurrence position of the word, word diameter, word length and is-in-title etc.

The final important step in keywords extraction is post-filtering, such as filtering short words, limiting the number of two Chinese words [12]. And adjacent words are also sometimes collapsed into phrases, for a more readable output.

Now, the research works on patents mainly focus on patent translation, patent retrieval and patent classification etc. And some other studies primarily concentrate on the research of technological competitiveness about enterprises, industries or regions [13]. By studying the patent through macro, middle or micro aspects, a multi-tridimensional comprehensive evaluation system of technological competitiveness is formed to compare the technological competitiveness among enterprises or other institutions. With the protection of intellectual property rights getting more and more attentions in China, domestic researchers have begun to study the Chinese patent documents. In a study of Liu [14], a semi-automatic patented-technical phrase extraction method was proposed, which achieved good results on Chinese patents and effectively reduced labor cost. In order to improve the patent retrieval speed, [15] linked all the patents by keywords, which were extracted based on an improved TF-IDF algorithms. Moreover, patents are also used to as a background knowledge base, which is helpful to design or realize a better keywords automatic extraction algorithm in other fields [16].

In summary, keywords extraction has made great achievements. However, we are aware that there are only few previous studies about keywords extraction algorithm for Chinese

patents. Therefore, we introduce an novel keywords extraction framework for Chinese patents, and obtain the higher precious, recall and F-measure than the state-of-the-art algorithms or the latest keywords extraction algorithms.

### III. AUTOMATIC KEYWORDS EXTRACTION FROM CHINESE PATENTS

#### A. Overview of the Framework

The framework of keywords extraction from Chinese patents is shown in Fig. 1, which consists of a domain dictionary construction module and a keywords extraction module, and will be detailed next.

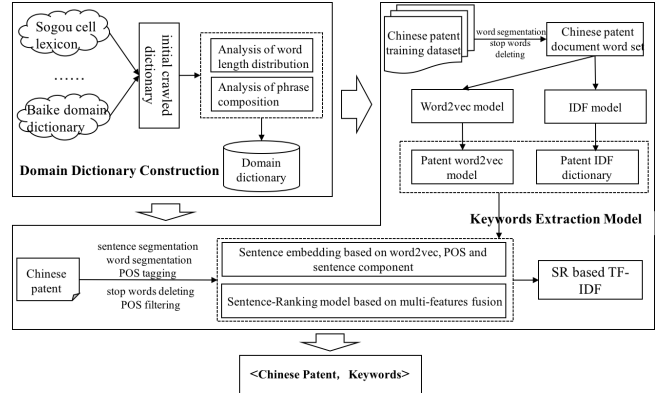


Fig. 1: The framework of keywords extraction from Chinese patents.

#### B. Domain Dictionary Construction

Current keywords extraction algorithms for Chinese texts rely on word segmentation. The higher the quality of word segmentation is, the better results of keywords extraction we shall have. Thus, some new techniques based on external dictionaries [16] or new word detection [17] have been proven to improve the accuracy of Chinese word segmentation and contribute in keywords extraction. Chinese patents obviously contain a large number of professional terms. In order to elevate keywords extraction, a domain dictionary is constructed through merging multi-source heterogeneous external lexicons. 861 lexicons under the “Engineering Application” of Sogou cell lexicon and all vocabulary entries under the scientific classification from Baidu baike are collected, and we get 130 million words (or phrases) in total. After analyzing the initial lexicons, there are several problems emerging in the dictionary, such as plenty of duplicate vocabulary entries with the same meaning caused by English case or Chinese simplified and traditional, or a large number for combined-words.

Several means are used to tackle these above issues in this paper to achieve a higher quality domain dictionary. Firstly, all words in the dictionary are converted into normalized form, that is English in uppercase form and Chinese in simplified form, and delete the repeated entries. Then we analyze the distribution of word length of all words in the dictionary, and keep the words with length from 2 to 7, which is accounted for



0.906. Finally, a word segmentation tool, such as LTP, is used to segment into the most granular words and tag POS for the dictionary words. According to the Chinese word combination rules [18], phrases generally do not contain conjunction (such as “和” (and)), preposition (such as “在” (in)), auxiliary (such as “是” (is)), adverb (such as “很” (very)), and punctuation (such as “.”). Thus, the vocabulary entries, which contain these ban-words, will be deleted. Eventually 284,328 words are obtained after the above process.

### C. Sentence-Ranking based Keywords Extraction Model

A Chinese patent keywords extraction model is proposed based on sentence-ranking model, which integrates the semantic graph and heuristic rules between sentences. The hidden idea is that the keywords is in the key sentences. That is to say, the bigger number of important words (such as keywords) a sentence has, the more important the sentence is. Correspondingly, the more frequently a word appears in an important sentence, the higher the importance of the word is, which is more likely to be the keyword of the document.

Generally, a sentence graph will be built to sort these sentences by the graph sorting algorithm, such as PageRank. In this paper, we introduce a sentence embedding model to better describe the semantic similarity between sentences, and several heuristic rules are also applied to sort sentences. Finally, the top- $K_W$  keywords are extracted by improved the TF-IDF algorithm from the top- $K_S$  percent candidate sentences with highest scores.

1) *Sentence Embedding*: Language models are a very important part of natural language processing, including the classic N-Gram model and the recently widely discussed deep learning model. Word2vec, a word embedding based on deep learning model, takes a large corpus of texts as its input and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. To our knowledge, there is no good sentence embedding model until now. In order to vectorize a sentence, this paper attempts to propose a sentence embedding model on the basis of combining word embeddings produced by word2vec with sentence structural feature with primarily considering the factors, including sentence is a collection of words, words with different POS have different contributions to the sentences and words in different sentence composition have different contributions to the sentences.

For a sentence  $S$ ,  $SW=(sw_1, sw_2, \dots, sw_m)$ , where  $sw_i(i=1, 2, \dots, m)$  is the  $i$ -th word in sentence  $S$ . The corresponding POS of  $SW$  is denoted as  $SN=(sn_1, sn_2, \dots, sn_m)$ , where  $sn_i(i=1, 2, \dots, m)$  represents the POS of the corresponding word  $sw_i(i=1, 2, \dots, m)$ . The word  $sw_i(i=1, 2, \dots, m)$  in  $SW$  can be represented by a  $k \times 1$  vector based on word2vec, name  $V_{sw_i}=[v_{i1}, v_{i2}, \dots, v_{ik}]^T$ , where  $v_{ij} \in \mathbf{R}$ ,  $j=1, 2, \dots, k$ . Thus, the sentence embedding model of this paper is defined as:

$$V_S = \sum_{i=1}^m (w_{pos} + w_{sc}) * V_{sw_i} \quad (1)$$

where  $V_S$  is the sentence embedding of sentence  $S$  and  $m$  is the total number of words in sentence  $S$ .  $sw_i$  represents the  $i$ -th word in sentence  $S$  and  $V_{sw_i}$  is its word embedding produced by word2vec.  $w_{pos}$  is the weight of word with different POS, and

$$w_{pos} = \begin{cases} 0.8, & \text{if } sn_i \text{ is noun.} \\ 0.5, & \text{if } sn_i \text{ is verb.} \\ 0.4, & \text{if } sn_i \text{ is adj.} \\ 0, & \text{if } sn_i \text{ is others.} \end{cases}$$

$w_{sc}$  is the weight of word in different composition, and

$$w_{sc} = \begin{cases} 0.5, & \text{if } sw_i \text{ is subject.} \\ 0.2, & \text{if } sw_i \text{ is predicate.} \\ 0.3, & \text{if } sw_i \text{ is object.} \\ 0, & \text{if } sw_i \text{ is others.} \end{cases}$$

#### 2) Multi-feature Fusion based Sentence-Ranking Model:

This paper firstly gives the symbolic definitions of some variables and formally describes the problems of sentence-ranking in Chinese patents scenario. Suppose that for a Chinese patent document  $P$ , the title is  $T$ . The abstract sentences in  $P$  is  $S$ , and  $|S| = n$ . The goal of sentence-ranking model is to compute an  $n$ -dimensional vector  $SR=[SR_1, SR_2, \dots, SR_n]^T$ , where  $SR_i$  is the weight of the  $i$ -th sentence. So that top- $K_S$  percent sentences with the highest score can be obtained from  $SR$ .

##### (1) Heuristic Rules

Patents, a kind of scientific literature, have strict and standardized templates and writing criterions. The patent name presents the subject and type of the patent in a brief and accurate manner. While, the patent abstract clearly states the technical field to which the patent belongs, the technical issues to be solved by the patent, and the primary technical characteristics and uses of the patent. According to the analysis of patent, the following heuristics rules are considered:

- The more similarity with title a sentence is, the more important the sentence is.
- The sentence in different position is with different importance. Generally, the first and last sentence are more important than others.

Suppose that  $S_i$  is the  $i$ -th sentence in the set  $S$  of patent abstract sentences, and that  $SW_i = (sw_{i1}, sw_{i2}, \dots, sw_{im})$  represents all words in the sentence  $S_i$ , where  $sw_{ij} (j=1, 2, \dots, m)$  is the  $j$ -th word in the sentence  $S_i$ . All words in patent title  $T$  are represented by  $TW = (tw_1, tw_2, \dots, tw_t)$ , where  $tw_i (i=1, 2, \dots, t)$  is the  $i$ -th word in title  $T$ .

Thus, the similarity between the patent title  $T$  and the sentences in patent abstract  $S$  is shown in formula 2.

$$W_{TitleOverlap}(S, T) = [to_1, to_2, \dots, to_n]^T \quad (2)$$

where  $to_i$  is the similarity between the  $i$ -th sentence in patent abstract with the patent title  $T$ , which is calculated by Jaccard similarity,

Meanwhile, the weight of different position of patent abstract sentences is defined in formula 3.

$$W_{location}(S) = [loc_1, loc_2, \dots, loc_n]^T \quad (3)$$

where  $loc_i$  is the location weight of the  $i$ -th abstract sentence in  $S$ . According to the sampling statistics of P.E.Baxendale, 85% of the sentences, which reflect the theme of the document, appears at the beginning of the paragraph, and 7% is in the end. Therefore, the location weight  $loc_i$  of sentence  $S_i$  are defined as follows.

when  $n > 2$ , define  $loc_i$  by

$$loc_i = \begin{cases} 0.85, & \text{if } S_i \text{ is the first sentence.} \\ 0.07, & \text{if } S_i \text{ is the last sentence.} \\ \frac{0.08}{n-2}, & \text{if } S_i \text{ is the other sentences.} \end{cases} \quad (4)$$

When  $n = 2$ , there are only two sentences in the patent abstract, the first sentence and the last sentence, and define  $loc_i$  by

$$loc_i = \begin{cases} 0.89, & \text{if } S_i \text{ is the first sentence.} \\ 0.11, & \text{if } S_i \text{ is the last sentence.} \end{cases} \quad (5)$$

When  $n = 1$ , there is only one sentence in the patent abstract and define  $loc_i$  by

$$loc_i = 1 \quad (6)$$

According to formula 2 and 3, the weight of patent abstract sentences based on heuristic rules proposed in this paper is defined as 7.

$$loc_i = \alpha * W_{TitleOverlap}(S, T) + (1 - \alpha) * W_{location}(S) \quad (7)$$

where,  $\alpha$  is one of the weight parameter of the heuristic rules, the other is  $1 - \alpha$ . While  $W_{TitleOverlap}(S, T)$  is calculated by formula 2 and  $W_{location}(S)$  is calculated by 3.

## (2) A Sentence Semantic Graph

By considering the potential semantic information between sentences, a sentence semantic graph named  $G$  is built based on sentence embeddings, which takes the sentences in patent abstract as the vertex and the similar relation between the sentences as the edges. And PageRank is selected as the graph sorting algorithm in this paper. The adjacency matrix of semantic similarity between sentences is defined as  $P_{sim}(S) = [S_{ij}]_{n \times n}$ , where  $S_{ij}$  is the weight of  $(S_i, S_j)$ , which is defined by the semantic similarity between the  $i$ -th and the  $j$ -th sentence in patent abstract sentences  $S$ .

And the cosine similarity based on sentence embedding 8 is used to calculate the weight of edges in this paper.

$$s_{ij} = \cos(V_{S_i}, V_{S_j}) = \frac{V_{S_i} \bullet V_{S_j}}{\|V_{S_i}\| \|V_{S_j}\|} \quad (8)$$

where  $S_i$  is the  $i$ -th sentence.  $V_{S_i}$  is the  $i$ -th sentence embedding, which is calculated by formula 1.

The iterative formula based on the idea of PageRank, which is used to achieve the sentence weight on the sentence semantic graph  $G$ , is as follows:

$$w_{PR}(S_i) = (1 - d) + d * \sum_{s_{ij} \neq 1} \left[ \left( \frac{s_{ij}}{\sum_k s_{ik}} \right) w_{PR}(S_j) \right] \quad (9)$$

where,  $d$  is the damping factor. And  $w_{PR}(S_i)$ , which can be any non-negative values at initialization, is given by the last iteration in the subsequent iterations.

Like the random walk model, the above iterative process can be converted into matrix operations. Suppose that  $W_{PR}^i$  is the weight vector of patent abstract sentences in the  $i$ -th iteration, then the formula 9 can be re-expressed as:

$$W_{PR}^i = P W_{PR}^{i-1} \quad (10)$$

The above matrix expression gives a more concise iterative process of sentence weight calculation based on a sentence semantic graph. That is, the vector is first initialized with random values and then iteratively updated according to formula 10 until convergence.

In summary, the sentence-ranking model for Chinese patents in this paper is defined as the linear combination of heuristic rules and sentence semantic graph, is as follows:

$$SR(S) = \beta * W_{rule}(S) + (1 - \beta) * W_{PR}(S) \quad (11)$$

where,  $\beta$  is the weight parameter of the heuristic rules. While  $W_{rule}(S)$  is calculated by formula 7 and  $W_{PR}(S)$  is calculated by 10.

3) *Sentence-Ranking based Keywords Extraction Algorithm*: If a sentence contains important information, it and its semantically similar sentences will get higher scores after using the sentence-ranking model. In this way, with the elimination of noise sentences, the effect of keywords extraction from Chinese patents will be greatly improved. However, the sentences of documents are treated equally without considering the semantic importance of different sentences in traditional keywords extraction algorithms. Thus, this paper introduces the semantic weight parameters of sentences produced by sentence-ranking model into the state-of-the-art algorithm TF, so that the semantic importance of sentences can be transferred to the words. TF is defined as follows in this paper.

$$TF(w_i) = \sum_{j=1}^{K_S * n} SR_j * TF_j(w_i) \quad (12)$$

where,  $w_i$  is the  $i$ -th word in patent abstract.  $n$  is the total number of sentences in patent abstract.  $K_S * n$  is the number of sentences with the highest weight.  $SR_j$  is the semantic weight of the  $j$ -th sentence.  $TF_j(w_i)$  is the term frequency of word  $w_i$  in the  $j$ -th sentence.

There are totally six parameters in our algorithm, called SR based TF-IDF. The damping factor  $d$  is generally taken as 0.85 according to PageRank algorithm. The remaining parameters will be discussed next.

## IV. PERFORMANCE EVALUATION

### A. Datasets and Metrics

**Dataset 1.** A large amount of original Chinese patent dataset. The original Chinese patents are collected from SIPO during the period from 2016.11.01 to 2016.11.30. We finally accumulated about 1.21 million well-structured Chinese patents, which will be used to, (1) Train a word2vec model for Chinese patents. (2) Generate a IDF dictionary for Chinese patents. (3) As the source of manually annotated Chinese patent corpus.

**Dataset 2.** A manually annotated patent dataset, which consists of 557 Chinese patents. This dataset is manually annotated by three masters major in computer science, which has the following requirements: (1) Assign 3-6 keywords to each Chinese patent. (2) Keywords with 2-7 Chinese characters in length. (3) Try to select the word whose POS is noun, verb or adjective.

Finally, we adopt the union of pairwise intersections between the annotations as the human-annotated gold standard dataset for Chinese patents [19].

It can be found from the result of manual annotation that the manually annotated keywords are generally long phrases with specific meanings. In this paper, we primarily focus on the keywords that make up these key phrases. In order to better evaluate the performance of keywords, we consider two forms of agreement:

- Exact-Match: when two phrases match exactly.
- Relaxed-Match: when two phrases either match exactly, or can be made identical by adding a single word to the beginning or end of the shorter phrase.

Micro-averaged precision, recall and F-score under these two settings are calculated by the same formula as [19].

### B. Parameters Selection

There are totally six parameters in SR based TF-IDF algorithm. Beside the damping factor  $d$  which is generally taken as 0.85 according to PageRank algorithm. There are five parameters left, including  $\varepsilon$ ,  $\alpha$ ,  $\beta$ ,  $K_S$  and  $K_W$ . Where  $\varepsilon$  determines the convergence rate of SR based TF-IDF. Eventhough  $\varepsilon = 10^{-7}$ , the number of iterations still within 20, so  $\varepsilon$  is taken as  $10^{-7}$  in this paper.  $\alpha$  is used to adjust the weights between different heuristic rules, and this paper treats them equally. So that  $\alpha$  is taken as 0.5 for the two rules each. The remaining four parameters,  $\beta$ ,  $K_S$  and  $K_W$ , will be discussed one by one in what follows.

In order to discuss the remaining parameter values of  $\beta$ ,  $K_S$  and  $K_W$ , SR based TF-IDF are used to extract keywords from Chinese patents, and we respectively calculate the F-score of Exact-Match and Relaxed-Match, as shown in Table 1 and 2.

The relation between  $K_W$  and the optimal F-score is counted as following Table 3 from Table 1 and 2. It is obviously that whatever  $\beta$  and  $K_S$  is, the F-score will obtain more optimal values when  $K_W$  is 4 (underlined-bold numbers in Table 1 and 2). Therefore,  $K_W$  in this paper will be taken as 4 for keywords extraction from Chinese patents.

With the fixed value  $K_W = 4$  and the random value  $K_S$ , F-score of Relaxed-Match can get most optimal values only when  $\beta = 3$  (boxed-underlined-bold numbers in Table 1). However, no matter what the value of  $\beta$  is (3, 4 or 5), they all can get the best F-score of Exact-Match, and F-score gets the most optimal value when  $\beta = 5$ . Thus, to ensure the maximum coverage rate and the minimum average error of optimal F-score in Relaxed-Match and Exact-Match (Table 4), the final value of parameter  $\beta$  is 0.3. Similarly, when  $K_S = 0.85$ , F-score of Relaxed-Match and Exact-Match can obtain the global optimal value. On one hand, less noise data can

**Table 1:** F-score of Relaxed-Match.

$K_S$	$K_W$	$\beta$					
		0.1	0.2	<b>0.3</b>	0.4	0.5	0.6
0.75	Top3	0.533	0.558	0.568	0.560	0.562	0.561
	<b>Top4</b>	0.561	0.573	<b>0.588</b>	<b>0.577</b>	<b>0.575</b>	<b>0.570</b>
	Top5	<b>0.567</b>	<b>0.575</b>	0.572	0.568	0.562	0.556
	Top6	0.551	0.56	0.558	0.552	0.544	0.535
0.8	Top3	0.528	0.558	0.568	0.558	0.564	0.561
	<b>Top4</b>	0.556	<b>0.577</b>	<b>0.587</b>	<b>0.578</b>	<b>0.576</b>	<b>0.573</b>
	Top5	<b>0.562</b>	0.576	0.571	0.566	0.561	0.556
	Top6	0.551	0.560	0.558	0.552	0.543	0.534
<b>0.85</b>	Top3	0.532	0.553	0.567	0.559	0.564	0.561
	<b>Top4</b>	0.55	<b>0.577</b>	<b>0.589</b>	<b>0.583</b>	<b>0.576</b>	<b>0.573</b>
	Top5	<b>0.559</b>	0.576	0.572	0.565	0.561	0.556
	Top6	0.547	0.557	0.558	0.550	0.543	0.536
0.9	Top3	0.533	0.554	0.566	0.559	0.564	0.561
	<b>Top4</b>	0.548	<b>0.575</b>	<b>0.587</b>	<b>0.583</b>	<b>0.576</b>	<b>0.573</b>
	Top5	<b>0.560</b>	0.574	0.576	0.565	0.559	0.555
	Top6	0.549	0.556	0.559	0.551	0.546	0.537

**Table 2:** F-score of Exact-Match.

$K_S$	$K_W$	$\beta$					
		0.1	0.2	<b>0.3</b>	0.4	0.5	0.6
0.75	Top3	0.172	0.203	0.208	0.210	0.210	<b>0.215</b>
	<b>Top4</b>	0.186	<b>0.207</b>	<b>0.211</b>	<b>0.212</b>	<b>0.213</b>	0.211
	Top5	<b>0.198</b>	0.202	0.203	0.201	0.199	0.197
	Top6	0.195	0.197	0.197	0.193	0.191	0.186
0.8	Top3	0.171	0.200	0.207	0.208	0.210	<b>0.215</b>
	<b>Top4</b>	0.185	<b>0.207</b>	<b>0.210</b>	<b>0.211</b>	<b>0.213</b>	0.211
	Top5	<b>0.196</b>	0.200	0.201	0.201	0.199	0.197
	Top6	0.195	0.195	0.196	0.193	0.191	0.186
<b>0.85</b>	Top3	0.173	0.197	0.207	0.208	0.212	<b>0.216</b>
	<b>Top4</b>	0.182	<b>0.206</b>	<b>0.212</b>	<b>0.213</b>	<b>0.213</b>	0.211
	Top5	<b>0.195</b>	0.199	0.200	0.201	0.200	0.197
	Top6	<b>0.195</b>	0.197	0.194	0.191	0.190	0.186
0.9	Top3	0.172	0.197	0.207	0.208	0.212	<b>0.216</b>
	<b>Top4</b>	0.178	<b>0.203</b>	<b>0.211</b>	<b>0.213</b>	<b>0.213</b>	0.211
	Top5	0.194	0.198	0.198	0.199	0.198	0.197
	Top6	<b>0.195</b>	0.196	0.193	0.191	0.190	0.185

be removed because of the normative and refined contents of Chinese patents, which is consistent with the facts. On the other hand, we can still get better keywords through reducing less noise in Chinese patents. So that the final value of  $K_S$  in this paper is 0.85 for keywords extraction from Chinese patents.

### C. Keywords Extraction Results and Discussion

According to section 4.2, the parameters of SR based TF-IDF algorithm have the following values:  $d = 0.85$ ,  $\varepsilon = 10^{-7}$ ,  $\alpha = 0.5$ ,  $\beta = 0.3$ ,  $K_S = 0.85$  and  $K_W = 4$ . In this paper, we compared SR based TF-IDF with a variety of other keywords extraction algorithms, such as TF-IDF, TextRank and the latest word2vec weighted TextRank [20] based on precision, recall and F-score under Exact-Match and Relaxed-Match.

As can be seen from Table 5, the result of the state-of-the-art TF-IDF algorithm and TextRank algorithm is almost the same. The TF-IDF algorithm is simple and effective, and the result is more in line with the actual. However, as the abstracts of Chinese patents are concise, there will be a lot of noise words

**Table 3:** The relation between  $K_W$  and the optimal F-score.

$K_W$	3	4	5	6
the number of optimal F-score	11	<b>64</b>	11	1
the proportion of optimal F-score (%)	0.131	<b>0.762</b>	0.131	0.012

**Table 4:** The relation between  $\beta$  and the optimal F-score.

$K_W$	0.3	0.4	0.5
the coverage rate of optimal F-score	<b>0.643</b>	0.143	0.357
the average error of optimal F-score (%)	<b>0.0022</b>	0.0053	0.0089

**Table 5:** The performance of keywords extraction algorithm from Chinese patents.

Method	Exact-Match			Relaxed-Match		
	P	R	F	P	R	F
SR based TF-IDF	<b>0.223</b>	0.207	<b>0.212</b>	<b>0.623</b>	<b>0.571</b>	<b>0.589</b>
TF-IDF	0.087	0.084	0.085	0.515	0.473	0.487
TextRank	0.105	0.125	0.113	0.466	0.533	0.491
word2vec weighted TextRank (2016)	0.121	<b>0.268</b>	0.165	0.49	0.549	0.518

with same and low frequency, which cannot be distinguished from the keywords in the state-of-the-art TF-IDF algorithm.

In TextRank, a network topology graph is constructed by the co-occurrence relations between words to get the keywords. However, the low co-occurrence of words in abstracts of Chinese patents leads to a sparse words graph, which cannot make good use of the connectivity of network to transfer the weights between words. In order to improve the sparsity of the words graph, a word2vec weighted TextRank [20] is proposed to enrich the semantic relations between words. And the performance is obviously improved from Table 5.

In a word, it is indeed useful for keywords extraction to reduce the noise words and enrich the semantic relationship between words. Then the SR based TF-IDF algorithm proposed in this paper uses a sentence-ranking model to sort the candidate sentences and transfers the semantic weights of sentences into candidate words. Not only consider the semantic relations between words and sentences, the noise of sentences is also reduced. So that the F-score of Exact-Match and Relaxed-Match based on SR based TF-IDF algorithm all achieve the highest score.

## V. CONCLUSIONS

Keywords extraction from Chinese patents is largely an open problem, with potentially important benefits given the growing number of Chinese patents that we have to handle.

In this paper, we build a sentence-ranking model based on a semantic sentence embedding graph and heuristic rules, and use the model to reduce the noise in Chinese patents. Then the semantic weights of sentences based on the sentence-ranking model are used to calculate the weights of keywords, which make sure that the importance of sentences is transmitted to words. Finally, the experimental results on Chinese patents show that SR based TF-IDF algorithm proposed in this paper improves the performance of keywords by 6% to 13% in F-score, which demonstrated the new idea of selecting key sentences from original documents can effectively filter out

noisy sentences and leverage the performance of keywords extraction.

However, the manually annotated patent keywords are always special significant key-phrases. The experiments in this paper concern the keywords that make up those key-phrases, which may cause certain limitations of results. In the future, we will consider the mergence of keywords to get more proper key-phrases to improve the effectiveness of SR based TF-IDF.

## ACKNOWLEDGMENT

This research is financially supported by National Natural Science Foundation of China (grant number 61462073) and Science and Technology Committee of Shanghai Municipality (STCSM) (grant number 17DZ1101003).

## REFERENCES

- [1] Lai, Chaoan, and X. U. Cuilu. "The Application of Patent Mining in the Forecast of Smart Home Industry." *Lancet* 381.9876(2016):1458-9.
- [2] Fujii, A., et al. "Overview of the patent translation task at the NTCIR-8 workshop." *Proc. NTCIR-8 (2010)*:293-302.
- [3] Zhang hongying. "Chinese Key Words Extraction Algorithm." *Computer Systems & Applications* (2009).
- [4] Scientific, Join Faculty Of Computer, et al. "Keyword Extraction Based on New Word Detection." *Microcomputer Information* (2010).
- [5] Csomai, Andras, and R. Mihalcea. "Investigations in Unsupervised Back-of-the-Book Indexing." *FLAIRS Conference*, 2007:231-42.
- [6] You, Wei, D. Fontaine, and J. P. Barthes. "Automatic Keyphrase Extraction with a Refined Candidate Set." *Ieee/wic/acm International Conference on Web Intelligence and Intelligent Agent Technology IEEE Computer Society*, 2009:576-579.
- [7] Beliga, Slobodan, A. Meštrović, and S. Martinčić-Ipšić. "An Overview of Graph-Based Keyword Extraction Methods and Approaches." *Journal of Information & Organizational Sciences* 39.1(2015):1-20.
- [8] Nan, Jiangxia, et al. "Keywords extraction from Chinese document based on complex network theory." *ISCID*, Vol. 2. IEEE, 2014.
- [9] Li, Junfeng, X. Lv, and S. Zhou. "Patent Keyword Indexing Based on Weighted Complex Graph Model." *New Technology of Library & Information Service* (2015).
- [10] Li, Wengen, and J. Zhao. "TextRank Algorithm by Exploiting Wikipedia for Short Text Keywords Extraction." *International Conference on Information Science and Control Engineering IEEE*, 2016:683-686.
- [11] Hasan, Kazi Saidul, and V. Ng. "Conundrums in unsupervised keyphrase extraction: making sense of the state-of-the-art." *International Conference on Computational Linguistics: Posters Association for Computational Linguistics*, 2010:365-373.
- [12] Zhang, Qingguo, et al. "Automatic Keyword Extraction Based on KNN for Implicit Subject Extraction." *Journal of the China Society for Scientific & Technical Information* 28.2(2009):163-168.
- [13] Cao, Ming, et al. "Comparative research on technology competitiveness based on patent analysis." *Studies in Science of Science* (2016).
- [14] Liu, Dacheng, et al. "Technology Effect Phrase Extraction in Chinese Patent Abstracts." *Asia-Pacific Web Conference Springer, Cham*, 2014:141-152.
- [15] Ding, Wei, Y. Liu, and J. Zhang. "Chinese-keyword fuzzy search and extraction over encrypted patent documents." *International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management IEEE*, 2015:168-176.
- [16] Chen, Yiqun, et al. "Mining Patent Knowledge for Automatic Keyword Extraction." *Journal of Computer Research & Development* (2016).
- [17] Liu, Duan Yang, and L. F. Wang. "Keywords extraction algorithm based on semantic dictionary and lexical chain." *Journal of Zhejiang University of Technology* (2013).
- [18] Juan, Y. U., and Y. Z. Dang. "Chinese term extraction based on POS analysis & string frequency." *Systems Engineering-Theory & Practice* (2010): 016.
- [19] Lahiri, S, R. Mihalcea, and P. H. Lai. "Keyword extraction from emails\*." *Natural Language Engineering* 23.2(2016):295-317.
- [20] Wen, Yujun, H. Yuan, and P. Zhang. "Research on keyword extraction based on Word2Vec weighted TextRank." *IEEE International Conference on Computer and Communications IEEE*, 2017:2109-2113.

# Deep Learning based Information Extraction Framework on Chinese Electronic Health Records

Bing Tian <sup>☆</sup>      Yong Zhang <sup>☆</sup>      Kaixin Liu <sup>☆</sup>      Chunxiao Xing <sup>☆</sup>

<sup>☆</sup> RIIT, Beijing National Research Center for Information Science and Technology,  
Department of Computer Science and Technology, Institute  
of Internet Industry, Tsinghua University, Beijing, China

## Abstract

*Electronic Health Records (EHRs) store a large amount of clinical data associated with each patient. Information extraction on unstructured clinical notes in EHRs is important which could contribute to huge improvement in patient health management. Previous studies mainly focused on English corpus. However, at the same time there are very limited research work on Chinese EHRs. Due to the challenges brought by the characteristics of Chinese, it is difficult to apply existing techniques for English on Chinese corpus. In this paper, we propose a deep learning based framework for information extraction from clinical notes in Chinese EHRs. Our framework consists of three components: data preprocessing, feature generation and entity and relation extractor. For clinical entity recognition, we propose a novel Conditional Random Field (CRF) based model and introduce effective features by leveraging the characteristics of Chinese language. For relation extraction, we utilize Convolutional Neural Network (CNN) to obtain high quality entity-relation facts. To the best of our knowledge, this is the first framework to apply deep learning to information extraction from clinical notes in Chinese EHRs. We conduct extensive sets of experiments on real-world datasets from hospital. The experimental results show the effectiveness of our framework, indicating its practical application value.*

**Key words:** *Electronic Health Records; Deep learning; Information extraction; Entity recognition; Chinese*

## 1 Introduction

Electronic Health Records (EHRs) store a large amount of clinical data associated with each patient encounter, including demographic information, current and past diagnoses, prescriptions etc [1]. Information extraction from unstructured clinical notes in EHRs, which serves as the first step towards constructing medical-domain specific knowledge graph, can be beneficial for many fields such as disease inference, clinical decision support systems and risk prediction etc [2, 3, 4]. As such, recently years have seen lots

of studies concentrated on information extraction from English clinical notes.

However, when it comes to Chinese domain, very limited work has been done especially for relation extraction due to the challenges brought by the Chinese clinical notes. On one hand, the different characteristics of Chinese language determine that the methods on English corpus can not be directly applied on Chinese documents. For example, there is no blank space representing word boundaries between Chinese words, and words have no morphological changes in different situations. Besides, some Chinese function words which are important for semantic understanding, such as ”的”, ”了” are often omitted. On the other hand, since there are a large number of professional terms, abbreviations and medical-domain based knowledge contained in clinical notes. It is difficult to adopt existing Chinese-based work focusing on other domains, such as Chinese social media [5, 6], to our problem.

To address these challenges, we propose a deep learning based information extraction framework on clinical notes in Chinese EHRs. Our framework contains three major components: data preprocessing, feature generation and entity and relation extractor. For data preprocessing, we clean the raw corpus and invite medical experts to make necessary annotations. For feature generation, we then select high quality features from multiple aspects according to the characteristics of clinical notes and Chinese language. Finally, we adopt such features in a novel CRF-based model to identify boundaries and type of clinical entities in clinical notes. Next we consider the superior performance obtained by deep learning based methods in information extraction these years and creatively utilize the convolutional neural network (CNN) model in our task. CNN has been used in many fields [7]. Compared with state-of-the-art methods on English documents which heavily depend on manual feature engineering [8, 9], our CNN-based model can achieve better performance while avoiding intensive human labor. We conduct extensive experiments on a real world EHR dataset from a famous medical institute. And the ex-

perimental results demonstrate the effectiveness of our proposed framework.

The rest of paper is organized as follows. Section 2 provides an overview of the existing information extraction approaches. Section 3 introduces our deep learning based information extraction framework. Section 4 and Section 5 respectively describe our clinical entity recognition and relation extraction methods in detail. Section 6 reports the experiments and discusses the results. Finally, we draw our conclusions in Section 7.

## 2 Related Work

In this section, we first review the related work about information extraction on English clinical notes and then introduce the information extraction methods on Chinese and their applications in health-related domain.

Recently, a large amount of work has focused on information extraction on English clinical notes. Due to the unstructured nature, most work utilize the statistical machine learning methods. For example, Seol et al. [10] proposed a clinical Problem-Action relation extraction framework based on CRF and Support Vector Machine(SVM). Skeppstedt et al. [11] studied the usefulness of features extracted from unsupervised methods and applied them in clinical named entity recognition problem. It is noteworthy that these methods have depended on manually engineered features which have seen limited adoption. As such, some recent studies have proposed several methods using deep learning. Jagannatha et al. [12] regarded clinical named entity recognition as a sequence labeling problem and utilized Recurrent Neural Network(RNN) based model. Sahu et al. [13] focused on extracting relations from clinical discharge summaries and exploited the power of CNN to learn features automatically.

Despite the great challenges of information extraction on Chinese documents, there has been a lot of work focused on it recently. For example, in Chinese social media domain, Peng et al. [5] jointly trained word segmentation with an LSTM-CRF model for named entity recognition problem. He et al. [6] further improved the performance for named entity recognition on the same datasets by proposing a unified model combining cross-domain learning and semi-supervised learning. In health-related domain, Yao et al. [14] focused on the text classification on traditional Chinese medicine(TCM) clinical records and proposed a novel method combining deep learning text representation with TCM domain knowledge. He et al. [15] studied the corpus construction of Chinese clinical texts.

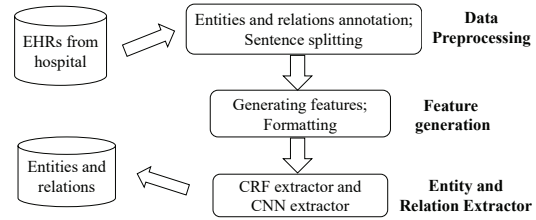


Figure 1: Framework Architecture

```
患\者\无\明\显\诱\因\出\现\胸\痛\B-symptom 痛\I-
symptom, \服\硝\酸\甘\油\B- Treatment 酸\I- Treatment 甘\I- Treatment
油\I- Treatment 可\缓\解\。 \
```

Figure 2: BIO Format of Entity Annotation

## 3 Framework Architecture

The key idea of our framework is extracting clinical entities and relations between them. As shown in Figure 1, there are three main components in our framework: data preprocessing, feature generation and entity and relation extractor. The data preprocessing component processes the raw clinical notes from hospital. Firstly, to generate a high quality corpus for training and testing, we have invited professionals from hospital to help annotate the corpus. And to apply CRF based algorithms to the entity recognition problem, annotated entities should be typically converted into a BIO format. Specifically, it assigns each word into a class as follows: B means the beginning of an entity, I means inside an entity, and O means outside of an entity. For the sentence “患者无明显诱因出现胸痛，服用硝酸甘油可缓解”(No obvious cause of chest pain, taking nitroglycerin can relieve symptoms), the BIO format of annotation is shown in Figure 2. Annotated relations are expressed in a triple format  $[h, r, t]$ , the triple means there exists a relation named  $r$  between the entities named  $h$  and  $t$ . Secondly, considering most relations are existed within one sentence, the preprocessing component splits the clinical notes into sentences using natural language processing tools.

Feature generation component is mainly designed to generate features needed in entity and relation extractor component and normalizes the format of training data so that it can meet the requirements of extractor component. Generally speaking, the data should consist of multiple tokens, and a token consists of multiple columns representing the features.

The entity and relation extractor component learns two extractors: CRF-based clinical entity recognition and CNN-based relation extraction. The two extractor enable extracting clinical entities and relations from clinical notes automatically. Clinical entities and relationships are actually the knowledge contained in clin-

ical notes in health domain so that can be further used in the construction and application of medical-domain specific knowledge graph.

## 4 Clinical Entity Recognition

In this section, we apply the CRF-based model to Chinese Entity recognition problem. First we introduce the features we choose and then we propose our CRF-based model based on these features.

### 4.1 Features

According to the characteristics of clinical notes and Chinese language, we select the bag-of-characters feature, Part of Speech(POS) tag feature, and dictionary feature etc. as our feature sets for clinical entity recognition problem.

**Bag-of-characters feature** As the basic units of Chinese, both characters and phrases can express basic information of Chinese documents. For clinical entity problem, the operations on phrases to generate bag-of-words tend to be more synonymous to complex model than to better performance. So in this paper, we select the bag-of-characters as our feature rather than bag-of-words.

**POS tag feature** Besides bag-of-characters feature, the POS tag information can help improve the efficiency and precision of clinical entity recognition. Through the analysis of clinical notes, we find that different kinds of clinical entities show different characteristics in the POS tag composition. In addition, usually there will be a verb in front of the entity “test” and “treatment” etc. POS tag features can be generated through the existing natural language processing tools.

**Dictionary feature** Clinical notes are highly specialized medical relevant texts which contain a large number of medical terminology. Therefore, the introduction of medical entity dictionaries can effectively improve the accuracy of clinical entity recognition. But there are no such dictionaries available in Chinese domain yet. Considering this situation, we construct a Chinese-based medical dictionary as our feature by cooperating with the professionals from hospital. we first extract numerous clinical entities by referring to large amounts of books and literatures as our basic dictionary and then expand it by crawling and filtering data from Internet. The details of the dictionary are shown in Table 1.

### 4.2 CRF-based Model

In natural language processing domain, CRF is mainly used to solve sequence annotation problems.

Table 1: Medical Entity dictionary

Entity	Example	Number
Disease	胸椎键盘突出 (Thoracic keyboard protrusion)	31450
Medicine	接骨续筋片 (Fracture tablets)	38726
Treatment	齿槽再造术 (Guttural reconstruction surgery)	8493
Test	CT 造影增强扫描 (CT)	3473
Organ	胸口 (Chest)	6089
Physical indicator	血清触珠蛋白 (Serum haptoglobin)	3314

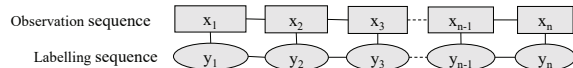


Figure 3: Chain Structure of CRF

Not only can it capture a large amount of human observational experience but also enable capture Markov-chain dependencies between different tags. What’s more, by adding customized features according to specific task, CRF has achieved good results on many entity recognition problems.

In this paper, we regard the clinical entity recognition as a sequence labelling problem. Under this situation, we believe that the CRF dependency graph is a chain structure. And what we attempt to do is modelling the conditional probability of multiple variables by giving their observation values. Specifically, as shown in Figure 3, assuming that the observation sequence is  $\vec{X} = (x_1, x_2, \dots, x_n)$ .  $\vec{Y} = (y_1, y_2, \dots, y_n)$  is the corresponding labelling sequence, and  $y_i$  means the label of the  $i$ th instance of sequence  $X$ . Our goal is to construct the conditional probability model  $P(\vec{Y} | \vec{X})$ . Here,  $\vec{X}$  is the entire Chinese character sequence of a sentence in the clinical notes.  $\vec{Y}$  is the sequence of entity labels corresponding to each word in the sequence  $\vec{X}$ . And we define our feature function as  $f_{a,i}(y_{i-1}, y_i, \vec{X}, i)$ . In this function,  $a \in A$  represents the type of feature,  $x_i$  is the word that we are going to label.  $\lambda_a$  are the corresponding parameters we need to train. For observation sequence  $\vec{X}$  and labelling sequence  $\vec{Y}$ , the conditional probability is as follows:

$$p(\vec{Y} | \vec{X}; \lambda) = \frac{1}{Z(\vec{X}, \lambda)} \exp\left(\sum_{a \in A, y_1, y_2 \in \vec{Y}} \lambda_{a, y_1, y_2}\right) \prod_{i=1}^n f_{a,i, y_1, y_2}(y_{i-1}, y_i, \vec{X}, i) \quad (1)$$

$Z(\vec{X}, \lambda)$  is the regularization term. And the final labelling sequence we get based on this model is as follows:

$$\vec{Y}^* \stackrel{def}{=} \underset{\vec{y} \in \vec{Y}^n}{\operatorname{argmax}} p(\vec{y} | \vec{x}; \lambda) \quad (2)$$

Table 2: The relations and their occurrence frequencies

Relation	Type	Number
Treatment and disease	治疗施加于疾病 (TrAD)	1460
	治疗改善疾病 (TrID)	260
Treatment and symptom	治疗改善症状 (TrIS)	910
	治疗导致症状 (TrCS)	70
	治疗施加于症状 (TrAS)	2760
	因症状未治疗 (TrNAS)	10
Test and disease	检查证实疾病 (TeRD)	440
	为证实疾病而检查 (TeCD)	90
Test and symptom	检查证实症状 (TeRS)	3340
	因症状而检查 (TeAS)	3010
Disease and symptom	疾病导致症状 (DCS)	1930
	症状表明疾病 (SID)	300

## 5 Relation Extraction

Relation extraction is the process of identifying how the given clinical entities are related within the clinical note where they exist. And these relationships contain a lot of clinical semantic knowledge. And these knowledge can then be applied in many fields [16, 17]. For this task, we creatively design a CNN-based model and achieve exciting results. First of all, we identify 12 common relation types. Their names and occurrence frequencies are shown in Table 2.

### 5.1 CNN-based Model Architecture

As shown in Figure 4, in the training process, the outermost layer of the model is initial input. It is the sentence in clinical notes. The last layer refers the output which is a vector and each value of the vector corresponds the possibility of a relation. Besides these, there are 5 more layers in the model including feature layer, embedding layer, convolution layer, pooling layer and fully connected layer.

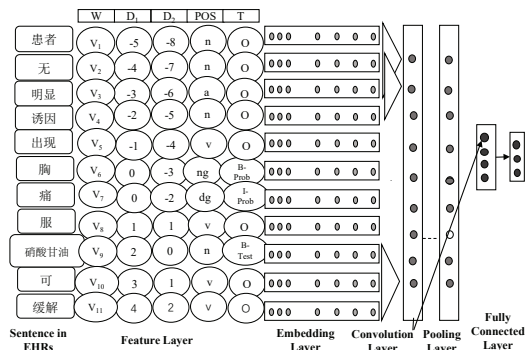


Figure 4: CNN-based Model Architecture

**Feature Layer** In the feature layer, we introduce five features to represent each word which are word itself ( $W$ ), distance to clinical entity one ( $D_1$ ) and entity two ( $D_2$ ), POS tag ( $POS$ ) and entity type of the word ( $T$ ).

- $W$ : the specific words in the sentence

- $D_1$ : the distance between the current word and clinical entity one. Here the clinical entity one and two represent two entities on which we are going to classify relation. And the distance refers to the number of words between the current word and the entity.
- $D_2$ : similar to  $D_1$ ,  $D_2$  refers to the distance between the current word and clinical entity two.
- $POS$ : POS tag of the current word.
- $T$ : the entity type of the current word. And the type is typically converted into a BIO format.

After obtaining these features, we construct a feature dictionary and all the features are ultimately represented by a numerical matrix.

**Embedding Layer** In the embedding layer, each feature corresponds to a vector of the embedding feature matrix. Supposing  $M^i \in R^{n \times N}$ ,  $i \in \{1, 2, \dots, 5\}$  is the embedding feature matrix of the  $i$ th feature (here  $n$  represents the dimension of the feature vector,  $N$  represents the number of possible values of the feature or the size of the feature dictionary), then each column in the matrix  $M^i$  is the representation of the value of  $i$ th feature. Assuming that one hot representation of the  $j$ th value of the  $i$ th feature is  $a_j^i$ , then when the value of the  $i$ th feature is  $j$ , its vector representation  $f_j^i$  is expressed as follows:

$$f_j^i = M^i a_j^i \quad (3)$$

For word embedding, we used word2vec tool<sup>1</sup> to train the word vectors on 55000 clinical notes from a famous medical institute and Q & A data from Chinese medical platform 39 Health<sup>2</sup>.

**Convolution Layer** In convolution layer, we obtain the local features of the sentence by convolution operations. Supposing  $x^1 x^2 x^3 \dots x^m$  is a feature vector sequence of a sentence with length  $m$ , where  $x^i$  is the feature vector of the  $i$ th word and the length of the filter is  $c$ , then the output sequence of the convolution layer is computed as given below:

$$h^i = f(w \cdot x^{i:i+c-1} + b) \quad (4)$$

$f(x)$  is the ReLU function:  $f(x) = \max(0, x)$ .  $w$  and  $b$  are the parameters we need to train.

**Pooling Layer** In the pooling layer, we choose the max-pooling to obtain the global feature of each sentence. Not only does this reduce the dimensions of the output, but it still retains the most salient features.

<sup>1</sup><https://code.google.com/archive/p/word2vec/>

<sup>2</sup><http://www.39.net/>



Table 3: Clinical entities and their frequencies

Entity Type	Number	Entity Type	Number
disease	3110	disease type	220
symptom	24730	test result	3900
test	3230	treatment	4910

**Fully connected Layer** In the fully connected layer, we use the forward propagation to determine the predicted output. Supposing the output of the pooling layer is a vector  $\vec{z}$  whose values come from different filters, then the output  $o$  of connecting layer computed as given below:

$$o = W \cdot \vec{z} + b, (W \in R^{[r] \times l}, b \in R^{[r]}) \quad (5)$$

$[r]$  indicates the number of relationship types.

## 6 Evaluation

We conducted extensive experiments to evaluate the performance of our model for recognizing clinical entities and extracting relations. In this section, we first introduce our experimental settings. And then report the experimental study results.

### 6.1 Experimental Settings

All the experiments are done on the real-world clinical notes collected from Beijing Anzhen Hospital. To obtain well-labeled corpus, we first implemented a tool for annotating entities and relations conveniently. And then we invited two medical experts to help annotate the corpus. Specifically, the corpus contains 2200 clinical notes and more than 2039000 words. For clinical entities recognition, we identify 6 clinical entities including “疾病”(disease), “疾病诊断分类”(disease type), “自诉症状”(symptom), “异常检查结果”(test result), “检查”(test) and “治疗”(treatment). Table 3 shows the clinical entities and their occurrence frequencies.

We used the cross validation and chose three metrics: precision (P), recall (R) and F1-score to evaluate all the results. For clinical entity recognition, we combined different features as inputs to evaluate the effect of each one. For relation extraction, we compared the performance of our CNN-based model with two state-of-the-art SVM-based models.

### 6.2 Experimental Results on Clinical Entity Recognition

For a CRF based model, feature selection is the key to whether the model can achieve good results. To compare the contributions of each feature, we conducted a series of experiments with different features. We started with the model which only use the bag-of-characters feature(W) as our baseline. Table 4 shows the different templates designed with bag-of-characters

Table 4: Templates of bag-of-characters feature

Templates	T1	T2	T3	T4
Window size	4	4	5	5
Feature	bigram	trigram	bigram	trigram
Templates	T5	T6	T7	T8
Window size	6	6	7	7
Feature	bigram	trigram	bigram	trigram

feature. And Figure 5 shows the performance of different templates.

It can be observed that for bag-of-characters feature, the templates with bigram features obtained a better performance than templates with trigram features. And the templates with the context window size of 5 achieved the best performance.

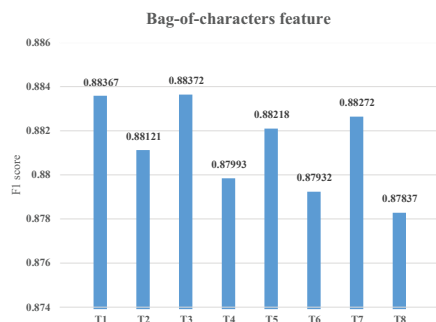


Figure 5: Performance of different templates

Figure 6 describes the performance of the model when POS tag feature(P) and dictionary feature(D) been used. Template 1(T1) is the baseline with only bag-of-characters feature been used. Template 2(T2) to Template 5(T5) respectively added POS tag feature and dictionary feature and used window sizes of 3 to 6. As we can see from the figure 6, with the different size of context window, the templates with the POS tag feature and the templates with the dictionary feature showed the same changing trend and the better performance was generated when the window size is 3. At the same time, the best performance, F1 score of 88.825% was achieved when bag-of-characters feature, POS tag feature and dictionary feature were combined in template 6(T6).

### 6.3 Experimental Results on Relation extraction

**Implementation** While implementing our model, we set the word embedding dimension to be 50 and the other 4 feature dimensions to be 5. In other words, the dimension of each word is 70. In convolution layer, we use the combination of filter lengths 3, 4 and 5 together empirically. And we set the number of filters as 100 for every length. Moreover, we use dropout with a probability of 0.50 to prevent overfitting.

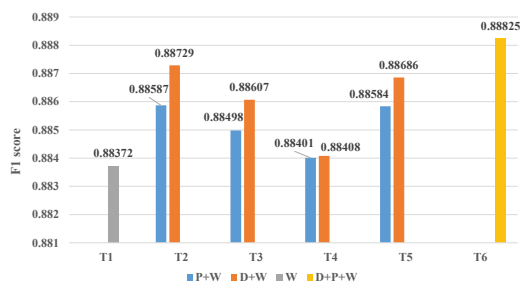


Figure 6: Performance of different feature templates  
Table 5: Comparative performance of CNN based model and SVM based models

Model	P	R	F1 score
CNN	87.7%	76.8%	81.4%
Multi-Class SVM	80.3%	62.8%	70.5%
Single SVM	72.0%	75.3%	73.7%

**Comparison with featured based models** As described before, existing studies for relation extraction problem are mainly based on statistical machine learning methods which heavily depend on manual feature engineering. Here, we compare the performance of our CNN-based model with two state-of-the-art SVM-based models. And we build the SVM classifiers using features defined, respectively, in [8] and [9]. Table 5 shows the comparison of best results obtained by SVM-based models and our CNN-based model.

From the results, we can see that the single SVM model has the lowest precision. But it achieves higher recall than multi-class SVM model since it introduces some new features. And our CNN based model all significantly outperform the two baseline methods, which indicates the effectiveness of our approach.

## 7 Conclusion

We worked on information extraction on unstructured clinical notes in Chinese EHRs from hospital. Our framework consists of three components: data preprocessing, feature generation and entity and relation extractor. For clinical entity recognition, we propose a novel CRF based model and introduce effective features by leveraging the characteristics of clinical notes and Chinese language. For relation extraction, we utilize CNN to obtain high quality entity-relation facts. A series of experimental results showed that our methods are significantly effective comparing with existing state-of-the-art models.

## Acknowledgement

Our work is supported by NSFC(91646202), the National High-tech R&D Program of China (SS2015AA020102), Research/Project 2017YB142

supported by Ministry of Education of The People’s Republic of China, the 1000-Talent program, Tsinghua University Initiative Scientific Research Program.

## References

- [1] G. S. Birkhead, M. Klompas, and N. R. Shah, “Uses of electronic health records for public health surveillance to advance public health,” *Annual review of public health*, vol. 36, pp. 345–359, 2015.
- [2] P. C. Austin, J. V. Tu, J. E. Ho, D. Levy, and D. S. Lee, “Using methods from the data-mining and machine-learning literature for disease classification and prediction: a case study examining classification of heart failure subtypes,” *Journal of clinical epidemiology*, pp. 398–407, 2013.
- [3] M. A. Musen, B. Middleton, and R. A. Greenes, *Clinical Decision-Support Systems*, 2014.
- [4] Y. Cheng, F. Wang, P. Zhang, and J. Hu, “Risk prediction with electronic health records: A deep learning approach,” in *SDM*. SIAM, 2016, pp. 432–440.
- [5] N. Peng and M. Dredze, “Improving named entity recognition for chinese social media with word segmentation representation learning,” in *ACL*, 2016, pp. 149–155.
- [6] H. He and X. Sun, “A unified model for cross-domain and semi-supervised named entity recognition in chinese social media,” in *AAAI*, 2017, pp. 3216–3222.
- [7] J. Wang, Z. Wang, D. Zhang, and J. Yan, “Combining knowledge with deep convolutional neural networks for short text classification,” in *IJCAI*, 2017, pp. 2915–2921.
- [8] A.-L. Minard, A.-L. Ligozat, and B. Grau, “Multi-class svm for relation extraction from clinical reports,” in *Ranlp*, vol. 59, 2011, pp. 604–609.
- [9] B. Rink, S. Harabagiu, and K. Roberts, “Automatic extraction of relations between medical concepts in clinical texts,” *Journal of the American Medical Informatics Association*, vol. 18, no. 5, pp. 594–600, 2011.
- [10] J.-W. Seol, W. Yi, J. Choi, and K. S. Lee, “Causality patterns and machine learning for the extraction of problem-action relations in discharge summaries,” *International journal of medical informatics*, pp. 1–12, 2017.
- [11] M. Skeppstedt, “Enhancing medical named entity recognition with features derived from unsupervised methods,” in *EACL*, 2014, pp. 21–30.
- [12] A. N. Jagannatha and H. Yu, “Bidirectional rnn for medical event detection in electronic health records,” in *NAACL*, vol. 2016, 2016, p. 473.
- [13] S. K. Sahu, A. Anand, K. Oruganty, and M. Gattu, “Relation extraction from clinical texts using domain invariant convolutional neural network,” *arXiv preprint arXiv:1606.09370*, 2016.
- [14] L. Yao, Y. Zhang, B. Wei, Z. Li, and X. Huang, “Traditional chinese medicine clinical records classification using knowledge-powered document embedding,” in *BIBM*. IEEE, 2016, pp. 1926–1928.
- [15] B. He, B. Dong, Y. Guan, J. Yang, Z. Jiang, Q. Yu, J. Cheng, and C. Qu, “Building a comprehensive syntactic and semantic corpus of chinese clinical texts,” *Journal of Biomedical Informatics*, vol. 69, pp. 203–217, 2017.
- [16] K. Zhao, Y. Zhang, Z. Wang, H. Yin, X. Zhou, J. Wang, and C. Xing, “Modeling patient visit using electronic medical records for cost profile estimation,” in *DASFAA*, 2018.
- [17] Y. Zhang, X. Li, J. Wang, Y. Zhang, C. Xing, and X. Yuan, “An efficient framework for exact set similarity search using tree structure indexes,” in *ICDE*, 2017, pp. 759–770.

# Hot Topic Mining based on the Heat of Micro-blog

Wang Siyao

627188726@qq.com

**Abstract**—With the popularity of social networks, users interact with each other and comment on current events through online social network more and more frequently, how to extract the hot topic has become the focus of natural language processing research. In this paper, we propose a hot topic extraction method based on micro-blog popularity. Combining the heat of micro-blog and word2vec model to assign weight for each word, and we use bidirectional LSTM to conduct document semantic coding and single-pass method for topic mining. The experiment results show that the proposed method performs better and has stronger robustness than the traditional topic detection method.

**Keywords**—OSN, topic mining, word2vec, Neural Network

## I. INTRODUCTION

Online Social Networks (OSN) have gradually become popular in daily life and become one of the most important media for users to interact with each other, comment on current events and keep track of topics, like Facebook, Twitter as well as Weibo. By 2016 there are more than 2.2 billion registered users in Facebook. In the second quarter of 2017, there are more than 2 billion active users in Facebook. How to effectively extract the topic of the user's published content has become the focus of the current natural language processing research.

Social media text data contain many new words, abbreviations and emoticons and the length of text is relatively short<sup>[1,2]</sup>. This type of text data contains more noise, so the text matrix yield by topic model is extremely sparse and difficult to analysis and calculate. Commonly used probability topic model LDA and its variants<sup>[3]</sup> can display the semantic information of the texts through taking topic as the expression of the middle layer features. However, this model treats each word as an entity without considering the contextual relationships between words and the temporal dependencies. The machine-learning based approach is mainly used to process ordinary texts, using words as a basic attribute, represent texts as sets of words, and apply machine learning methods for matrix decomposition (LSI, NMF)<sup>[23]</sup>, topic clustering (like k-means, incremental clustering, hierarchical clustering, single-pass clustering, etc.)<sup>[24-26]</sup>. In addition, the development of deep learning provides a new research direction and technical method for the topic mining. Recently, word embedding and recurrent neural network (RNN) have become a research focus of deep learning in natural language processing. The word embedding method is to express the

vocabulary as a dense real number vector on a low-dimensional space, in this way, the expression of lexical semantic features and the construction of language model can be realized.

However, the current study of hot topics mining in the OSN media mainly focus on the level of text, ignoring the interactivity of online social network: every text published by users hide a lot of social information. For example, the more the text was commented and forwarded, the more probability that text contain the hot topic. Therefore, based on the above ideas, we propose a heat-based topic mining method. In this paper, the main works are as followed:

Given that the word2vec model can not distinguish the importance of words in the text, we give each word a value based on the tweets popularity and reconstruct word embedding model.

Using LSTM to encode documents and characterize the dependencies between words, and we apply single-pass clustering method to mine the topic model.

## II. RELATED WORK

In this section, we review the prior study on the issue of hot topic mining. Social media text expression is not standardized, and there is a large number of short text messages, to solve such problems, Danushka and Liu<sup>[11,12]</sup> et.al proposed the use of search engines and hownet to expand the original text with the aim of weakening the low-frequency feature words on the clustering results. However, the methods are implemented by introducing a large number of external features, as a result the excessive time consumption is not suitable for large-scale expected research.

In terms of the expression of words, word embedding has become a very popular tool in recent years and is widely used in every aspects of natural language processing. Collobert & Weston et al.<sup>[27]</sup> proposed a multi-layer neural network model structure based on recurrent neural network for processing variable length word sequence input. In 2012, Eric H. Huang improved the method of C&W and proposed a word embedding training model based on cyclic neural network considering the local and global contexts<sup>[28]</sup>. In 2013, Google's Mikolov team proposed the word2vec word embedding model<sup>[6,7]</sup>, expressing the vocabularies as a dense real number vector on a low-dimensional space, in order to achieve a lexical semantic feature expression. In 2014, Jeffrey and Socher fused the idea of local context information and

global word co-occurrence matrix decomposition to further explore a global linear regression model Glove [29]. Other methods to consider the global context include adding text vectors such as PV-DM and PV-DBOW [30] models and introducing external knowledge base, but due to the particularity of words in natural language processing in terms of semantic expression and multi-word synonymy and so on, how to obtain high-quality representation of word features has always been an important topic in natural language processing and text mining. Sundermeyer et. al [8] explained how to build language models using LSTM neural networks. Kim [9] used CNN to complete the task of sentence classification with the pre-process of word vector. In 2015, Tang et.al [10] used the neural network model consisting of CNN and LSTM to conduct text topic mining. After experimental verification, neural networks composed of CNN and LSTM have achieved good results in topic mining.

With the deep application of deep neural network in the field of natural language processing, Encoder-Decoder framework [16] has been the mainstream method of text sequence modeling in recent years, It also has far-reaching effects on reading comprehension [17], text abstract [18], machine translation [19], automatic question answering [20]. Wang et al. Proposed an end-to-end deep learning framework for fusion question matching to model reading comprehension questions [21]. The framework includes match-LSTM, a match expression model for questions and sentences, and a web-oriented Point Net for answer constraints that effectively enable reading comprehension on large datasets. The popularity of the Encoder-Decoder framework has led to many enhancements to codecs, and attention models [22] are the most powerful and most powerful enhancements available today. The attention model decodes the Decoder output to give a different weight to each input to the Encoder so that the weight of the input that is more important to the current output becomes larger.

### III. PROPOSED METHOD

#### A. Definition of Micro-Blog Heat

Our thinking about the way of calculating blog heat comes from the description of self-information in information theory: a small probability event contains a large amount of information, at the same time a large probability event contains a small amount of information. So the value of the information in event A is defined as formula(1):

$$I(A) = -\log P(A) \quad (1)$$

Learning from the idea of calculating the value of information, assuming that the number of comments on a micro-blog m is r, the forwarding number is s, the definition of the micro-blog heat is shown in formula(2)

$$Heat(m) = -\log \frac{1}{r+s+1} \quad (2)$$

The heat of micro-blog is equal to the sum of the heat of the term in the micro-blog. Therefore, when there are N words in the micro-blog, the heat calculation formula of a single word is described as formula(3):

$$Heat(w) = \frac{Heat(m)}{N} \quad (3)$$

#### B. Word2Vec model

Word embedding is a good way to express lexical features, and the vocabularies are expressed as dense real number vectors in low-dimensional space. Word2Vec is the most widely used word embedding technology, including two kinds of models: CBOV and Skip-Gram. The CBOV model uses the remaining words in the context to predict the probability of generating the target word, and Skip-Gram uses the target vocabulary to predict the probability of other terms in the context. Compared with the CBOV model, Skip-gram has higher semantic accuracy at the expense of higher computational complexity. This paper is based on the Skip-Gram model to improve the training of word vectors, so the prediction of contextual probability is defined as (4):

$$p(\overline{context}(w) | w) = \prod_{\overline{w} \in \overline{context}(w)} p(\overline{w} | w) \quad (4)$$

Word2vec applies a layered softmax function to improve computational efficiency, combined with the layered softmax function, equation (4) can be expanded to formula(5):

$$p(\overline{context}(w) | w) = \prod_{\overline{w} \in \overline{context}(w)} \prod_{j=2}^{l_{\overline{w}}} [\sigma(v_w^T x_{w,j-1}^-)]^{y_{w,j}^-} [1 - \sigma(v_w^T x_{w,j-1}^-)]^{1-y_{w,j}^-} \quad (5)$$

Where w denotes the target vocabulary,  $\overline{w}$  indicates the context of the target vocabulary,  $l_{\overline{w}}$  denotes the the path length of the context in the output layer hierarchy tree,  $v_w$  denotes the input word vector of target vocabulary.  $x_{w,j}^-$  represents the output word vector at the corresponding level under a certain contextual vocabulary.  $y_{w,j}^-$  is logistic output variable.

$$p(y_{w,j}^- | v_m, x_{w,j-1}^-) = \sigma(v_w^T x_{w,j-1}^-) \quad y_{w,j}^- = 0$$

$$p(y_{w,j}^- | v_m, x_{w,j-1}^-) = 1 - \sigma(v_w^T x_{w,j-1}^-) \quad y_{w,j}^- = 1$$

Where  $\sigma(\cdot)$  represents sigmoid function.

When training the model of word embedding, the robustness of the model is often enhanced by introducing some words that are not found in the corpus as negative samples, these negative samples can be pre-generated by

negative sampling<sup>[13]</sup>. The objective function of Skip-Gram model trained with negative sample is shown in formula(6):

$$L = \sum_{w \in C} \sum_{\bar{w} \in \text{context}} \sum_{(w)u \cup \{w\} \cup N_{\bar{w}}} \{y_{wu} \log[\sigma(v_w^T x_u)] + (1 - y_{wu}) \log[1 - \sigma(v_w^T x_u)]\} \quad (6)$$

Where  $N_{\bar{w}}$  is a negative sampling set under the vocabulary  $\bar{w}$ ,  $y_{wu}$  is logistic output variable,  $v_w$  is the sum of word embedding of context,  $x_u$  denotes the parameter of the model.

### C. Word2Vec based on micro-blog heat

Given training corpus dictionary  $\text{vocab} = \{t_i | i \in 1 \dots N\}$

and document  $d_i = \langle w_1, w_2, \dots, w_j \rangle$  where  $N$  is the word vector dimension. We use Word2vec model to train the corpus to get the word vector, accumulate the word vectors in the text  $d_i$  to get the vector representation of the text  $d_i$  shown in formula(7)

$$R(d_i) = \sum_t \text{word2vec}(t) \quad \text{where } t \in d_i \quad (7)$$

Next, we introduce the heat model to calculate the word weight in Word2vec model according to the heat of words, and the the weighted word vectors are accumulated to obtain a new vector representation of document  $d_i$  shown in formula (8)

$$\text{weight} \_ R(d_i) = \sum_t \text{word2vec}(t) \times \text{Heat}_i(w) \quad (8)$$

### D. Document Semantic Coding Based on LSTM

Automatic coding machine is a artificial neural network, using self-supervised learning to encode input samples, in order to achieve the purpose of reducing the data sample dimension. It is mainly divided into two parts: Encoder and Decoder. The encoder mainly compresses the original data into the output code:

$$\phi : X \rightarrow Z$$

The decoder will restore the output code closely to the original data:

$$\varphi : Z \rightarrow X'$$

In order for the automatic coder to retain the primary information in the original sample, the optimization goal is set as formula(9):

$$\phi, \varphi = \arg \min_{\phi, \varphi} \|X - \varphi(\phi(X))\|^2 \quad (9)$$

LSTM coding framework is divided into five layers, taking the word embedding expression of all the keywords in the document as input and the overall semantic embedding of the document as output. The details of these five levels are as follows:

**Word embedding presentation layer:** This layer is the input layer. Because some documents tend to have more lexicons and some unimportant words that have no direct effect on the expression of the document's theme, in this paper we only select the word embedding with larger heat value as input, and the word embedding for all words is obtained from word2vec model above.

**Bidirectional LSTM hidden layer:** Contains two LSTM hidden layers, forward and backward layer, at the same time, each word embedding is connected to both the forward and backward LSTM hidden layer unit, these two hidden layers are connected to the same output. The input word at the moment  $t$  is embedded as  $E_t$ , the output to the forward LSTM hidden layer cell is  $h_{t-1}^f$ , output to the backward LSTM hidden layer cell is  $h_{t-1}^b$  for the last moment. The output of the forward and backward layers at the current moment are shown in formula(10,11):

$$h_t^f = H(E_t, h_{t-1}^f, c_{t-1}, b_{t-1}) \quad (10)$$

$$h_t^b = H(E_t, h_{t+1}^b, c_{t-1}, b_{t-1}) \quad (11)$$

Where  $H(\cdot)$  denotes the function of LSTM hidden layer,  $c_{t-1}$  indicates the status value of the Cell unit at the last moment,  $b_{t-1}$  denotes the offset at the last moment.

**Bidirectional LSTM output layer :** Each output cell is connected to both the forward and backward LSTM hidden layer cells at same moment.

$$g_1 = \sigma(W_{hg}^f h_t^f + W_{hg}^b h_t^b + b_g) \quad (12)$$

Where  $W_{hg}^f$  and  $W_{hg}^b$  are respectively the connection weights between the forward, backward hidden layer and the bidirectional LSTM output layer,  $b_g$  denotes the offset.

**Average pooling layer:** We use average pooling to process the original eigenvalues and construct new features, as well as realizing the dimension reduction, enhancement and noise filtering of the original valid features. Through averaging all cell values over a certain range, local information can be taken into account, calculation is shown in formula(12):

$$pool(g) = \sum_{t=1}^T \frac{g_t}{T} \quad (12)$$

Where T is the length of the input word embedding sequence.

Semantic encoding output layer: The result of the average pooling layer can be calculated by activating the function to get the final semantic coding vector of the entire document

The dimension of the semantic code vector is consistent with the dimension of the input word embedding in order to facilitate similar calculation.

#### E. Hot topic clustering

After applying LSTM to semantic encoding, each document exists in the form of a vector in the hidden subject space, and its dimension is much lower than that of the feature space in vector space model. Therefore, we use Single-Pass clustering algorithm to conduct document clustering. After that we get the number of clusters K (that is, the number of topics), and the results of the division of each document on the K topics in the document set.

### IV. RESULT EVALUATION

Data set: We used crawlers to crawl over 10,000 micro-blogs in 15 hot topics in 2016 in Sina Weibo, and selected 200 micro-blogs from each topic, recording the number of each comments and forwarding on each micro-blog. As a result we take a total of 3000 micro-blogs as the experimental data set.

The data set need to be pre-processed before model training, including stop words, high and low frequency words, illegal characters. The word with the frequency of less than 5 in the corpus is considered as the low frequency word, and the word with the frequency more than 20% of the total number of words is considered as high frequency word.

#### Evaluation Measurement

For the results of the hot topic mining, we use the purity and the normalized information (UMI) as the evaluation measurement, and these two evaluation methods are applicable to the text data with label.

Purity: Purity is used to measure the proportion of correctly clustered documents in the total document. The greater the purity is, the better effect the topic clustering yields. The purity value is calculated as formula(13):

$$purity = \frac{1}{D} \sum_{i=1}^k \max_j |p_i \cap c_j| \quad (13)$$

Where D is the total number of micro-blogs, k is the number of topic clusters,  $p_i$  represents the set of words contained in the i-th cluster,  $c_j$  denotes the j-th micro-blog in the corpus.

normalized mutual information (UMI): The NMI value is calculated as formula(14):

$$NMI = \sum_{t=1}^K \sum_{1 < i < j < K} \frac{\log \frac{P(w_i, w_j)}{P(w_i)P(w_j)}}{-\log P(w_i, w_j)} \quad (14)$$

Where K represents the number of topics, N is the first N words under the topic.  $P(w_i, w_j)$  denotes the co-occurrence probability of word  $w_i, w_j$ .  $P(w)$  is the probability of the word w under the topic k.

Point-wise Mutual Information, PMI is the most commonly used semantic coherence evaluation measurement<sup>[14,15]</sup>. The results of PMI evaluation are often highly consistent with manual evaluation. The higher PMI scores, the stronger semantic coherence topic have. The PMI value is calculated as formula(15):

$$PMI(\phi_k) = \frac{2}{V(V-1)} \sum_{1 \leq i < j \leq V} \log \frac{p(w_i, w_j)}{P(w_i)P(w_j)} \quad (15)$$

Where  $p(w_i)$  is the probability of appearance of vocabulary  $w_i$  in the test document set.  $p(w_i, w_j)$  represents the joint probability of vocabulary  $w_i$  and  $w_j$  in the test document set, V is the dimension of the vocabulary list. Therefore, in this paper, for each subject word distribution, we only select the first 10 words with the highest probability value to calculate PMI.

#### Result analysis and comparison

We also implement the classic statistical-based TF\*IDF, NTM and LDA algorithm to extract key words from micro-blog. We conducted several sets of comparative experiments. In the experiment, we set the number of topics as 6, 20, 40, 80 respectively for the experiment of purity, and we set the number of topics as 20, 40, 60, 80, 100 respectively for experiment of NMI. The dimension of word vector size is 300, the experimental results are shown in Fig.1 and Fig.2

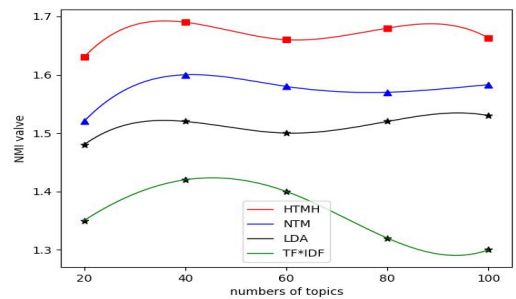


Fig.1 NMI value

## V. CONCLUSION

In this paper, we proposed a hot topic mining method based on micro-blog heat, based on the word2vec, we combined the idea of information theory and gave weight to each word. After using bidirectional lstm for semantic encoding, we applied single-pass clustering algorithm to conduct hot topic mining.

We then compared MTMH(Hot Topic Mining Based on the Heat of Micro-blog) model with TF \* IDF, LDA and NTM algorithms, and introduced measurement of the topic mining quality, such as purity, NMI and PMI. Through the weibo corpus we proved that the HTMH model performs better on topic mining.

Future research focuses on the impact of sequence on the distribution of topics, in addition, collaborative training of feature expression of topic detection, emotion analysis and word embedding is also the trend of large-scale social media data analysis.

## REFERENCES

- [1] Losada D E. The challenge of understanding the flow of sentiments in social media documents[C]// International Workshop on Search and Mining User-Generated Contents. ACM, 2011:1-2.
- [2] Liu H. Mining social media: issues and challenges[C]// ACM Sigm International Workshop on Social Media. ACM, 2011:1-2.
- [3] Blei D M, Ng A Y, Jordan M I. Latent dirichlet allocation[J]. Journal of Machine Learning Research, 2003, 3:993-1022.
- [4] Bengio Y, Vincent P, Janvin C. A neural probabilistic language model[J]. Journal of Machine Learning Research, 2006, 3(6):1137-1155.
- [5] Sutskever I, Vinyals O, Le Q V. Sequence to Sequence Learning with Neural Networks[J]. 2014, 4:3104-3112.
- [6] Mikolov T, Chen K, Corrado G, et al. Efficient Estimation of Word Representations in Vector Space[J]. Computer Science, 2013.
- [7] Mikolov T, Sutskever I, Chen K, et al. Distributed Representations of Words and Phrases and their Compositionality[J]. Advances in Neural Information Processing Systems, 2013, 26:3111-3119.
- [8] Sundermeyer M, Schlüter R, Ney H. LSTM Neural Networks for Language Modeling[C]// Interspeech. 2012:601-608.
- [9] Kim Y. Convolutional Neural Networks for Sentence Classification[J]. Eprint Arxiv, 2014
- [10] Tang D, Qin B, Liu T. Document Modeling with Gated Recurrent Neural Network for Sentiment Classification[C]// Conference on Empirical Methods in Natural Language Processing. 2015:1422-1432
- [11] Bollegala D, Ishizuka M, Matsuo Y. Measuring semantic similarity between words using web search engines[J]. Computer Science, 2015:757-766.
- [12] Liu Z, Yu W, Chen W, et al. Short Text Feature Selection for Micro-Blog Mining[C]// International Conference on Computational Intelligence and Software Engineering. IEEE, 2010:1-4.
- [13] Mikolov T, Sutskever I, Chen K, et al. Distributed Representations of Words and Phrases and their Compositionality[J]. Advances in Neural Information Processing Systems, 2013, 26:3111-3119.
- [14] Newman D, Lau J H, Grieser K, et al. Automatic evaluation of topic coherence[C]// Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 2-4, 2010, Los Angeles, California, USA. DBLP, 2010:100-108.
- [15] Jey Han Lau, David Newman, Timothy Baldwin. Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In EACL'14, pp.530-539, 2014
- [17] Seo M, Kembhavi A, Farhadi A, et al. Bidirectional Attention Flow for Machine Comprehension[J]. 2016.

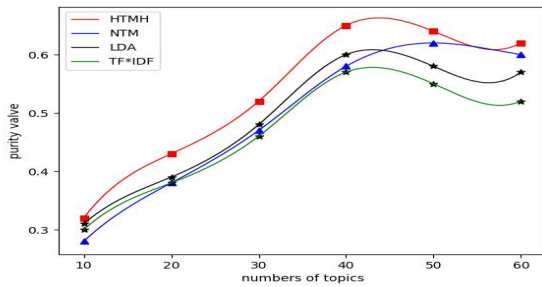


Fig.2 Purity valve

From the above two figures we can see, under the Sina Weibo corpus, the proposed HTMH(Hot Topic Mining Based on the Heat of Micro-blog) model has some improvement over other models in terms of purity and normalized mutual information, the purity reaches the maximum when the number of topics is 40, and the normalized mutual information (NMI) gets the highest accuracy when the number of topics is 20. This is also consistent with other models, HTMH has increased by 3% compared with TF \* IDF, and increased by 5% compared with LDA. This is because word embedding technique is introduced as a semantic supplement. This makes the semantic relationship between words and subject strengthened. And in the follow-up training process, word embedding and topic model training promote each other, so we can identify the topic more accurately.

We can see from Fig.3, the HTMH model proposed in this paper has a generally higher PMI value. It shows that the extracted hot topic has a strong semantic coherence and as the number of subjects increases, the PMI value basically remains unchanged. The traditional LDA model has the lowest PMI because it does not consider the semantic reinforcement of documents and words, NTM is a neural network reconstruction of the LDA model, but it does not take into account the reinforcement of vocabulary and semantics. So it performs relatively poorly.

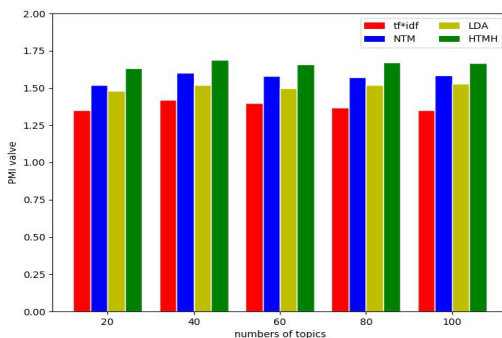


Fig.3 PMI valve

- [16] Sutskever I, Vinyals O, Le Q V. Sequence to Sequence Learning with Neural Networks[J]. 2014, 4:3104-3112.

- [18] Lopyrev K. Generating News Headlines with Recurrent Neural Networks[J]. Computer Science, 2015.
- [19] Cho K, Merrienboer B V, Bahdanau D, et al. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches[J]. Computer Science, 2014.
- [20] Pengcheng Yin, Zhengdong Lu, Hang Li, et al. Neural Enquirer: Learning to Query Tables with Natural Language[J]. Computer Science, 2016.
- [21] Wang S, Jiang J. Machine Comprehension Using Match-LSTM and Answer Pointer[J]. 2016.
- [22] Bahdanau D, Cho K, Bengio Y. Neural Machine Translation by Jointly Learning to Align and Translate[J]. Computer Science, 2014.
- [23] Wang Q, Xu J, Li H, et al. Regularized latent semantic indexing[C]// International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, 2011:685-694.
- [24] Yamron J P, Knecht S, Mulbregt P V. Dragon's Tracking and Detection Systems for the TDT2000 Evaluation[J]. Proceedings of the Broadcast News Transcription & Understanding Workshop, 2000:75--79.
- [25] Charikar M, Chekuri C, Motwani R. Incremental clustering and dynamic information retrieval[C]// Twenty-Ninth ACM Symposium on the Theory of Computing, El Paso, Texas, Usa, May. DBLP, 1997:626-635.
- [26] Corpet F. Multiple sequence alignment with hierarchical clustering.[J]. 1988.
- [27] Collobert R, Weston J, Karlen M, et al. Natural Language Processing (Almost) from Scratch[J]. Journal of Machine Learning Research, 2011, 12(1):2493-2537.
- [28] Huang E H, Socher R, Manning C D, et al. Improving word representations via global context and multiple word prototypes[C]// Meeting of the Association for Computational Linguistics: Long Papers. Association for Computational Linguistics, 2012:873-882.
- [29] Pennington J, Socher R, Manning C. Glove: Global Vectors for Word Representation[C]// Conference on Empirical Methods in Natural Language Processing. 2014:1532-1543.
- [30] Le Q V, Mikolov T. Distributed Representations of Sentences and Documents[J]. 2014, 4:II-1188.



# SocialGQ: Towards Semantically Approximated and User-aware Querying of Social-Graph Data

Riccardo Martoglia

FIM Department, University of Modena and Reggio Emilia, I-41125 Modena, Italy,

E-mail: [riccardo.martoglia@unimore.it](mailto:riccardo.martoglia@unimore.it)

**Abstract** – The proliferation of social and collaborative sites makes users increasingly active in the generation of social-graph data; however, such sea of data often hinders them from finding the information they need. In this paper, we present SocialGQ (“Social-Graph Querying”), a novel approach for the effective and efficient querying of social-graph data overcoming the limitations of typical search approaches proposed in the literature. SocialGQ allows users to compose complex queries in a simple way, and is able to retrieve useful knowledge (top-k answers) by jointly exploiting: (a) the structure of the graph, semantically approximating the user’s requests with meaningful answers; (b) the unstructured textual resources of the graph; (c) its social and user-aware dimension. An experimental evaluation comparing SocialGQ to leading approaches shows strong gains on a real social-graph data scenario.

**Keywords** – social-graph, knowledge management, approximate querying, user-aware techniques, semantic retrieval.

## 1. Introduction

In recent years, the web has evolved from a static web, where users consume information, to a “social web” where they are also able to produce them. The proliferation of social networks (e.g., Facebook, LinkedIn, ...), social bookmarking and collaborative tagging sites (e.g., BibSonomy, CiteULike and Delicious), social question-answering sites (e.g., Stack Overflow), microblogging sites (e.g., Twitter) and social components on “traditional” websites makes users increasingly active in the generation of content. This ever increasing amount of “social-graph” data is quite peculiar in its composition. First of all, it is typically characterized by a dual nature, both *graph-structured* (users/resources as nodes and relations, such as friendships, as arcs), and *unstructured* (the large amount of textual resources, e.g., documents and comments). Moreover, the *social/user-aware* component is obviously quite prominent, where the available nodes and resources are possibly created/modified by different users. Consider, for instance, the small excerpt from the BibSonomy graph data [9] depicted in Fig. 1, showing two content resources (nodes) with their associated annotations. In

particular, the BibTeX instance “C19837” (node  $n_{13}$ <sup>1</sup>) has been annotated with the “semantics” tag by user “U150” ( $n_4$ ), while Bookmark instance “C45829” ( $n_{14}$ ) has been tagged as “semantic web” by user “U881” ( $n_{12}$ ). The social nature of the data is highlighted in figure by small symbols (circle, triangle) depicting the nodes modified by the two users. Note that entity nodes (including instances) are depicted with rounded corners, differently from simple value nodes, representing textual content (e.g., abstracts, descriptions) and other attributes (e.g., publication years). Finally, consider the strong semantic characterization of such data, given by type hierarchies (both BibTeXs and Bookmarks are defined as Contents) and also user-specified tag hierarchies (for instance, user “U881” defined the “semantic web” tag as a specialization of the “semantics” tag).

In this context, a crucial problem is to extract useful knowledge from this wealth of information. Users should be able to compose complex queries (beyond the classic keyword model) in a simple way, allowing them to quickly find all the relevant information with respect to the their needs. Consider for instance the following three examples of information need:

- Q1.** Find (not already known by me) content having a summary about “semantic” and “text”;
- Q2.** Find BibTeX entries semantically related to BibTeX titled “Versatile...”;
- Q3.** Find documents that have been tagged as “semantics” (also considering sub-tags defined by me).

Such requests go well beyond what typical websites and standard search tools allow: they are very difficult or even impossible to express with simple keywords; they require semantic approximation in order to be effectively solved on the graph data, both in terms of labels (e.g., terms like “summary” and “document” are not defined in the data) and structure (e.g., Q1 asking for “content” should retrieve all its subclasses, i.e., both BibTeX and Bookmark entries; Q2 asks for BibTeXs generically related to the named one; Q3 also asks for content tagged with sub-topics); they also require unstructured content full-text search capabilities going beyond exact search (e.g., Q1), as well

<sup>1</sup>For clarity’s sake, nodes are univocally identified by the node ids  $i$  shown on the left upper corner and will be referenced as  $n_i$ ; moreover, some type relationships (e.g., users, blank nodes) are omitted from the figure.

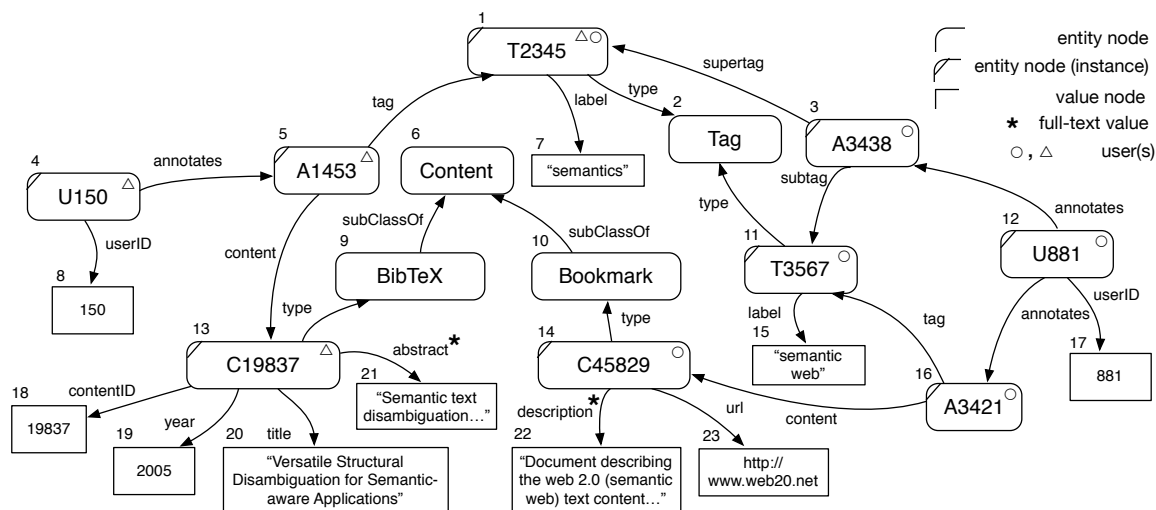


Figure 1. An excerpt from the BibSonomy graph data

as user-awareness (e.g., Q1, Q3); finally, they require an effective ranking in order to provide first the most relevant results.

Generally speaking, currently available search tools do not support all the above-mentioned requirements, typically making such complex requests impossible to be solved automatically, i.e., without requiring a lot of manual effort from the user. Typical social sites search capabilities are keyword-only, limited to exact search in specific fields, syntactic-only (absence of semantics) and typically disregard the social nature of the data (i.e., who created what). Similarly, looking at the current research state of the art, there are several (approximate) graph querying [4, 8, 14], user-aware textual search [12, 15] and social search [6, 7] proposals; however, no one combines all the required features in a single solution applicable to the context of the social-graphs.

In this paper, building on the acquired know-how on graph data management [11], we focus on defining the foundations of SocialGQ (“Social-Graph Querying”), a novel approach for the effective and efficient querying of social-graph data. SocialGQ is aimed to overcome the limitations of current approaches, by jointly exploiting: (a) the structure of the graph, by means of approximation techniques capable of semantically approximating the user’s request with meaningful answers; (b) the unstructured textual resources present in the contents of the graph; (c) the social and user-aware dimension. The capacity for semantic approximation also responds to the need for simplicity in the composition of the query itself, intelligently adapting the request to the graph. The final aim is to retrieve the most useful (top-k) answers in an efficient and automated way.

Fig. 2 provides an overview of the SocialGQ architecture: the Data Manager module (described in Sect. 3) organizes the social-graph data into ad-hoc data structures efficiently supporting the semantic approximation, full-text and user-aware requirements. The Query Processing and Ranking modules (Sect. 4) actually provide the top-k answers to the user query in input, avoiding to build useless solutions. The modules are based

on specific data, query and ranking models (briefly sketched in Sect. 2). An experimental evaluation comparing SocialGQ to existing approaches shows strong gains on a real social-graph data scenario (Sect. 5). Finally, Sect. 6 concludes the paper also by briefly analyzing related works.

## 2. Social-graph data, queries and answers

The aim of the SocialGQ data model is to have a flexible social-graph model (a) capturing key social-graphs’ features and (b) not bound to specific graph standards (e.g., RDF), even if easily supporting them. Data is generically represented as a connected multigraph (i.e., a graph with parallel edges) with node and edge labels. A social-graph essentially represents a portion of the real world through *entities* (concepts and their instances), *values*, and *relationships* between them. Considering our reference example (Fig. 1), scientific publications and annotations are described by concepts such as `BibTeX`, `Bookmark`, `Tag`;  $n_1, n_{11}, n_{13}$  and  $n_{14}$  are some instances. Instances are characterized by a *type* and a *userid*; the latter is represented in figure with small symbols in the upper right part and denotes users that created/modified them.

As to queries, in order to support complex requests such as those discussed in Sect. 1, we go beyond the keyword-based approach. A SocialGQ query is expressed as a labeled multigraph connecting entity nodes and conditions on values. Fig. 3 shows Q1, Q2 and Q3 expressed in our model. Note that users can annotate any node or edge with the wildcard “any label”, “#”. For instance,  $n_2, n_3$  and edge  $n_2-n_3$  in Q2 denote the user’s absence of knowledge about the specific nodes and edges connecting them. Conditions  $c$  can be defined on query nodes, specific cases characterizing a social-graph query are:

- the *full-text condition* (e.g., see  $n_3$  in Q1), supporting full text search in value nodes and returning a normalized TF-IDF score [13] *cs*. This allows SocialGQ to exploit the unstructured part of the social-graph;

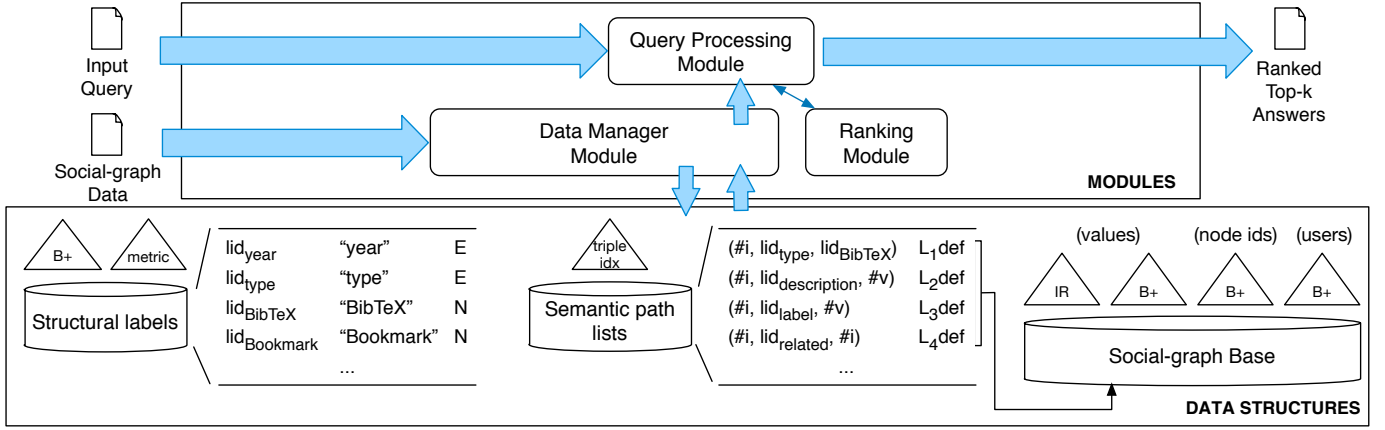


Figure 2. An overview of Social-GQ architecture

- the *user-aware condition* on query nodes (represented as +/- in figure), allowing users to restrict data matches to nodes not modified/known by them (-, as in Q1 for  $n_2$ ), or the opposite (+, as in Q3 for  $n_6$ , requesting tags related to “semantics” and defined by who submitted the query).

SocialGQ answers are portions of the data graph that *semantically approximate* the query. Two kinds of approximations are tackled: node/edge label and structural mismatch. To this regard, note that none of the sample queries finds an exact match on the reference graph. For instance, in Q1 (Fig. 3) the edge label *summary* is used instead of *abstract*, moreover no data edges are directly typed as *Content* (however, there are possible *BibTeX* and *Bookmark* matches, which are subclasses of *Content*). In SocialGQ, the degree of mismatch between labels is quantified by means of a user-definable semantic distance function  $d_L$  that, for any pair of labels, returns a value ranging from exact match (0) to total mismatch (1). As to structural mismatches, a purely topological approach which relaxes adjacency constraints by allowing arbitrary node/edge insertions in the data graph would not be able to produce meaningful answers. Instead, we consider meaningful sequences of consecutive edges (i.e., *paths*), named *semantic paths*, that match query edges with an approximation cost  $ac$ . For instance, the path  $n_2-n_1$  in Q1 (Fig. 3) could be approximated with the semantic path  $n_{13}-n_9-n_6$  (Fig. 1), which has the same “meaning”.

The goodness of each answer  $a$  to a query  $Q$  is quantified through a scoring function  $S$ :

$$S(a) = \alpha_n \cdot (1 - \text{avg}(d_L(n, \bar{n}))) + \alpha_e \cdot (1 - \text{avg}(d_L(e, \bar{e}) + ac(\bar{e}))) + \alpha_c \cdot \text{avg}(cs(c)), \quad (1)$$

where  $\alpha_n$ ,  $\alpha_e$  and  $\alpha_c$ ,  $\alpha_n + \alpha_e + \alpha_c = 1$ , are customizable coefficients (default=1/3) combining the average of: (a) label approximations ( $d_L$ ) occurring with each query node  $n$  and edge  $e$  ( $\bar{n}$  and  $\bar{e}$  denote matching data nodes/edges); (b) structural approximation costs ( $ac$ ) on each edge  $e$ ; (c)  $cs$  scores of each full-text query condition  $c$  (note that user-aware conditions prune out incompatible answers and do not affect ranking). The higher the returned score  $S(a)$ , in  $[0, 1]$ , the better the answer  $a$ .

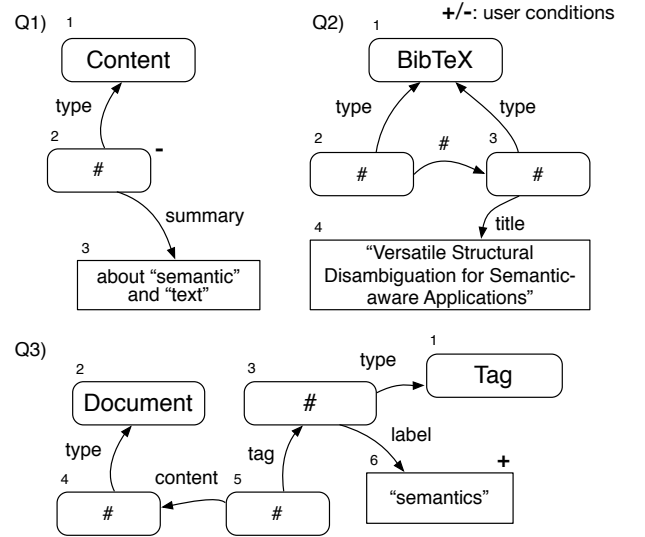


Figure 3. Three sample queries

More details on the data structures supporting SocialGQ query processing and how query processing itself is managed are presented in Sects. 3 and 4, respectively.

### 3 Data Manager

SocialGQ data manager organizes data in a “core” social-graph base (lower right part of Fig. 2), which is managed via the graph management system Neo4j [1], and in a series of additional ad-hoc auxiliary structures, which are the focus of this section. These provide advanced indexing and support for the innovative query features, ultimately helping the query processor in building the best answers as soon as possible.

First of all, the “Structural labels” table (lower left part of Fig. 2) stores structural (i.e., edges’ and entity nodes’) labels, associating the label to a short identifier (*lid*) and a kind (“E” for edges and “N” for nodes). Label data are indexed by means of B+ trees (allowing for exact search) and metric indexes (allowing for approximate search). As we will see, in query pro-

cessing this allows to quickly check if the structure of the given query is solvable on the social-graph data.

Besides dealing with labels, one of the most expensive operations for solving a query is to find the paths in the social-graph data which match the given query edges. To this end, the idea behind SocialGQ “semantic path lists” (lower central part of Fig. 2) is to organize repetitive data paths by means of identifying triples summarizing their structural role, ignoring specific instance and value information. Let us start the discussion by considering single edges. A single edge  $e$  connecting two nodes  $n$  and  $n'$  is straightforwardly identified by the involved label ids (i.e.,  $(lid_n, lid_e, lid_{n'})$ ). Value and instance nodes labels are generically represented with “#v” and “#i”, respectively. For instance, the `label` edges  $n_1-n_7$  and  $n_{11}-n_{15}$  (Fig. 1), connecting tag instances to their label value, are both associated to the triple  $(\#i, lid_{label}, \#v)$ . Triples are indexed and used in query processing to match with query edges (as we will see in next section). Associated to each triple, is a definition  $Ldef$  of a list pointing to the actual involved portions of the graph base (for instance, in our example, list  $L_{3def}$  points to the two above mentioned edges, among others). One interesting feature of SocialGQ is that such lists, which can be potentially quite large, do not need to be pre-computed or materialized. Being the core graph base managed in Neo4j, list definitions are (sets of) Cypher [1] scripts that can be easily customized by the data administrator. This flexibility is indeed particularly useful since such lists are not limited to identify relevant data edges, but also semantic paths. In this way, custom semantic rules capturing the specific meaning of the social-graph can be easily coded into the definitions so to allow meaningful structural approximations to be easily identified in the query processing phase.

Let us consider for instance the above discussed case of  $(\#i, lid_{label}, \#v)$  paths:  $L_{3def}$  can be defined to match single edges by means of the following Cypher MATCH clause:

```
(t:Tag) -[l:label]-> (v:Value)
```

but can also include a first level of approximation matching paths involving subtags (i.e., specializations of a tag label):

```
(t:Tag) <-[s:subtag]-n -[s:supertag]->
(t2:Tag) -[l:label]-> (v:Value)
```

In this way, for instance, Q3 edge  $n_3-n_6$  (Fig. 3) will match with data edge  $n_1-n_7$  (“semantics” label) but also with path  $n_{11}-n_3-n_1-n_7$  (“semantic web” label, a specialization).

Approximation levels (i.e., scripts) in list definitions are stored in order of increasing approximation cost  $ac$ ; the cost is automatically defined on the basis of the path length (in case, that can be freely customized). Other useful rules that could be easily incorporated in our social-graph example are for managing `type` paths (i.e., subclasses) or even for enriching the graph with useful relations not explicitly present in the data: for instance, a `related` relation between contents can be defined for contents sharing a common tag. This will match, for instance, with Q2 edge  $n_2-n_3$  (Fig. 3).

A number of indices on the graph base (lower right of Fig. 2) complete SocialGQ graph structures. These allow for efficient filtering of the semantic path lists on: (a) node values

(both B+trees for exact search and inverted indices for full-text search); (b) node ids (useful for joining edges in query processing); (c) users (for user-aware filtering). Such indices are directly managed in Neo4j, while inverted indices are externally coded in order to provide full TF-IDF score support.

## 4 Query processing and ranking

The goal of the query processor and ranking modules is to exploit the data structures made available by the data manager to generate the *top-k answers* approximating the query. Since social-graph are typically large and repetitive in their structure, a large number of approximate answers are typically available in the graph. Instead of generating the whole answer space and then applying the ranking formula (Eq.1), SocialGQ generates the *top-k answers* in an order that is already correlated with the ranking measure, avoiding to generate many useless results. The algorithm builds on the foundations of the Threshold Algorithm (TA) [3] and follows the steps summarized below:

1. search for (approximate) structural query node label matches in the “structural labels” tables, possibly pruning out unanswerable queries;
2. for each query edge, search associated triple in “semantic path lists” and associate the relevant list definition(s)  $Ldef$ ;
3. perform sorted access in parallel to each of the lists. For each access: (i) build answers involving the extracted data path, computing the score  $\mathcal{S}(a)$  of each answer  $a$ , and remembering it if one of the  $k$  highest; (ii) update  $uBound$ , the score of the set of the next data items under sorted access to the lists;
4. stop whenever at least  $k$  answers have been built whose grade is higher than  $uBound$ .

Please note that, as described in Sect. 3, each list  $Ldef$  returns data paths which are already sorted by approximation cost; this allows the algorithm to: (a) be aware of the goodness of upcoming answers by means of  $uBound$ ; (b) avoid unnecessary relaxations to the query (and, therefore, optimize data accesses).

To get a very simplified intuition of its working, consider Q1 in Fig. 3 submitted by user “U881”. Since the specified structural labels (i.e., `Content`, `type`, `summary`) are available in the graph base (`summary` is approximated by `abstract` and `description`), list definitions are retrieved for the two edges’ triples,  $(\#i, lid_{type}, lid_{Content})$  and  $(\#i, lid_{summary}, \#v)$ . Lists access is restricted to values matching the full-text condition (e.g., node  $n_{20}$ ) and instances not modified by user “U881” (e.g., node  $n_{14}$ ). The semantic paths extracted from the two lists are then accessed and joined to build such final answers as  $(n_{13}-n_{20})-(n_{13}-n_9-n_6)$ .

## 5. Experimental Evaluation

We will now present the preliminary results we obtained from an exploratory evaluation on a real social-graph data scenario by means of a first prototype of SocialGQ. This paper is

Query	#n	#e	#any	Description	Struct	(#	(mult	Label	Full	User	#exp
					relax	edge)	types)	appr	text	aware	
Q1	6	5	3	Users who tagged a BibTeX entry having author "Klaus Reuter"							1
Q2	7	7	4	Tags of BibTeX entries (-) having same year as BibTeX titled "Features of Similarity"						✓	48
Q3	6	5	3	Users who tagged a Content (-) having description about "Writing techniques"	✓		✓		✓	✓	2
Q4	5	4	3	Bookmarks (-) tagged "apple" (+)	✓					✓	18
Q5	4	4	3	Users (-) connected (# edge) to bookmark with URL "http://moodle.org/"	✓	✓				✓	1
Q6	5	4	3	BibTeX (-) connected (# edge) to BibTeX titled "Conceptual Knowledge Processing"	✓	✓				✓	17
Q7	6	5	3	Documents (-) tagged by profile with id 12	✓		✓	✓		✓	147
Q8	3	2	1	Content (-) having summary about "programming" and "database"	✓		✓	✓	✓	✓	23

Table 1. Features of the reference queries

focused on evaluating effectiveness, also in comparison to leading literature approaches. This requires a graph not particularly big in size but with a sufficiently complex structure, including different annotations and relation types. To this end, we consider as our reference collection a portion of the Bibsonomy graph dump [9], containing information about 139551 tag annotations, 28611 bookmarks, 11378 BibTeX entries, 8127 tags, 9566 tag specializations and 347 users. Moreover, a small hint regarding efficiency will be presented at the end of the section; in this case, we will also consider a larger graph involving over 1 million documents and 4 million annotations. The prototype incorporates the described advanced data indexes (including full-text ones) and query processing techniques, written ad-hoc in Python. It also exploits Neo4j graph management system, benefiting from node type management optimizations allowed by its built-in type management. User information is stored in Neo4j using node array properties. The chosen label distance  $d_L$  is a WordNet-based one we already used for disambiguation purposes [10]. All parameters are kept at their default values.

We consider a set of significant queries, named Q1-Q8, representative of a full-range of possible user information needs. Tab. 1 shows their features, including number of nodes, edges, "any label" wildcards, textual description, required features and number of expected answers. Q1 and Q2 are examples of exact queries which, even if not requiring special approximations, would be quite difficult to express using simple keywords. Queries Q3-Q8, instead, require several kinds of approximations; all queries except Q1 also require user-aware processing. For instance, Q4 asks for bookmarks not already modified/known by the user tagged as "apple": this should include, if available, documents tagged with subtags by the same user (e.g., "mac"). Structural relaxation is required to manage node subclasses (as for "content" in Q3 including types "BibTeX" and "bookmark"). Q5 and Q6 contain generic connection edges ('#'). Q7 and Q8 require label approximations (e.g., "document", "profile", "summary" used instead of "content", "user", "description"/"abstract", respectively). Finally, Q3 and Q8 also contain full-text conditions.

Tab. 2 shows the results of the effectiveness evaluation performed on Q1-Q8 in terms of: precision P (i.e. percentage of relevant retrieved answers w.r.t. the retrieved ones) and recall (i.e. percentage of relevant retrieved answers w.r.t. existing relevant ones). The results achieved by SocialGQ are also com-

Query	SocialGQ		Exact		Web #q	Non-semantic		NAGA		TALE	
	P	recall	P	recall		P	recall	P	recall	P	recall
Q1	1	1	1	1	5	0.0001	1	1	1	1	1
Q2	1	1	1	1	57	0.0000	1	1	1	1	1
Q3	1	1	n/a	n/a	n/a	0.0000	1	n/a	n/a	n/a	n/a
Q4	1	1	n/a	n/a	1	0.0000	1	1	1	0.07	1
Q5	1	1	n/a	n/a	n/a	0.0000	1	0	1	1	1
Q6	1	1	n/a	n/a	n/a	0.0000	1	0	1	0	1
Q7	1	1	n/a	n/a	n/a	0.0000	1	n/a	n/a	1	1
Q8	1	1	n/a	n/a	n/a	0.0000	1	n/a	n/a	n/a	n/a

Table 2. Effectiveness results and comparison

pared with the ones achievable through alternative approaches. Let us start with SocialGQ (left part of table): our approach is able to retrieve all results (recall is 1) for all queries, and all the retrieved results are relevant. This is achieved thanks to its combined semantic approximation, full-text and user-aware features. In particular, the high repetitiveness of the structure of a social-graph makes the employed structural and label approximations very effective and precise; label matches are also favored by, most notably, the relatively low number of different structural labels w.r.t. other scenarios and kinds of graph data (e.g., knowledge graphs). Moreover, the semantic path approach makes it easy to answer queries such as Q5 and Q6 without producing a large number of non-relevant results.

Other approaches, instead, typically struggle in supporting all the required features. For instance, an exact approach ("Exact" column) is only able to solve the first two queries. On the other hand, standard website search pages require a typically very high number of submitted queries ("Web" column) to solve a complex request. In our case, queries Q1, Q2 and Q4 can be partially answered through the Bibsonomy built-in search but: (a) this requires sending up to 57 simpler requests (Q2) and combining the results; (b) ranking and user-aware filtering are not supported anyway. We also see that the large number of nodes with same structural labels present in a social-graph makes it particularly infeasible to use syntactic-only approaches. A naïve approach ("Non-semantic" column) that computes node matches and connects them in all possible ways (e.g. a title to all available BibTeX nodes, disregarding semantic path information), achieves a near-null precision. Finally, we consider two flexible and well-known graph matching approaches presented in the literature. NAGA [8] is able to correctly deal with subclass approximations (e.g., Q4), however: it

manages ‘#’ edges in a syntactic way (very low precision n Q5 and Q6); it does not manage label approximation (Q7 and Q8 are not supported). TALE [14], instead, manages all structural approximations by defining a syntactic length threshold in the matching paths which proves not always effective: while Q5 is ok, in Q4 it retrieves results involving generalizations of “apple”, e.g., non-relevant “technology” documents. Please also note that neither NAGA or TALE natively support full-text and user-aware conditions.

As a final note regarding efficiency, for all the above queries the initial SocialGQ prototype was able to retrieve the top-5 results in under 0.04 secs (0.12 secs in the large collection) on a standard single-node configuration.

## 6. Concluding remarks

Successfully querying social-graphs requires to jointly exploit the complex nature of such kind of data: graph structure, semantic meaning, unstructured textual resources, user-aware dimension. As we have seen, this typically goes beyond the capabilities offered by the social sites’ search functions. In the literature, several works provide interesting, even if separate, results in each of the involved fields. In the area of (approximate) graph matching, many proposals [4, 8, 14] recognize the need to overcome the keyword-based paradigm and support early forms of approximation that, however, do not take into account semantics. [11] proposes a framework to support the semantic approximation of a complex query on a graph. None of these works, however, considers the specifics of social data and/or the contained unstructured contents. On the other hand, in the field of social data, a recent survey [2] analyses several search tool proposals, including [6, 7], all however limited to keyword-based search. Furthermore, it underlines the current sharp distinction between social search approaches (working only on the graph) and social web search ones (working only on unstructured content), noting the lack of approaches exploiting both components. Finally, a number of works [12, 15] highlight the effectiveness of identifying user-aware techniques that allow to customize the search results based on the user profile, but always in the context of unstructured data.

In this paper we laid the foundations of SocialGQ, a new proposal aimed to overcome the above mentioned limitations. Taking into account our past experiences in different scenarios (e.g., generic structured graph [11] and full-text enterprise search [12]), we sketched a framework combining for the first time all the features that are deemed essential for effective and efficient social-graph querying. The underlying techniques leverage on the strengths of approximate graph matching, semantic approximation, textual and user-aware retrieval to retrieve the most relevant answers first. Preliminary effectiveness results on a real data scenario are encouraging. Moreover, strong efficiency foundations have also been laid, being the proposed architecture based on extensions to widespread and reliable big data graph management technologies (i.e., Neo4j).

Making full use of the huge potential given by the ever-

increasing amount of social-graph data is indeed a very ambitious goal for the research community. Powerful social-graph search techniques can have a large impact and become the basis of a wide range of services integrated into leisure and business social networks: complex and personalized search tools to find products and information; intelligent help desk services to answer customer questions; tools to acquire a deep real-time knowledge on what is happening within the organization [5]. This work represents only one of the first steps toward this vision. In the future, we plan to: (a) work more deeply on efficiency evaluation; (b) perform detailed tests on additional and larger social-graph scenarios; (c) consider new techniques for automatic semantic path rules identification and refining.

This work is partially supported by UniMoRe within the FAR 2016 Department Project “SocialGQ”.

## References

- [1] Neo4j Graph Platform and Cypher language. <http://neo4j.com>.
- [2] M. R. Bouadjenek, H. Hacid, and M. Bouzeghoub. Social Networks and Information Retrieval, How Are They Converging? A Survey, a Taxonomy and an Analysis of Social Information Retrieval Approaches and Platforms. *Inf. Syst.*, 56(C):1–18, 2016.
- [3] R. Fagin, A. Lotem, and M. Naor. Optimal Aggregation Algorithms for Middleware. In *PODS*, pages 102–113, 2001.
- [4] W. Fan, J. Li, S. Ma, N. Tang, and Y. Wu. Adding Regular Expressions to Graph Reachability and Pattern Queries. In *Proc. of ICDE*, pages 39–50, 2011.
- [5] J. Hagel and S. K. Ellis. Four Ways Social Data Can Generate Business Value. <http://sloanreview.mit.edu/article/four-ways-social-data-can-generate-business-value/>.
- [6] X. He, M. Gao, M.-Y. Kan, Y. Liu, and K. Sugiyama. Predicting the popularity of web 2.0 items based on user comments. In *Proc. of SIGIR*, SIGIR ’14, pages 233–242, 2014.
- [7] D. Horowitz and S. D. Kamvar. The anatomy of a large-scale social search engine. In *Proc. of WWW*, pages 431–440, 2010.
- [8] G. Kasneci, F. Suchanek, G. Ifrim, M. Ramanath, and G. Weikum. NAGA: Searching and Ranking Knowledge. In *Proc. of ICDE*, pages 953–962, 2007.
- [9] KDE Group, University of Kassel. Benchmark Folksonomy Data from BibSonomy, 2017-07-01 dump. <https://www.kde.cs.uni-kassel.de/bibsonomy/dumps/>.
- [10] F. Mandreoli and R. Martoglia. Knowledge-based sense disambiguation (almost) for all structures. *Information Systems (Information)*, 36(2):406–430, 2011.
- [11] F. Mandreoli, R. Martoglia, and W. Penzo. Approximating expressive queries on graph-modeled data: The GeX approach. *Journal of Systems and Software*, 109:106–123, 2015.
- [12] R. Martoglia. AMBIT: semantic engine foundations for knowledge management in context-dependent applications. In *Proc. of SEKE*, pages 146–151, 2015.
- [13] G. Salton and C. Buckley. Term-Weighting Approaches in Automatic Text Retrieval. *Inf. Process. Manage.*, 24(5):513–523, 1988.
- [14] Y. Tian and J. Patel. TALE: A Tool for Approximate Large Graph Matching. In *Proc. of ICDE*, pages 962–973, 2008.
- [15] T. Vu, A. Willis, U. Kruschwitz, and D. Song. Personalised query suggestion for intranet search with temporal user profiling. In *Proceedings of CHIIR ’17*, pages 265–268. ACM, 2017.

# A Model-based Approach for Build Avoidance

Milena Neumann  
PTV Group  
Karlsruhe, Germany  
milena.neumann@ptvgroup.com

Kiana Busch  
Karlsruhe Institute of Technology  
Karlsruhe, Germany  
kiana.busch@kit.edu

Robert Heinrich  
Karlsruhe Institute of Technology  
Karlsruhe, Germany  
robert.heinrich@kit.edu

**Abstract**—In large software systems, we frequently encounter change scenarios which require long build times. In many cases, it would suffice to build only a subset of the dependent build components to generate sound build results. Current approaches for change-specific identification of affected build components rely on knowledge about the language-specific propagation of changes, which renders them inapplicable to multi-language systems. In this paper, we present a model-based approach to derive the affected build components for a change scenario using an existing change propagation approach. This way, we make the advantages of a set of change-specific dependencies also accessible to those members of the development team who are less knowledgeable about the build process. Our approach enables the use of change-specific dependencies in multi-language software systems and shortens build times. We implemented our approach in a productive build environment to show the feasibility and practicability in a user study.

**Index Terms**—software modeling, build automation, change propagation

## I. INTRODUCTION

Modern software engineering practices such as continuous integration use builds as fast feedback mechanisms to validate the quality of software [5]. Build systems usually face two requirements: i) Building should be fast. ii) The build process should yield reliable and reproducible results. Existing build tools often allow structuring a system into build components and defining a dependency graph on them. This enables developers to run partial builds (e.g., the modified build component and all its dependents). However, the structural dependencies defined by the dependency graph can differ significantly from actual dependencies with regard to the nature of a change.

A small change to a large software system may cause long build times, when considering only a subset of the dependent build components may be sufficient to produce sound build results. Making use of this knowledge and building only the affected build components is referred to hereafter as a build shortcut. Building only those components which are affected by a specific change can save a lot of time. However, in large development teams, there is often only a small number of developers who can identify the build components that are actually affected by a change. As the evolution of the product affects the dependency graph, the change-specific dependencies are also subject to change, but keeping all team members up to date is a time-consuming task.

Existing build tools, like grexmk [1] or vroom [3], speed up build times by executing build tasks in parallel, but come at the cost of the employed hardware. Approaches such as jmake for Java [13] allow language-specific change propagation analysis. However, they can only be applied to a specific set of software projects. Approaches like pluto [6] offer extensive customization of build tasks, but at high integration costs.

In this paper we propose an approach which enables developers less-experienced with build dependencies to selectively build those components that are affected by a specific change. To achieve this, our approach utilizes build shortcuts. The build experts of a development team annotate their knowledge on change-specific dependencies in the model of the software's build architecture. This model and the change scenarios are input for an existing change propagation approach for software architectures – Karlsruhe Architecture Maintainability Prediction (KAMP) [12]. The change propagation algorithm of KAMP identifies the build components to be built in order to implement the change. This way, we avoid rebuilding unaffected build components and consequently shorten the build times. Our model-based approach allows the formulation of build shortcuts for multi-language projects. Due to its simple interface, it can be used to extend an existing build system with little effort. The content of this paper has been developed in the context of the thesis *KAMP for Build Avoidance on Generation of Documentation* [9]. We evaluated our approach in a productive build environment by means of a user study. Our approach produced builds that were up to 27 times faster than building with the established build tool. The subjects of the user study assessed the approach as practicable.

The following section illustrates the PTV xServer, where our approach was applied. The foundations are described in Section III. Section IV gives an overview of the related work. We present our approach for build avoidance in Section V. In Section VII, we describe the results of our evaluation. This paper is concluded with a summary and an outlook on future work in Section VIII.

## II. THE PTV xSERVER

We use the PTV xServer – a product of PTV Group – as a running example in this paper. The PTV xServer provides logistic and geographic solutions such as geocoding and trip-planning. PTV xServer is currently being developed by thirty developers from Germany and France.

The Application Programming Interface (API) of the server is described in so-called XServer Interface Description Language (XSIDL) files, which are used for the generation of C++ and Java source files. Furthermore, the comments in the XSIDL files are used in front-end components of the xServer to generate the product documentation for the customers. To build the xServer, an in-house build tool (the so-called b-Tool) is used, which is based on Apache Maven<sup>1</sup>. Apache Maven utilizes a dependency graph to identify the affected build components (called Maven projects) by a change. According to those dependencies, a change to one of the XSIDL files (e.g., a change the comments) requires a large part of the product to rebuild. Currently, this build process takes around four hours on average. However, rebuilding just the affected front-end components would only take a few minutes.

To avoid rebuilding unaffected build components, we use KAMP to analyze the scope of the change propagation in the PTV xServer. For this purpose, we modeled seven build shortcuts to consider various change scenarios. Furthermore, we extended the b-Tool by a new build command, called `shortcut`.

### III. FOUNDATIONS

Our approach is based on KAMP to identify the relevant build components affected by a specific change request. The KAMP approach aims to "analyse the change propagation caused by a change request in a software system based on the architecture model" [12]. For this purpose, KAMP uses a change propagation algorithm, which is validated through an empirical study [12]. Starting with the initial change request, the change propagation algorithm calculates the affected elements in the architecture model.

To model the architecture of a component-based software system, a Palladio Component Model (PCM) [10] can be used. In a PCM, the parts of a software system such as interfaces, components and their relations are modeled [10].

### IV. STATE OF THE ART

The build tool grexmk [1] splits monolithic builds into multiple sand-boxed "mini-builds", taking advantage of loose coupling between system components. By executing these mini-builds incrementally or simultaneously, overall build times are shortened. The Vroom approach [3] analyzes the dependencies of test cases and carries out long-running tests in parallel. Although parallelization shows immediate results and is scalable by increasing hardware resources, it comes at the cost of this additional hardware. Instead of reducing time through parallelization, our approach saves time by not building unaffected components in the first place. Subsequently, the workload on build machines is lowered.

The build system pluto "supports the definition of reusable, parameterized, interconnected builders" [6]. As it abstracts from the programming language, pluto is applicable to a wide variety of projects. It can extract specific information from

a file through domain-specific stampers. These file stampers can be used to implement change-specific rebuild behavior. But especially for large-scale projects, pluto would require a lot of implementation work, which also brings in additional risks. Our approach can coexist with a preexisting build environment by extending it, without introducing potentially breaking changes.

Smith describes the concept of smart dependencies [13]. This technique analyzes whether dependents need to be rebuilt by applying knowledge of the language-specific change propagation. The tool jmake [13, 8] applies smart dependencies for Java. If it detects modifications to publicly visible parts (e.g., method signatures), it rebuilds the dependent files. The addition of a new method or changes to comments, private methods, or the code within a method only requires a rebuild of the modified file itself. A build tool based on smart dependencies is language-specific, therefore it cannot be applied to multi-language projects. Our approach views dependencies between build artifacts on a more abstract level and is language agnostic. This comes with the disadvantage that shortcuts for specific change scenarios have to be defined manually and cannot be automatically derived.

### V. APPROACH FOR BUILD AVOIDANCE

Our approach allows the developers of a software system to build only the subset of those build components which are affected by a change. Figure 1 compares our build approach using build shortcuts (right) and the build process by Apache Maven (left). Apache Maven rebuilds the whole subgraph of the dependent build components, while our approach allows rebuilding only a specific subset of this subgraph.

The input of our approach is a PCM of the software system's build architecture, which contains the build shortcuts and the initial change requests. In the next step, KAMP is used to identify the build components which are potentially affected by the initial change requests. The result is a set of potentially affected build components. Deploying KAMP as a web service allows a loosely coupled build tool to access the build information. This way, only the model in the web service needs to be updated in order to change a dependency or add a new build shortcut. In the following subsections, we discuss our approach in more detail.

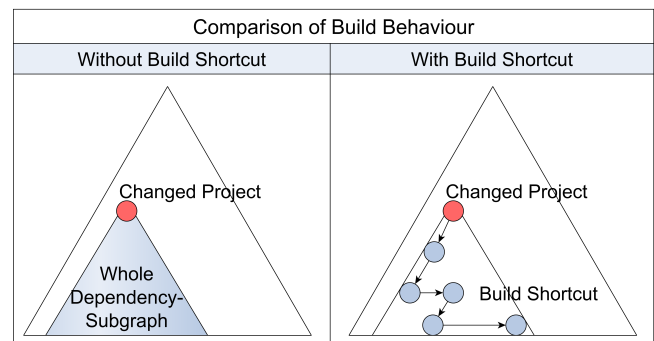


Fig. 1. Comparison of build behavior without and with build shortcut [9]

<sup>1</sup><http://maven.apache.org>



### A. Modeling of Build Shortcuts using Palladio Component Model

The dependency of a build component may be change-specific. Translated into a PCM, a component can provide several interfaces, which are required by different subsets of its dependent components. If a change to one of these interfaces is made, only the components that require this interface are potentially affected by the change (i.e., not all components that require interfaces of a specific component) [12].

Figure 2 shows a simplified example from the PTV xServer. The change scenarios of the component `model` are represented by the two interfaces `model` and `documentation`. The `model` and `documentation` interfaces are required by `frontend` component. The `model` interface is required by `services` and `runtime` components. Let us assume that `documentation` interface is changed. In this case, only the `frontend` component is affected by the change, and will not affect `services` and `runtime` components.

Our approach uses a PCM to model the build architecture, which can differ from the software architecture. Originally, PCM was designed to model component-based software architectures. However, it can also be used to model the build architecture, which describes relations between the build components, thus providing a more technical view on the software system.

A build component can contain several software components, as illustrated in Figure 2. Although the build component may provide more than one functionality, it cannot always be divided into subcomponents in accordance to the change scenarios. In the example of the `model` component in the PTV xServer, the separation of the `documentation` from the API `model` may cause the developers to neglect keeping the `documentation` up to date.

### B. Utilization of KAMP's Change Propagation Analysis

As described above, the build architecture can differ from the software architecture. The Apache Maven projects can be mapped to components in PCM, while different types of change scenarios for each Maven project represent different interfaces. Based on an initial change request, KAMP uses its change propagation algorithm to identify the affected build components [12]. KAMP's change propagation algorithm consists of a set of change propagation rules [12]. An example of a

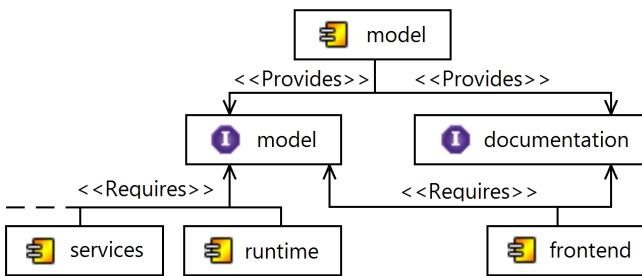


Fig. 2. Modeling of a build shortcut with PCM [9]

rule is the change propagation from a modified interface to all components that provide or require this interface [12]. Applied to the build architecture, this method allows identifying the affected build components. By utilizing the domain knowledge annotated in the PCM, KAMP can reduce the set of build components that are rebuilt.

## VI. PROTOTYPICAL IMPLEMENTATION

Figure 3 illustrates the deployment of our approach at PTV Group. The components of our approach run on a server (i.e., the server side) in the PTV intranet and on the individual developer workstations (i.e., the client side). On the server side, the `KAMP-WS Server` is deployed together with the PCM of the software under development (i.e. the PTV xServer PCM) and `KAMP`. On the client side, the build tool (i.e., PTV b-Tool) is deployed. The PTV b-Tool is extended by the command `shortcut`, which uses the web service.

When a build with the `shortcut` command is triggered, the available change scenarios for the modified Maven project are requested from the KAMP web service. This list is presented to the user, who selects the appropriate scenario over the command line interface. Then, a second request is sent to the KAMP web service querying the dependent projects for the selected change scenario. After `shortcut` receives the response, it triggers an Apache Maven build of specifically the dependent projects.

## VII. EVALUATION

To evaluate our approach, we follow the Goal Question Metric (GQM) [2] plan. We define two evaluation goals:

- **Feasibility** aims to "validate the prediction accuracy [...] by comparing the prediction results to reference values" [7]. It is assumed that inputs are correct.
- **Practicability** aims to "validate the practicability of a method, when it is applied by target users, instead of method developers" [7].

In the following, we define research questions (RQ) for each evaluation goal. Furthermore, we propose the metrics to evaluate the corresponding evaluation goals and research questions.

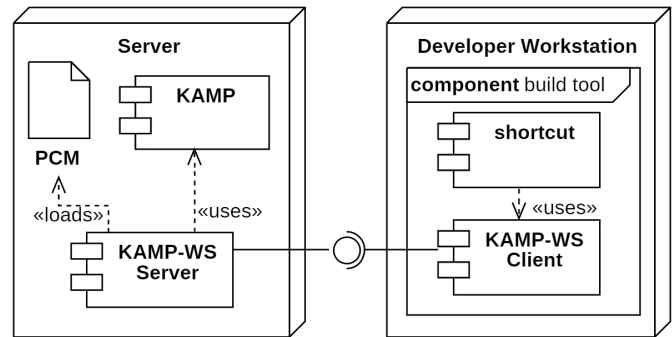


Fig. 3. Deployment diagram of the system [9]

### A. Feasibility

To evaluate the feasibility of our approach, we compare the actual outputs of our approach to expected results. The build experts provided reference lists for a total of seven build shortcuts, each containing the build components that have to be actually built to validate a change scenario. We annotated the respective build shortcuts in the xServer PCM used by the web service. Thus, we need to check the equality of the two sets for each change scenario: the set of Maven projects proposed by our approach (i.e.,  $P_{\text{shortcut}}$ ), and the set of Maven projects in the corresponding reference list (i.e.,  $P_{\text{ref}}$ ). In other words, the following formula should be true for each of the seven change scenarios:

$$P_{\text{shortcut}} = P_{\text{ref}} \Leftrightarrow P_{\text{ref}} \subseteq P_{\text{shortcut}} \wedge P_{\text{shortcut}} \subseteq P_{\text{ref}}$$

This leads to the following research questions and hypotheses:

**RQ 1:** Does the output of our approach include all build components that have to be built for the respective change scenario?

Hypothesis **H1** is that the output of our approach includes all build components that have to be built according to the reference lists (i.e.,  $P_{\text{ref}} \subseteq P_{\text{shortcut}}$ ). In other words, we assume that the build components that are necessary to validate the change scenario are included.

**RQ 2:** Does the output of our approach not include build components that are unaffected by the respective change scenario?

Hypothesis **H2** is that the output of our approach does not include build components that are not on the reference list of the change scenario (i.e.,  $P_{\text{shortcut}} \subseteq P_{\text{ref}}$ ).

We designed a set of system tests to gather metrics to verify hypotheses **H1** and **H2**. Two types of tests exist for each change scenario. The first test checks whether our approach builds all build components specified in the reference lists. The second test checks whether the build does not include an unaffected build component. If both tests are successful, the list of build components that are built with our approach for a given change scenario is the same as the corresponding reference list.

As described above, we implemented our approach as a new build command `shortcut` and added it to PTV's b-Tool. We want to verify that building with the `shortcut` command takes less time than building with `down`, which is an established b-Tool command of the xServer development team and builds a build component and its dependents.

**RQ 3:** Is the duration of a build process triggered with the `shortcut` command shorter than the duration of a comparable build process triggered with the `down` command?

Hypothesis **H3** is that the duration of a build triggered with `shortcut` is shorter than the duration of a comparable build triggered with `down`.

To answer this research question we define the execution time metric for both `shortcut` and `down` commands and

compare the build execution times measured by developers on their workstations, as described in the following.

### B. Practicability

Our evaluation of practicability is based on the Technology Acceptance Model (TAM) [4]. According to TAM, the intentions of a person to use a technology determines the actual use [4, 9]. For each of TAM's variables, a research question and a hypothesis was formulated, taken from [9]:

**RQ 4:** Do the subjects find the activities provided by our approach (i.e., `shortcut` command) important?

Hypothesis **H4** is that the subjects find the activity provided by our approach (i.e., building only the affected Maven projects in a change scenario) important.

**RQ 5:** Does our approach ease performing the activities?

Hypothesis **H5** is that our approach eases performing the activities (i.e., Perceived Usefulness [4]).

**RQ 6:** Is our approach easy to use in practice?

Hypothesis **H6** is that our approach is easy to use in practice (i.e., Perceived Ease of Use [4]).

**RQ 7:** Do the subjects have a specific intention to use our approach?

Hypothesis **H7** is that the subjects have a concrete intention to use our approach (i.e., Behavioral Intention to Use [4]).

**RQ 8:** Do the subjects expect concrete consequences by using our approach?

Hypothesis **H8** is that the subjects expect concrete consequences by using our approach (i.e., Attitude Toward Using [4]).

We conducted a user study to evaluate the previously described research questions and validate the respective hypotheses. The subjects of the study were the potential users of our approach at PTV. They evaluated the usage and performance of the `shortcut` command in comparison to the `down` command.

### C. User Study Design

The user study consisted of a task sheet and a questionnaire. In the task sheet, the subjects were asked to run these three different builds on the `model` Maven project (cf. Section II and Figure 2):

- Trigger the build process using `down` command (hereafter referred to as `down`).
- Trigger the build process using `shortcut` with the model change scenario (hereafter referred to as `shortcutmodel`).
- Trigger the build process using `shortcut` with the documentation change scenario (hereafter referred to as `shortcutdoc`).

We chose these build scenarios because most members of the development team are familiar with this build component and have worked on it before. Thus, the change scenarios were well understood by the subjects.

The subjects were asked to provide the resulting build times in the questionnaire. Furthermore, the subjects provided information about their usual build behavior and how they evaluate our approach.

The questionnaires contained a set of free text questions. Also, the subjects had to rate the following statements, taken from [9], on a 6-point Likert scale, where 1 means "I strongly disagree" and 6 means "I strongly agree":

- S1 I frequently validate changes through local builds.
- S2 Building (parts of) the xServer is important for my work.
- S3 I could work more efficiently if local builds were faster.
- S4 Local builds take too long.
- S5 I try to only build the projects that were affected by my change.
- S6 `shortcut` command will be useful to me.
- S7 `shortcut` command is easy to use.
- S8 I am motivated to use `shortcut` command for my local builds in the future.

We aim to assess the importance of the build process to the subjects with statements **S1** and **S2**. Statements **S3** – **S5** are included to verify that the subjects are dissatisfied with the current build times. The remaining statements **S6** – **S8** aim to evaluate **RQ 5** – **RQ 7**.

The xServer team consists of 30 Java and C++ developers, of which 18 participated in our study. Note that the C++ parts of the xServer have longer build times than the Java parts. An evaluation of the answers is presented in the following.

#### D. Results

**System Test Results:** The system tests for each of the seven build shortcuts were successful. In other words, the statements  $P_{ref} \subseteq P_{shortcut}$  and  $P_{shortcut} \subseteq P_{ref}$  hold true for each shortcut. That confirms **H1** and **H2**. Thus, we can conclude  $P_{shortcut} = P_{ref}$ .

**User Study Builds:** The subjects were asked to provide the build times for the builds they had to trigger. Table I gives an overview of the build times. The *down* build scenario took longest (i.e., from about 1.5 hours to almost 9 hours). The builds of the *shortcut<sub>model</sub>* scenario took between 30 minutes and almost 2.5 hours, while the builds of *shortcut<sub>doc</sub>* took only 3 to 12.5 minutes. On average, the builds of *shortcut<sub>model</sub>* were 3.45 times faster than the builds of *down*, while the builds of the *shortcut<sub>doc</sub>* scenario were 27.07 times faster than *down*.

The significant difference in build times for a single build scenario can be accounted to different hardware of the developer workstations and network latency, as some subjects used a VPN connection from France. The build execution times of

TABLE I  
EVALUATED BUILD TIMES FOR THE SCENARIOS IN THE USER STUDY [9]

	<i>down</i>		<i>shortcut<sub>model</sub></i>		<i>shortcut<sub>doc</sub></i>	
Max	8h	55min	2h	27min	12min	30s
Avg	3h	42min	1h	1min	30s	9min
Min	1h	32min		30min	6s	2min

TABLE II  
DETAILED RATINGS FOR STATEMENTS S1-S8 IN THE USER STUDY [9]

	1: Strongly disagree	2: Disagree	3: Rather disagree	4: Rather agree	5: Agree	6: Strongly agree
S1:	0	0	0	0	2	17
S2:	0	0	0	0	3	16
S3:	0	0	0	2	4	13
S4:	0	0	1	2	7	9
S5:	0	0	0	1	5	13
S6:	0	0	2	1	3	13
S7:	0	0	1	1	4	13
S8:	0	1	1	1	3	13

the *shortcut<sub>model</sub>* and *shortcut<sub>doc</sub>* scenarios are lower than the build time of *down*. This confirms **H3**.

**Activity Importance:** Statements **S1** and **S2** aim to assess the importance of builds to the subjects, as illustrated in Table II. All subjects rated statements **S1** and **S2** with 5 (i.e., agree) or 6 (i.e., strongly agree), which confirms **H4**.

**Satisfaction with Builds:** Table II shows how the subjects rated statements **S3**–**S5**. Those subjects who rated statement **S4** with 4 or lower contribute mainly to Java parts of the xServer, which have lower build times in general. We conclude that developers perceive long builds as a problem and they try to shorten them.

**Usefulness:** Statement **S6** aims to assess the perceived usefulness of our approach. As seen in Table II, most subjects rate our approach as useful. This confirms **H5**. Only two Java developers specified that they rather disagree with the statement. One of them noted that they work on a partial checkout of the xServer repository. They could not use our approach, because the utility functions used by the implementation of `shortcut` work under the assumption of a full checkout. We did not consider this requirement (i.e., working on partial checkouts) during development of our approach, but it could be supported in the future.

**Ease of Use:** Most subjects found our approach easy to use (i.e., statement **S7**), as seen in Table II. This confirms **H6**. Eight subjects additionally mentioned in the free text fields that they found our approach easy and intuitive to use. One subject rated statement **S7** with 3 (i.e., rather disagree), however, gave no further comment in the free text fields.

**Intention To Use:** Most subjects rated Statement **S8** (i.e., using our approach `shortcut` for the local builds in the future) with 4 or higher. That confirms **H7**. One subject, who works on the Java parts of the xServer, rather disagreed with

this statement. Also, the subject who stated that they work on partial checkouts disagreed with the statement.

**Attitude Toward Using:** The questionnaire contained two free-text questions regarding the advantages and disadvantages of our approach. We can derive from the given answers which consequences the subjects expect by using our approach. The following advantages were expected by the subjects, taken from [9] (The number in parentheses states how often it was mentioned by the subjects):

- Faster builds (10)
- Fewer unnecessary builds (6)
- Increased work efficiency (3)
- Fewer failed builds (2)
- Less knowledge about dependencies required (2)
- Conveniently trigger subtarget builds (2)
- Faster validation of code / Earlier detection of defects (2)
- Sharing of useful build shortcuts
- No more manual triggering of different artifacts

The following disadvantages were expected by the subjects, taken from [9]:

- Maintenance cost of build shortcuts (7)
- Shortcut names may become confusing (2)
- Far-reaching consequences if a shortcut is incorrect (2)
- No batch-compatibility
- Less pressure to solve dependency problems
- Less pressure to optimize build times
- Developer’s knowledge about dependencies decreases

Each subject stated at least one expected consequence. This confirms **H8** (i.e., subjects expect concrete consequences by using `shortcut`). The number of advantages mentioned is higher than the disadvantages. In other words, subjects expected predominantly positive consequences.

By the results of our user study, we see both the feasibility and the practicability of our approach confirmed.

### VIII. CONCLUSION AND FUTURE WORK

In this paper, we presented an approach to shorten the build times, based on build shortcuts. It makes the advantages of build shortcuts also accessible to those members of the development team who are less knowledgeable about the build process.

The change-specific dependencies are modeled in a PCM. The model of the build architecture and the change requests are used as input for the KAMP approach. The KAMP approach uses a change propagation algorithm to identify the affected build components for a given change scenario.

We evaluated our approach using a set of system tests and a user study. The system tests, which checked whether the builds produced by our approach are conform to the reference lists created by build experts, were all successful.

In the user study, 18 developers were asked to compare the results of three different build scenarios. The execution time of the build process for the model scenario was on average about 3.5 times faster than the reference build with the `down` command, while the documentation build was 27 times faster.

In the questionnaires of the study, the subjects indicated that they found the new command useful and have the intention to use it in the future. We see both feasibility and practicability of our approach confirmed through the results of the evaluation.

The KAMP web service continues to be used and maintained in the PTV xServer project. Much of the feedback received in the user study was already incorporated (e.g., the extension of `shortcuts` API to support batch-compatibility). Furthermore, additional shortcuts were added to the model.

As future work, we plan to extend the KAMP web service to assist development teams with other tasks such as analyzing the change propagation of backlog items to coordinate the work of multiple scrum teams in a scrum of scrums.

### ACKNOWLEDGMENT

This work was partially supported by the DFG (German Research Foundation) under the Priority Programme SPP1593 (RE1674/12-1).

### REFERENCES

- [1] Glenn Ammons. “Grexmk: speeding up scripted builds”. In: *International workshop on Dynamic systems analysis*. ACM, 2006, pp. 81–87.
- [2] Victor Basili et al. “The Goal Question Metric Approach”. In: *Encyclopedia of Software Engineering 2.1994* (1994), pp. 528–532.
- [3] Jonathan Bell et al. “Vroom: Faster Build Processes for Java”. In: *IEEE Software* 32.2 (2015), pp. 97–104.
- [4] Fred Davis et al. “User acceptance of computer technology: a comparison of two theoretical models”. In: *Management science* 35.8 (1989), pp. 982–1003.
- [5] Paul Duvall, Stephen Matyas, and Andrew Glover. *Continuous Integration: Improving Software Quality and Reducing Risk*. Addison-Wesley, 2007.
- [6] Sebastian Erdweg et al. “A sound and optimal incremental build system with dynamic dependencies”. In: *SIGPLAN Notices*. Vol. 50. ACM, 2015, pp. 89–106.
- [7] Robert Heinrich. *Aligning Business Processes and Information Systems: New Approaches to Continuous Quality Engineering*. Springer, 2014.
- [8] JMake. *JMake*. <https://github.com/pantsbuild/jmake>. [Online; accessed April 2018]. 2014.
- [9] Milena Neumann. “KAMP for Build Avoidance on Generation of Documentation”. Bachelor’s thesis. Karlsruhe Institute of Technology, 2017.
- [10] Ralf Reussner et al. *Modeling and simulating software architectures: the Palladio approach*. MIT Press, 2016.
- [11] Kiana Rostami et al. “Architecture-based Change Impact Analysis in Information Systems and Business Processes”. In: *ICSA2017*. IEEE, 2017, pp. 179–188.
- [12] Kiana Rostami et al. “Architecture-based Assessment and Planning of Change Requests”. In: *11th International ACM SIGSOFT Conference on Quality of Software Architectures*. ACM, 2015, pp. 21–30.
- [13] Peter Smith. *Software Build Systems: Principles and Experience*. First. Addison-Wesley Professional, 2011.

# Conceptual Software: The Theory Behind Agile-Design-Rules

Iaakov Exman

Software Engineering Department  
The Jerusalem College of Engineering – JCE - Azrieli  
Jerusalem, Israel  
iaakov@jce.ac.il

**Abstract**—Software practices often evolved sets of efficient software design rules embodying together a kind of methodology. However, methodologies per se are no substitute for a rigorous software theory. Methodologies can live side by side with a software theory which explains and justifies the widely accepted wisdom of the field. This paper shows that Linear Software Models, an algebraic Software Theory together with its basis, the Conceptual Integrity principles, indeed explain the deeper contents of the so-called “four rules of simple design”, which we concisely name as Agile-Design-Rules. These rules are a succinct expression of agile design methodologies that emerged from Extreme Programming (XP). Thus one obtains the best benefits from Software Theory and methodology co-existence: 1<sup>st</sup>, the explained rules reinforce the Software Theory plausibility; 2<sup>nd</sup>, the Software Theory selectively clarifies roles of the Agile-Design-Rules enabling quantitative calculations for their application in practice; 3<sup>rd</sup>, co-existence leads to the idea of *Design Tests* as illustrated by case studies.

**Keywords:** *Conceptual Software; algebraic Software Theory; Agile-Design-Rules; Software Design; Linear Software Models; Modularity Matrix; Modularity Lattice; Conceptual Integrity; Propriety; Orthogonality.*

## I. INTRODUCTION

A well-known set of rules for software system design is the so-called “four rules of simple design” which we concisely name as Agile-Design-Rules. These rules were first formulated by Kent Beck (see page 57 in [2]) at the end of the previous century, within the context of Extreme Programming (see e.g. [3]), commonly abbreviated as XP. This well-known kind of agile design methodology had a significant influence on approaches to practical software system development.

Some outstanding XP development characteristics [3] are:

- *Simple Design* – expressed, e.g. by the Agile-Design-Rules;
- *Tests* – software development is driven by tests written and run in parallel to the software itself.
- *Pair Programming* – production code is often written by two people at one screen/keyboard/mouse.

We claim, that however successful in practice, any development methodology such as XP is no substitute for a rigorous Software Theory. On the other hand, a theory totally disconnected from practical directives in a field eminently application-oriented such as Software Engineering is of no use.

This paper has two-way goals: a- to argue that the algebraic Software Theory is a rigorous basis for applicable software design methodologies; b- to show that the Agile-Design-Rules essence is selectively explained and justified by the referred Software Theory. We introduce the Agile-Design-Rules, the basics of Conceptual Software design, and the algebraic Linear Software Models.

### A. Agile-Design-Rules for Software Development

There are several formulations of the Agile-Design-Rules, differing by the wording of each rule and the rules' order. The rules' essence is common to all formulations. Here, we choose a formulation by Ron Jeffries [25], reordering rules 2 and 3:

1. ***Test Everything*** – All the tests for the SUD (Software Under Development) are passing;
2. ***Explicit Intent*** – Express the ideas the software's author wants to express;
3. ***Eliminate Duplication*** – Contain no duplicate code;
4. ***Minimize Entities*** – Minimize classes and methods.

The selective interpretation of these rules will be given later on in this paper, after the theory basics are explained.

### B. Conceptual Integrity

Conceptual Integrity is a deep software design idea proposed by Frederick Brooks [6], [7], much earlier than the Agile-Design-Rules. Historically, the earlier ideas were not recognized as the basis for the practical rules.

Three principles suggested by Brooks [6] were verbally, not formally, explained by Jackson et al. [10], [23], [24]. We focus on the two most relevant to clarify Agile-Design-Rules:

1. ***Orthogonality*** – individual functions should be independent of one another;
2. ***Propriety*** – a product should have just the functions essential to its purpose and no more;

The Conceptual Integrity principles can be expressed in terms of modularity and design simplicity. Orthogonality is a basic modularity mechanism. Propriety is an optimization: the fewer the functions performing *exactly* the same tasks, the simpler the software product. One immediately perceives their relevance to the Agile-Design-Rules. Full interpretation will follow from the theory, presented next.

### C. Linear Software Models: the Modularity Matrix and the Modularity Lattice

We concisely characterize two Linear Software Models' algebraic structures, representing a software system. These are the Modularity Matrix and the equivalent Modularity Lattice. Here we explain the primitive terms of these models relevant to this work; for further details, see [11], [12].

Software systems are assumed to be hierarchical. *Structors*, the Modularity Matrix columns, are vectorial expressions of software structure, generalizing classes for any hierarchical level. *Functionals*, the Modularity Matrix rows, are vectorial expressions of software behavior, generalizing functions which are provided by structors. The standard Modularity Matrix is block-diagonal. *Modules*, illustrated in Fig. 1, are sub-system blocks along the Modularity Matrix diagonal, made of structor and functional sub-sets, disjoint to other module sub-sets.

As shown by Exman and Katz [18], Modularity Matrix design optimization neatly corresponds to Conceptual Integrity principles. Propriety justifies Linear Independence of structors among themselves and functionals among themselves. Orthogonality implies the existence of modules.

The Modularity Lattice [14] is obtainable from the Modularity Matrix, by well-known algorithms embodied in software tools (e.g. Concept Explorer) building the lattice from a given *Formal Context*. This Context is a rectangular Boolean matrix showing relations between a set of attributes and a set of objects. The standard Modularity Matrix can be seen as a special case of Context: it is square, with relations respectively between structors and functionals. A fitting Modularity Lattice is obtained (Fig. 2 corresponds to Fig. 1), in which the Top node contains the set of all functionals and the Bottom node contains the set of all structors. Modularity Lattice modules (see Exman and Speicher [14]) are the connected components, obtained by erasing the Top and Bottom nodes, which represent the whole system, and not specific modules.

The goal of Linear Software Models is to reach the standard Modularity Matrix for a software system, where all blocks are orthogonal. An outlier matrix element coupling modules, not orthogonal anymore, demands a system redesign. Due to the Modularity Lattice to the Modularity Matrix equivalence, all conclusions extracted from the Matrix are valid for the Lattice.

#### Paper Organization

The remaining of the paper is organized as follows. Section II describes related work. Section III concerns the intent of Conceptual Software Design. The central section IV formulates the quantitative algebraic software theory of Agile-Design-Rules. Section V illustrates Design Tests with case studies. Section VI concludes the paper with an overall discussion.

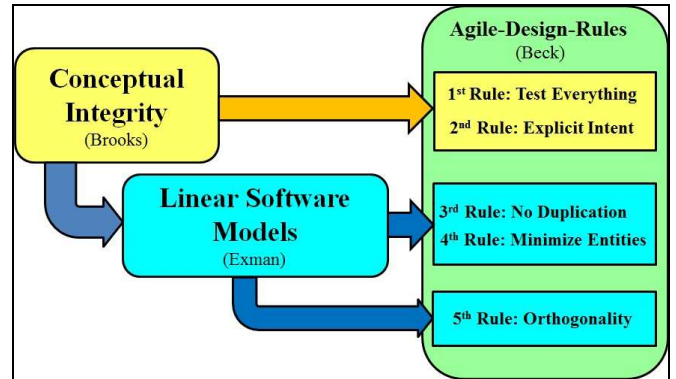


Figure 1. Software Theory explains Agile-Design-Rules – “*Conceptual Integrity*” directly explains the first two rules and is a conceptual basis for the algebraic “*Linear Software Models*”. These Models directly explain the three other Agile-Design-Rules. This diagram motivates the paper organization.

		Structors			
		S1	S2	S3	S4
Functionals	F1	1			
	F2		1		
	F3			1	1
	F4			0	1

Figure 2. An Abstract Standard Modularity Matrix – It has 4 Structors (matrix columns) and 4 Functionals (matrix rows). Three block-diagonal modules are seen (in blue): two strictly diagonal (S1, F1) and (S2, F2), and one 2\*2 block (S3, S4, F3, F4). Matrix elements outside the modules (in white) have zero values (omitted for easier visualization). F3, an example of functional inheritance, is provided by both classes S3 and S4.

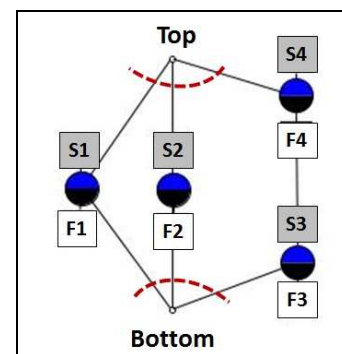


Figure 3. Abstract Modularity Lattice diagram – This Lattice exactly fits the Modularity Matrix in Fig. 1. It has three modules: two with just one-vertex (S1, F1) and (S2, F2), and one with two vertices (S3, S4, F3, F4). Structor labels are shown above and functional labels below the vertices. A vertex above another one, provides all the functionals below the higher one: e.g. S4 provides both F4 and F3, as also seen in the matrix in Fig. 1, while S3 provides only F3. Modules are the connected components remaining after cutting Top and Bottom nodes, as shown by the red dashed lines.

## II. RELATED WORK

### A. Agile-Design-Rules for Software Design

Agile-Design-Rules were formulated by Kent Beck, in the first 1999 edition of his book “*Extreme Programming Explained: Embrace Change*” [2]. They were reformulated by Beck himself [3] and by several authors in different rules’ order and specific wording. Martin Fowler collaborated with Kent Beck, writing together the “*Planning Extreme Programming*” book [4], and later on wrote a blog entitled *Beck-Design-Rules* [19] with his own version of the Rules.

Corey Haines published a whole book entitled “*Understanding the 4 Rules of Simple Design*” [21] using the *Game of Life* to illustrate the rules. Several other variations of the Agile-Design-Rules are due to Bekkers [5], Rainsberger [30] and Sironi [31], among others.

Hunt and Thomas [22] in their book “*The Pragmatic Programmer*” mention the simple design rules, stressing in Chapter 2 the relationship between Duplication (3<sup>rd</sup> rule) and Orthogonality: “The first warns not to duplicate knowledge throughout your systems, the second not to split any one piece of knowledge across multiple system components”.

### B. Applications of Conceptual Integrity

After Frederick Brooks’ proposal of Conceptual Integrity as a fundamental idea for software development, researchers tried to apply Conceptual Integrity in practice. Despite the absence of formal quantitative criteria, these authors interpreted the Conceptual Integrity principles in ways similar to the Agile-Design-Rules, in particular rules 3 and 4.

Kazman and Carriere [27] extracted a meaningful software architecture using *conceptual integrity*. The guideline was a small number of components connected in regular ways, minimizing numbers of entities (rule 4). Kazman [26] described a SAAMtool, in which *Conceptual Integrity* is estimated by the number of primitive patterns of a system.

Clements et al. [9] interpreted *conceptual integrity* as “similar things should be done in similar ways”, with parsimonious data and control, i.e. duplication avoidance and minimization of entities (rules 3, 4). They suggested counting entities as a way to quantify *Conceptual Integrity*.

### C. Algebraic Structures for Software Systems

In this work we focus on the Modularity Matrix [13]. Other matrices have been used for modularity analysis. The Laplacian (von Luxburg [28]) has been used in various applications. Exman and Sakhnini [17] derived a Laplacian matrix with equivalent information to the Modularity Matrix, obtaining the same modular design for a given software system.

The ‘Design Rules’ by Baldwin and Clark [1], despite the name similarity to the Agile-Design-Rules, have a very different character. This approach is based upon a Design Structure Matrix (DSM), whose design quality is estimated by an external economic theory superimposed on the DSM. It has been mostly applied to non-software systems, and also to some software systems, e.g. Cai et al. [8]. A key difference from the Modularity Matrix is the lack of model linearity of the DSM.

Conceptual lattices, analyzed within Formal Concept Analysis (FCA) were introduced in Wille [32]. An overview of its mathematical foundations is given by Ganter and Wille [20]. The equivalence between Modularity Matrices and Conceptual Lattices has been shown by Exman and Speicher [14], which justifies dealing with structures as concepts.

## III. CONCEPTUAL SOFTWARE DESIGN: REVEALING INTENTION

We start here the systematic interpretation of the Agile-Design-Rules. By Fig.1 Conceptual Integrity directly explains the first two rules.

### A. A Separability Principle for Software

The first Agile-Design-Rule deserves special consideration. To this end we need a Separability Principle for Software Engineering. We have formulated such a principle in [16]. It states the following:

**“Software Proper vs. Human Concerns Separability Principle”** – theories dealing with software proper are separable from theories dealing with human concerns of software engineering.

This Separability principle says that theories dealing with properties of the software system proper are independent of theories dealing with human stakeholder concerns, either developer processes or end-user interactions with developers.

### B. Relevance to Agile Software Development

The Separability Principle is relevant to the possible meaning of the 1<sup>st</sup> Agile-Design-Rule, which tells us to continuously run all tests and make sure they still pass. This depends on the particular interpretation of this rule.

The first interpretation is trivial: a code with bugs is not runnable, and no next rule is applicable to code quality analysis. Successful tests are a pre-condition for the next rules.

Another interpretation directly touches pair-programming characteristics of XP. Pair-programming works by one person writing the code while the other person of the pair writes tests to be run on the written code, and then they switch the programmer/tester roles. From this viewpoint this rule concerns the human social aspects of development, and is not relevant to the software product proper.

In our view, the best interpretation touches the motivation for testing. The importance of tests is not just for finding bugs, but rather to enforce system redesign, in case design problems were identified. In this view, testing is an inherent part of the software product design and not an extraneous human concern. But was this the truly original motivation behind this rule?

### C. 1<sup>st</sup> Agile-Design-Rule: Passing Design Tests

Whatever was the original motivation behind the first Agile-Design-Rule, we propose here a novel interpretation consistent with our emphasis on Design instead of implementation or development process.

The goal of the 1<sup>st</sup> Agile-Design-Rule is to pass systematic “*Design Tests*”, viz. to reveal design problems conflicting with

Conceptual Integrity. This new focus on design means that the tests themselves should be carefully designed to be consistent with the SUD (System Under Design) Conceptual Integrity. Design test examples will be given in section V.

#### D. 2<sup>nd</sup> Agile-Design-Rule: Revealing Intention

The focus on design interpretation of the first rule is a suitable transition to the deep meaning of the 2<sup>nd</sup> Agile-Design-Rule. This rule in the formulation presented in the Introduction of this paper (in sub-section A) reads “Explicit Intent”, viz. to explicitly express the ideas of the software author. In other words, the concepts embodied in the software design units should both reflect the main ideas of the software system and be clearly understood by other stakeholders reading the software. Summarizing, *Conceptual Integrity* is not only essential to high-quality design, it should be explicitly revealed in the software itself, and not just in its documentation.

### IV. THE ALGEBRAIC SOFTWARE THEORY IS QUANTITATIVE!

To be applicable to the practice of software system design an actual Software Theory should be quantitative, as it is clear even in the naïve formulation of the rules: “*no duplication*” and “*minimize entities*”. In this section we provide formulas for calculating the relevant quantities, to explain rules 3 and 4 and later on propose a 5<sup>th</sup> rule.

#### A. A Quantitative Theory of Agile-Design-Rules

The quantitative algebraic Software Theory, the Linear Software Models, which in turn is based upon Conceptual Integrity (see Fig. 1), obeys the following demands:

- **Software represented by a mathematical structure** – be it a matrix or a lattice; in this paper we chose the matrix representation;
- **Quantities in formulas amenable to calculation** – getting precise numbers for each obtained design;
- **Standard Criteria for design quality** – allowing comparison of proposed designs with standards;

Quantities involved in the Conceptual Integrity calculations are normalized. These quantities are independent of the vector/matrix sizes, by dividing results by relevant entity sizes.

#### B. 3<sup>rd</sup> Agile-Design-Rule: No Duplication

“No duplication” in terms of vectors, is the simplest case of linear independence: any set of identical structures are obviously linearly dependent and all but one should be eliminated. The same is true for identical functionals. Thus, the 3<sup>rd</sup> Agile-Design-Rule is a particular case of the 4<sup>th</sup> rule discussed next.

#### C. 4<sup>th</sup> Agile-Design-Rule: Minimize Entities i.e. Propriety

Following Exman and Katz [18], the naïve “Minimize Entities” rule corresponds to the generic linear independence *Propriety* principle of Conceptual Integrity. Linear independence within a module is evaluated by equation (1), in which  $\mathbf{r}$  is the rank and  $\mathbf{c}$  is the number of columns of the module sub-matrix. Since module sub-matrices are square, one

could use as well the number of rows instead of the number of columns. The module propriety criterion in equation (1) has a value between zero and the maximum propriety value of 1 obtained when  $\mathbf{r}$  equals  $\mathbf{c}$ .

$$\mathbf{Propriety} = 1 - ((\mathbf{c} - \mathbf{r})/\mathbf{c}) \quad (1)$$

#### D. Orthogonality

As already mentioned, Hunt and Thomas [22] linked in their book the “No duplication” rule with Orthogonality. The latter quantity is calculated as follows. Assume a pair of normalized vectors  $\mathbf{u}$  and  $\mathbf{v}$  i.e. all their elements are divided by the length of the respective vector. Their Orthogonality is calculated by equation (2), where  $(\mathbf{u} \cdot \mathbf{v})$  is the vectors’ scalar product. Orthogonality has a value between zero and the maximal value 1 obtained for zero scalar product.

$$\mathbf{Orthogonality} = 1 - (\mathbf{u} \cdot \mathbf{v}) \quad (2)$$

Software system calculations, using the above equations, should be done for the whole set of Modularity Matrix modules to obtain the combined system conceptual integrity.

### V. DESIGN TESTS ILLUSTRATED BY CASE STUDIES

The Agile-Design-Rules are here illustrated by Case Studies. They are numbered and presented according to the rational interpretation given by the algebraic Software Theory, and adding a fifth Orthogonality rule.

#### A. 1<sup>st</sup> Agile-Design-Rule: Design Tests – ATM Conceptual Integrity Case Study

*Design Tests* are distinct from Unit Tests whose purpose is to find syntactic or logical errors. A design test, may check the Conceptual Integrity of a sub-system. For instance, an ATM (Automatic Teller Machine) is a reasonable machine to deposit or withdraw cash or deposit checks. But it is not currently an acceptable way to obtain a house mortgage.

Thus, a design test to verify an ATM design for Conceptual Integrity is a loop on a Financial Ontology, looking for and flagging for deletion all concepts appearing in the ATM design that are related or sub-types of the mortgage concept.

#### B. 2<sup>nd</sup> Agile-Design-Rule: Revealing Intention – Interdisciplinary Ambiguity Case Study

Revealing Intention is again a matter of Conceptual Design verification. Trivial cases are to demand naming of classes and functions by meaningful names such as “Bridge” or “Liquid”, instead of meaningless names such as “X” or “Y” (see e.g. [29]), or even worse, misleading names.

Less trivial cases deal with ambiguity, for instance in an interdisciplinary software in which the same term has different meanings in two disciplines. An example is the usage of the “Bridge” software design pattern within an application for



civil engineering dealing with tunnels and “bridges”. Another example is the usage of “Liquid” financial assets within an application about “Liquid” chemicals.

In order to verify ambiguity absence one may build an SUD (Software Under Development) Application Ontology, from the domain ontologies intersection, and check whether the same term appears in different Application Ontology branches dealing with the different disciplines.

### C. 3<sup>rd</sup> Agile-Design-Rule: No duplication – Circle Functionals Case Study

As already stated above, “No duplication” is a particular simple case of Linear Dependence. Whenever there are two or more identical functionals (similarly for identical structors), one should eliminate all of them except one.

For instance, assume a geometrical application involving circles. The Modularity Matrix has a “circle” structor. Suppose it also has two functionals – calculate *area* by  $\Pi \cdot \text{Radius}^2$  and calculate *perimeter* by  $2 \cdot \Pi \cdot \text{Radius}$ . Then there are two identical rows in the matrix, in which there are 1-valued elements for these two rows in the same circle structor column. One should eliminate duplication, since both these functions depend only on the Radius variable; when one fixes either the Area or the Perimeter, the Radius is determined and also the value of the other function. These functionals are trivially dependent.

### D. 4<sup>th</sup> Agile-Design-Rule: Minimize Number of Entities – General Propriety Case Study

The Propriety principle of Conceptual Integrity effectively minimizes the numbers of structors and respective functionals in a Modularity Matrix representation of a software system. Whenever there are linear dependences of row or column vectors within the matrix, one must eliminate some vectors to obtain total linear independence in the matrix. This is checked by equation (1), in which the matrix rank  $r$  should be equal to the number of structors (columns), or equivalently the number of rows (functionals). If Propriety is less than 1 by equation (1), some vectors must be eliminated by the software engineer, using semantic considerations.

For instance, in elementary trigonometry there are various cases of mutually dependent functions, in which one needs a lesser number of independent functions. To calculate the values of sine, co-sine and tangent functions of an angle in radians, one needs at most two of these functions.

### E. 5<sup>th</sup> Agile-Design-Rule: Orthogonality – Redesign to Eliminate Coupling Case Study

The Software Theory leads us to add a fifth Agile-Design-Rule in our formulation to comply with the Orthogonality principle of Conceptual Integrity, which is obeyed by the standard Modularity Matrix. It means that all structors and functionals of a given module should be respectively orthogonal to all structors and functionals of all other modules

in the software system represented by the Modularity Matrix. Orthogonality is calculated by repeated application of equation (2). If the overall matrix orthogonality is not 1, with some sparse modules, there is a case of coupling and the software system must be redesigned by the software engineer to eliminate coupling and assure orthogonality.

For instance, in a sub-system whose purpose is geodesy applications, a module performing proper geodetic calculations should be orthogonal to a module containing generic algebraic functions needed for e.g. matrix computations that may be needed within the geodetic calculations. Any redefinition of a generic algebraic function within a proper geodetic class, causes coupling of the geodetic and the algebraic modules, in need of redesign.

## VI. DISCUSSION

### A. Agile-Design-Rules: Plausibility of the Conceptual and Algebraic Software Theory

Our analysis in this work of the four original Agile-Design-Rules in the formulation by Jeffries, as displayed in sub-section A of the Introduction to this paper, shows the following picture:

- For consistency of the 1<sup>st</sup> rule on running tests with the other rules, we proposed a novel interpretation in which tests should be essentially **Design Tests**, instead of just debugging unit tests;
- The 2<sup>nd</sup> rule says that Conceptual Integrity besides being a general demand, it must be explicitly expressed in the names of the entities, such as classes and functions;
- The 3<sup>rd</sup> and 4<sup>th</sup> rules are completely explained by the Propriety principle which is part of the Conceptual Integrity approach; quantitatively it corresponds to the demand of Linear Independence among structors and among functionals in the Modularity Matrix;

Overall, the explanations for the Agile-Design-Rules reinforce the plausibility of the algebraic Linear Software Models, based upon Conceptual Integrity, as a Software Theory of software composition.

### B. Rules Variability: Selectivity, Numbers and Order

Any theory proposed to explain and justify methodological rules of development, must be a self-consistent theory. A possible outcome is that justification must be selective, i.e. not all practical rules are derivable from the Software Theory and the theory may generate additional practical rules.

In the particular case of the Agile-Design-Rules, the 1<sup>st</sup> rule, on running tests, has a novel interpretation in order to comply with the Software Theory self-consistency. Furthermore, a new reasonable 5<sup>th</sup> rule of Orthogonality has been explicitly generated, as suggested by Hunt and Thomas [22].

The particular order of the rules seems less important, as long as they rigorously follow from the Software Theory. The

rule order is perhaps of interest for rule classification, in which the 1<sup>st</sup> and 2<sup>nd</sup> rules strictly belong to a Conceptual viewpoint and the 3<sup>rd</sup> and 4<sup>th</sup> rules belong to an algebraic viewpoint.

### C. Future Work

In order to solidify the explanation and justification for the Agile-Design-Rules one needs to analyze software system examples of a variety of sizes.

Another open issue is the applicability of these or similar rules to other development methodologies.

While linear independence is relevant to Modularity Lattices, their orthogonality deserves further investigation.

### D. Main Contribution

There are three main contributions of this paper. 1<sup>st</sup>, it argues that Linear Software Models, the algebraic Software Theory based upon Conceptual Integrity, is a rigorous basis for software design methodologies. 2<sup>nd</sup>, it shows that the Agile-Design-Rules essence is selectively explained and justified by the Software Theory. 3<sup>rd</sup>, it proposed the idea of systematic Design Tests.

### ACKNOWLEDGMENT

The author thanks Reuven Yagel for his useful suggestions which contributed to improve the paper.

### REFERENCES

- [1] C.Y. Baldwin and K.B. Clark, *Design Rules*, Vol. I. The Power of Modularity, MIT Press, Cambridge, MA, USA, 2000.
- [2] K. Beck, *Extreme Programming Explained: Embrace Change*, 1<sup>st</sup> edition, Addison-Wesley, Boston, MA, USA, 1999.
- [3] K. Beck, "Embracing Change with Extreme Programming", *IEEE Computer*, Vol. 32, pp. 70-77, October 1999. DOI: [10.1109/2.796139](https://doi.org/10.1109/2.796139)
- [4] K. Beck and M. Fowler, *Planning Extreme Programming*, Addison Wesley, Boston, MA, USA, 2000.
- [5] N. Bekkers, "4 Rules of Simple Design", 2016. Web: <https://www.theguuild.nl/4-rules-of-simple-design/>
- [6] F.P. Brooks, *The Mythical Man-Month – Essays in Software Engineering – Anniversary Edition*, Addison-Wesley, Boston, MA, USA, 1995.
- [7] F.P. Brooks, *The Design of Design: Essays from a Computer Scientist*, Addison-Wesley, Boston, MA, USA, 2010.
- [8] Y. Cai and K.J. Sullivan, "Modularity Analysis of Logical Design Models", in *Proc. 21<sup>st</sup> IEEE/ACM Int. Conf. Automated Software Eng. ASE'06*, pp. 91-102, Tokyo, Japan, 2006.
- [9] P. Clements, R. Kazman and M. Klein, *Evaluating Software Architecture: Methods and Case Studies*. Addison-Wesley, Boston, MA, USA, 2001.
- [10] S.P. De Rosso and D. Jackson, "What's Wrong with Git? A Conceptual Design Analysis", in *Proc. of Onward! Conference*, pp. 37-51, ACM, 2013. DOI: <http://dx.doi.org/10.1145/2509578.2509584>.
- [11] I. Exman, "Linear Software Models", video presentation of paper at GTSE 2012, KTH, Stockholm, Sweden, 2012b. Web site: <http://www.youtube.com/watch?v=EJfzArH8-ls>.
- [12] I. Exman, "Linear Software Models: Standard Modularity Highlights Residual Coupling", *Int. Journal of Software Engineering and Knowledge Engineering*, Vol. 24, pp. 183-210, 2014. DOI: [10.1142/S0218194014500089](https://doi.org/10.1142/S0218194014500089).
- [13] I. Exman, "Linear Software Models: Decoupled Modules from Modularity Matrix Eigenvectors", *Int. Journal of Software Engineering and Knowledge Engineering*, Vol. 25, pp. 1395-1426, 2015. DOI: <https://doi.org/10.1142/S0218194015500308>
- [14] I. Exman and D. Speicher, "Linear Software Models: Equivalence of the Modularity Matrix to its Modularity Lattice", in *Proc. 10<sup>th</sup> ICSOFT'2015 Int. Conference on Software Technology*, pp. 109-116, ScitePress, Portugal, 2015. DOI: [10.5220/0005557701090116](https://doi.org/10.5220/0005557701090116)
- [15] I. Exman, "Linear Software Models: An Algebraic Theory of Software Composition", in *Proc. 28<sup>th</sup> Int. Conf. on Software Engineering and Knowledge Engineering*, Keynote Abstract, KSI Research, Redwood City, CA, USA, 2016.
- [16] I. Exman, D.E. Perry, B. Barn and P. Ralph, "Separability Principles for a General Theory of Software Engineering: Report on the GTSE 2015 Workshop", *ACM SIGSOFT Software Engineering Notes* 41 (1): 25-27 (2016). DOI = [10.1145/2853073.2853093](https://doi.org/10.1145/2853073.2853093)
- [17] I. Exman and R. Sakhnini, "Linear Software Models: Modularity Analysis by the Laplacian Matrix", in *Proc. 11<sup>th</sup> ICSOFT'2016 Int. Conference on Software Technology*, Volume 2, pp. 100-108, ScitePress, Portugal, 2016. DOI: [10.5220/0005985601000108](https://doi.org/10.5220/0005985601000108)
- [18] I. Exman and P. Katz, "Conceptual Software Design: Algebraic Axioms for Conceptual Integrity", in *Proc. 29<sup>th</sup> Int. Conf. on Software Engineering and Knowledge Engineering*, pp. 155-160, KSI Research, Pittsburgh, PA, USA, 2017. DOI: <https://doi.org/10.18293/SEKE2017-148>
- [19] M. Fowler, "Beck Design Rules", Blog, March 2015, URL: <https://martinfowler.com/bliki/BeckDesignRules.html>.
- [20] B. Ganter and R. Wille, *Formal Concept Analysis: Mathematical Foundations*, Springer-Verlag, Berlin, Germany, 1998.
- [21] C. Haines, "Understanding the Four Rules of Simple Design", Leanpub, 2014.
- [22] A. Hunt and D. Thomas, *The Pragmatic Programmer: From Journeyman to Master*, Addison-Wesley, Boston, MA, USA, 1999.
- [23] D. Jackson, "Conceptual Design of Software: A Research Agenda", CSAIL Technical Report, MIT-CSAIL-TR-2013-020, 2013. URL: <http://dspace.mit.edu/bitstream/handle/1721.1/79826/MIT-CSAIL-TR-2013-020.pdf?sequence=2>
- [24] D. Jackson, "Towards a Theory of Conceptual Design for Software", in *Proc. Onward! 2015 ACM Int. Symposium on New Ideas, New Paradigms and Reflections on Programming and Software*, pp. 282-296, 2015. DOI: [10.1145/2814228.2814248](https://doi.org/10.1145/2814228.2814248).
- [25] R. Jeffries, "Essential XP: Emergent Design", October 2001. URL: <https://ronjeffries.com/xprog/classics/expemergentdesign/>.
- [26] R. Kazman, "Tool Support for Architecture Analysis and Design", in *ISAW'96 Proc. 2<sup>nd</sup> Int. Software Architecture Workshop*, pp. 94-97, ACM, New York, NY, USA, 1996. DOI: [10.1145/243327.243618](https://doi.org/10.1145/243327.243618)
- [27] R. Kazman and S.J. Carriere, "Playing Detective: Reconstructing Software Architecture from Available Evidence." Technical Report CMU/SEI-97-TR-010, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, USA, 1997.
- [28] U. von Luxburg, "A Tutorial on Spectral Clustering", *Statistics and Computing*, 17 (4), pp. 395-416, 2007. DOI: [10.1007/s11222-007-9033-z](https://doi.org/10.1007/s11222-007-9033-z)
- [29] K. Owen, "What's in a Name? Anti-Patterns to a Hard Problem", 2016. Web: <https://www.sitepoint.com/whats-in-a-name-anti-patterns-to-a-hard-problem/>
- [30] J.B. Rainsberger, "The Four Elements of Simple Design", 2016. Web: <http://blog.jbrains.ca/permalink/the-four-elements-of-simple-design>
- [31] G. Sironi, "The 4 rules of simple design", 2011. Web: <https://dzone.com/articles/4-rules-simple-design>
- [32] R. Wille, "Restructuring lattice theory: an approach based on hierarchies of concepts" In: I. Rival (ed.): *Ordered Sets*, pp. 445-470, Reidel, Dordrecht-Boston, 1982.

# Classutopia: A Serious Game for Conceptual Modeling Design

Felipe Larenas<sup>a</sup>, Beatriz Marín<sup>a</sup>, Giovanni Giachetti<sup>b</sup>

<sup>a</sup>Facultad de Ingeniería y Ciencias, Universidad Diego Portales, Chile  
{felipe.larenas, beatriz.marin}@mail.udp.cl

<sup>b</sup>Universidad Tecnológica de Chile INACAP, Chile (ggiachetti@inacap.cl)

**Abstract**— One of the more complex topics to teach to software engineering students is the conceptual modeling design, which has several concepts that students must learn in order to specify the structural, behavioral and interaction views of software systems. Learning the design of class diagrams is of paramount importance since these diagrams are used to guide concrete development tasks such as programming and software testing, and -consequently- to avoid defective software products. Applying novel teaching/learning techniques in this topic may help students to reduce the defects that are committed at the moment of designing a class diagram. One interesting technique is the use of serious games, due to the fact that they provide learning environments free of risks and pressure for students, allowing the students to know the topics that they must learn in a fun way. Serious games have been widely used in programming courses. Thus, we aim to investigate the feasibility to replicate this experience for conceptual modeling of class diagrams at software engineering courses. In this paper, we present a role-playing game specially focused in the class diagram, which is called *Classutopia*. This serious game provides modeling challenges, comprehension and correction of diagrams with different complexity levels for learning conceptual modeling design.

**Keywords**— *Gamification, serious games, class diagram, software engineering.*

## I. INTRODUCTION

Conceptual modeling is one of the more complex topics that must be learnt by software engineering students since they need to abstract concepts from reality and express them in computational terms. The conceptual modeling design is of paramount importance in the development of software projects because it specifies the different views of a software system, which guide the programming and the testing tasks. The class diagram is one of the most representative design approaches for software modeling [1], which serves as a guideline to develop the structures and the methods of a software product. A faulty class diagram misleads the software development teams, leading to mistakes in coding and delays in project implementations, and subsequently causing project failures.

Some common defects encountered in the development of class diagrams [2–5] are as follows: confusion amidst elements such as aggregation and composition; overspecification; errors in the cardinality, i.e., creating infinite recursive associations; poor choices of names for classes and attributes; class

replications; lack of inheritance between classes of the same type; lack of classes, attributes, and lost methods.

The defects mentioned above can be overcome if the students actively participate in the learning process. The use of games and simulation-based experiences has been of great help for teaching and offering learning environments without any risk. These games are designed for a different purpose than to solely entertain and are known as serious games [6,7]. Moreover, the use of videogames characteristics such as points or rewards in a game is known as gamification. Currently, there are several systematic reviews in the literature [8,9] that have reported the positive results of serious game-based learning, wherein the use of this type of game has been verified in various domains of learning in topics related to health, education (school and university), culture, social skills, and vocational training, among others. Within the domain of education, specifically in the area of computing [10], it has been used to teach topics such as programming, project management software, and operating systems.

We believe that by implementing serious games, specifically focused on learning conceptual modeling, it is possible to motivate and entertain students by modifying their behavior in a positive manner when coping with class diagrams. Therefore, the design of a serious game for learning and understanding conceptual modeling design is presented in this paper. The game is a Role-Playing Game (RPG) wherein a character hero (characterized by a robot) is assigned to a player who must complete an adventure by overcoming three types of challenges to defeat the villains (characterized as evil wizards). As the game progresses, the difficulty of the challenges will increase based on the predefined problems.

Therefore, the contribution of this work is the design of a serious game for conceptual modeling learning of the class diagram and the implementation of a mobile application for Android that provides support for running the game on tablets. This contribution is useful for students that want to learn how to do a class model design avoiding defects, and it is also useful for researchers that want to add new challenges to the *Classutopia* serious game.

The rest of the article is organized as follows: Section 2 presents some relevant related works. Section 3 presents the design of a serious game developed for learning conceptual modeling of the class diagram. Section 4 presents the

evaluation of the game. Finally, Section 5 presents the main conclusions and future works.

## II. RELATED WORK

Even though there are several serious games published in literature, as we can observe in the systematic literature reviews presented in [8,9], we did not find literature of serious games that had been applied to teaching/learning class diagram. Nevertheless, some of the games that are focused on teaching object-oriented programming concepts are closely linked to the modeling of classes, so that we review these documents in this section.

Java Ninja [11] evaluated the effects of utilizing serious games focused on teaching specific concepts of modeling and programming, wherein the students receive help in understanding the inheritance in object-oriented programming and show positive results in their learning processes and motivation.

In [12] a game-theme instructional model is presented, which demonstrates the concepts of object-oriented programming, specifically encapsulation, polymorphism, and inheritance. The game uses virtual-reality to engage students with 3 problems related to the programming topics mentioned above. The student must answer correctly in the inheritance situation in order to construct a class model diagram. Results indicate that 80% of students agreed that the module developed their interest in programming.

In [13] a game focused in object-oriented programming concepts is presented. In this game, the players can select an object and the class diagram is displayed in order to show the inheritance relationships of the selected object.

A serious game focused in the education of object-oriented analysis and design and artificial intelligence is presented in [14]. The game is about the Sokovan problem, where five types of relationships can be identified: association, composition, dependency, inheritance and aggregation. Authors state that to facilitate applying serious game a promising strategy would be accumulating and opening game-based materials. So that, they provides this game with a puzzle style, but no information is provided about the usage or empirical evaluations.

In summary, we didn't find serious games specifically focused in learning conceptual modeling design, nevertheless clear examples of the use of class diagrams in challenges of serious games have been obtained. Considering the importance and difficulty of conceptual modeling learning in software engineering, a serious game has been created specifically for class diagram learning, which is shown in the next section.

## III. DESIGN OF CLASSUTOPIA SERIOUS GAME

The design of a serious game for the conceptual modeling of class diagram learning has been developed using the engine Unity, aiming toward it being executed in a mobile application for devices with Android operating system. The conceptual framework of the MDA (Mechanics, Dynamics and Aesthetics) [15] is used to explain the design of the game, which allows the implementation of the game by taking into account the player and the developer viewpoints.

### A. Aesthetics

For making the game attractive and fun for the player, the game has a minimalist graphic style, displaying easy-to-interpret buttons, lists, and menus. In a similar way, common visual representations of RPG games are used, such as health bars, characters' dialog windows, and sprites for the same. The visual feedback of the player's interaction with the elements on the screen is based on the changes of hues and approaches.

Moreover, an interesting context has been created to motivate playing the game. *Classutopia* is a technological utopia wherein human beings have reached the pinnacle of civilization thanks to the help of technology. Amidst all significant technological advances achieved, five are considered the most important, called *The Pillars*, in which all this social prosperity is underpinned: a satellite plant of solar energy, a synthesized food system, an organ cultivation center, an anti-pollution multi-industrial system, and a robots factory for unwanted tasks.

All the basic needs of humanity (health, food, and energy) have been met. The villain in this story is the last descendant of a caste of sorcerers, who is bothered by the fact that humanity has reached fullness thanks to technology and not magic. Therefore, he has initiated an attempt to sabotage *The Pillars*, making use of his magic tricks to modify this systems software by altering the class models to overthrow the utopia. However, a maintenance robot has become aware of these failures; the reason why it pursues an original adventure to repair the damages and ruin the sorcerer's plans. This context works as a guideline for the creation of the diagrams arranged for the missions since each mission represents the robot's attempt to save one of *The Pillars*. A diagram is modeled to represent the satellite plant of solar energy, synthesized food system, and organ cultivation center; these diagrams are presented in the missions 1, 2 and 3, respectively.

*Classutopia* is available for download at <https://www.dropbox.com/s/sfcjz0jlp6a911/Classutopia.apk?dl=0>

### B. Mechanics

The game is based on three types of mechanics: modeling the character, correcting defective diagrams, and understanding diagrams.

#### 1) Character Modeling

The main character is a robot with three basic attributes: attack, defense, and speed. Attack relates to the damage the protagonist can cause to the enemy, defense is the resistance posed to the damage received from the enemy, and speed is the management of the limited time to solve the problems. These attributes can be modified and enhanced through modeling of class diagrams, which describe the features and components of the robot. Initially, the robot needs to be modeled correctly, and subsequently, as the game progresses, new improvements can be unlocked, which should be properly included in the class model of the robot so that the changes take effect. These improvements, in addition to modifying the character's attributes, also unlock some special powers that can be used in the clashes between the protagonist robot and its enemy (the magician).

### 2) Correction of Defective Diagrams

During the game, the hero battles against the enemy; the disputes are based on the challenge of correcting defective class diagrams. In these challenges, given a defective class diagram, the player can tap the screen of the tablet to highlight the various elements that make up the class diagram, which have been considered to be defective. Therefore, *Classutopia* offers options to correct the possible defects. Once the correction has been made, feedback on whether the player has made a mistake or done it correctly is not instantaneous but the player must “Attack” to be able to assess the correction. This has been designed in this manner to give an option to the player to evaluate one or more modifications made to the diagram by pressing a single button, in which mistakes result in harming the protagonist (robot) and the successful corrections affect the enemy (magician). This is graphically presented on the characters’ health bars that indicate the status for each character. In addition, *Classutopia* delivers feedback in the class diagram; it indicates in green the modification that was applied correctly and in red where it has been incorrectly modified. It also includes a button to undo the recent applied modifications.

### 3) Understanding Diagrams

During the battles, the player can activate special powers in the robot that allow him to perform various effects such as enabling damage modifiers to apply more force on the attacks, reducing corrections in the diagram to finish it on time, or healing the protagonist. These modifiers have limited use and are activated by means of a challenge in understanding class diagrams. By pressing the button which activates a certain power, the player must answer a question with respect to the concepts that can be used in a class diagram. This question displays a small diagram or some conceptual construct and offers between two and four possible answers. If the player answers correctly, the power is effective; if answered incorrectly, the effect will not be applied. The powers are arranged in the battle according to the previously modeled characteristics for the robot.

## C. Dynamics

The mechanics previously described have been presented to the player in a smooth, consistent, and understandable way during the execution of the game. To obtain the expected gameplay, a series of screens, menus, and systems are implemented, which are explained below.

### 1) Control Systems

Like most mobile applications available today, the game is controlled via touch-screen by simply tapping on buttons and menu items displayed on the screen. In some game mechanics, multi-touch gestures are used to zoom in and out and scroll through the proposed class diagrams.

### 2) Start screen and save game handling

When the player runs the application, a screen is shown with the name of the game, a representative image, and a Start button to initiate the game. It is a simple landing screen like that of most mobile games. Once the player has clicked on the

Start button, the button disappears presenting to the player two new options within the same screen (see Figure 1): one to start a new game and the other to load a previous game. The game uses a system of unitary games, i.e., you can only save one game at a time, therefore, creating a new game automatically deletes (if any) the previous game. For this reason, as a precaution, a warning window to confirm deletion is displayed to avoid accidental loss of the game. The games are saved automatically, indicating this with a symbol at the end of each challenge. If the player presses the button to start a new game, the stored data are reestablished and the game moves on to the main menu. If the player presses the button Load Previous Game, it goes directly to the main menu preserving the conditions recorded during the last auto-save.



Fig. 1. Game selection. Upper text: Classutopia. Lower left text: START NEW GAME. Lower right text: LOAD PREVIOUS GAME

### 3) Main menu

The main menu displays three buttons (see Figure 2): one to access the construction challenge of modeling the main character, one to access the missions’ menu, and one to return to home screen. In addition, it displays a summary table with the current attributes of the character robot. The first time the player accesses this screen (when he/she starts a new game), the missions button is disabled in order to force the player to model the character on the construction screen before accessing the missions’ menu.



Fig. 2. Main menu. Upper right text: CURRENT ATTRIBUTES: ATTACK 1, DEFENSE 1, SPEED 1, HEALTH 100. Left text: BUILD UP, MISSIONS.

### 4) Build and upgrades

On the construction screen (see Figure 3), the mechanical modeling of the character is presented, as has been previously described. Initially, the player must connect the classes that represent the parts of the robot correctly, for example, sensors, extremities, core, and weapons. The player can see, as he/she models the robot appropriately, how its attributes (attack, defense, speed, and unique skills) modify according to the characteristics of the diagram. Defects in modeling during the

construction of the diagram result in the non-modification of the attributes, i.e., the classes do not apply their effects and the features of the robot are not improved. There is a button to save the changes made, continuing to the main menu. Within this screen, the list of improvements for the robot is presented, those that get unlocked, as a reward, each time a mission is completed. These represent the improved components of the robot or new components that, by being adequately included in the model, increase the attributes and activate new powers.

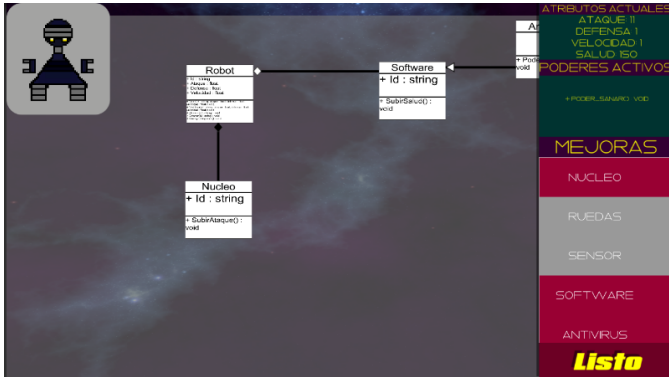


Fig. 3. Building screen.

### 5) Missions

After modeling the robot for the first time, in the main menu, the button to access the missions' menu is enabled. The missions correspond to each of the clashes the robot must complete by defeating the magician (see Figure 4). These tasks are presented as a list and are sequential, i.e., the second mission is enabled when the previous mission is completed, and the third is activated upon completion of the second mission. Each mission is shown with a name and its level of difficulty. The difficulty of the missions is incremental. In the first instance, the game has three missions: the basic level, mid-level, and advanced level. The missions can be repeated for practicing or unlocking new rewards (those corresponding to the accomplished level). To start a mission, the player must tap on it, by doing so it gives way to the battle.



Fig. 4. Missions menu.

### 6) Battle

The battle screen shows the class diagram to be corrected (as explained in Mechanics Section), which can be navigated using tactile gestures (see Figure 5). The avatars that represent

the parts in conflict are shown at the top of the screen with their respective health bars and with the remaining time for the mission. The confrontation ends when the time runs out, when one of the parties loses all their health or when all defects are corrected. If the player (robot) wins, the level is completed, regards are awarded, progress of the game is saved, and the missions menu appears. If the player (robot) loses, the missions menu appears without rewards but with the option to reattempt to complete the mission.

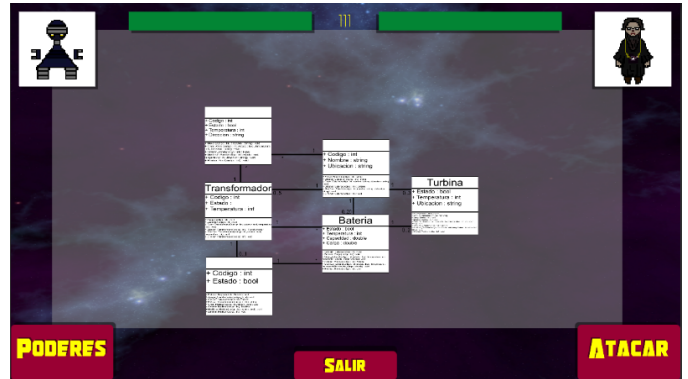


Fig. 5. Battle screen.

Each level focuses on the correction of different groups of defects in class diagrams. The basic level focuses on learning the essential elements within a class (names and attributes). The medium level focuses on learning the correct specification of services in a class. The advanced level focuses on learning the associations and cardinalities. Table 1 shows the injected defects in each of the levels.

TABLE I. DEFECTS INJECTED IN CLASSUTOPIA LEVELS

Level	Defects
Basic	<ul style="list-style-type: none"> <li>• Classes without name</li> <li>• Classes without id</li> <li>• Classes with repeated attributes</li> <li>• Classes with attributes without data type</li> </ul>
Medium	<ul style="list-style-type: none"> <li>• Classes without creation service</li> <li>• Classes with service without return</li> <li>• Classes without visualization of attributes</li> </ul>
Advanced	<ul style="list-style-type: none"> <li>• Associations of wrong type</li> <li>• Association without cardinality</li> <li>• Minimum cardinality of 1 at both ends</li> <li>• Descending cardinality at one end of the association</li> </ul>

The more advanced levels include the types of defects of the levels of lesser complexity. These defects are applied randomly at the beginning of a confrontation on a diagram without defects, i.e., the game can generate any of these defects in any of the elements of the class diagram presented in a battle and generate another type of defect, completely different, in the same elements when repeating the mission.

### 7) Special Powers

On the battle screen, there are buttons to activate the special skills obtained in the construction of the robot (see Figure 6). By tapping on a skill, the understanding problem is presented, which must be answered correctly for its activation. While this occurs, time is not paused. Once the problem is addressed, the

battle continues and if answered correctly, the effects of the skill are applied; otherwise, the effects do not apply. Whether it is answered correctly or incorrectly, the selected skill is disabled for the rest of the combat. The types of effects that the skills have are as follows: Multipliers of special powers, Health recovery of the protagonist, Mission's time manipulation or Reduction of remaining defects.



Fig. 6. Special powers panel.

#### IV. EVALUATION OF CLASSUTOPIA

An exploratory empirical evaluation of *Classutopia* has been conducted with the aim of assessing the understanding of the game and the perceived utility to support the conceptual modeling learning in the Software Engineering course. For the design of this activity, the guidelines previously defined in [16] were used. The following research question was defined:

*RQ1: Do the students find the use of Classutopia for learning the conceptual design of class diagrams beneficial?*

The context of the experiment is included in the Software Engineering course, which is a fourth-year course of the Engineer degree in Computer Science and Telecommunications at the Diego Portales University (Chile). The subjects are students of this course that have already attended classes on conceptual modeling. These classes have been conducted using traditional techniques of teaching/learning such as PowerPoint slides and inspection exercises in diagrams printed on a sheet.

In the experiment, the subjects receive a small description of what is *Classutopia* and then they must play. They begin building the robot and go through the missions. Finally, the subjects must respond to a questionnaire in order to obtain the perceptions of the usefulness of the game as a new technique of teaching/learning of conceptual modeling in Software Engineering. The sentences of the questionnaire are presented at Table 2, and subjects must answer using a Likert scale, where 1 is totally disagree to 5 totally agree.

TABLE II. QUESTIONNAIRE TO ASSESS CLASSUTOPIA

Sentence	
1.	Classutopia facilitates the understanding of a class diagram.
2.	Classutopia helps determine how to correct a class diagram.
3.	Classutopia helps to understand the concepts of the class diagram.
4.	I find Classutopia easy to understand.
5.	Classutopia allows you to learn about the design of a class diagram in an easier way than by using text.
6.	In general, I found Classutopia to be useful.

#### A. Operation of the empirical evaluation

The research was conducted in first semester of 2017 with students who have completed the course of Software Engineering in the second half of 2016. Thirteen students participated voluntarily. Participation in the experiment was not related to the score of the students but the students were urged to make their best effort. The start and end of each student for each mission and the times they had to conduct a mission to win it were registered. It is important to mention that each student should complete mission 1 (M1) to unlock mission 2 (M2) and further complete M2 to unlock mission 3 (M3).

The students had 30 minutes to conduct the activity. However, most subjects wanted to continue playing to defeat the magician. The students realized that each time they earned a mission, they unlocked new features that could improve the robot and they were going to build the robot to improve their attributes. In addition, the subjects used the easier mission to unlock the features thanks to they win many times to build the robot to defeat the most powerful wizard in the last mission.

#### B. Results

Results are shown in Table 3. It can be observed that 7 subjects were able to correctly build the robot and detect all defects injected in the diagrams.

TABLE III. RESULTS

Subject	Time M1	Num. of M1	Time M2	Num. of M2	Time M3	Num. of M3	Win
S1	5	2	13	3	14	7	yes
S2	2	1	25	8	-	-	no
S3	3	1	27	6	22	9	yes
S4	4	3	7	2	15	5	yes
S5	5	3	22	5	-	-	no
S6	1	1	12	5	7	3	no
S7	3	2	10	4	6	3	yes
S8	11	11	7	2	46	16	no
S9	11	5	37	12	20	8	no
S10	2	1	1	1	6	3	yes
S11	4	2	28	7	21	6	no
S12	4	2	17	5	22	8	yes
S13	3	2	8	2	30	10	yes

Of these 7, 3 subjects exceeded the game in 30 minutes but 4 subjects were engaged in the game to overcome it. Four subjects correctly build the robot but did not exceed M3 that had defects in the associations of the classes. Two subjects did not exceed M2 that had defects in the services of the classes and did not continue to play after 30 minutes. It is important to note that the four subjects who did not win had to stop playing the game although they were eager to continue playing.

The results indicate that all students (13 subjects) could learn how to build the robot and to detect defects in the class attributes (basic level). 11 of them were able to learn how to build the robot and to detect defects at basic and medium levels. From these 11 students, 7 students could learn how to build the robot and to detect defects in the class diagrams at basic, medium, and advanced levels. This allows to observe whether it is feasible to use a serious game to learn about the conceptual modeling software, the game improves the

students' understanding of class diagram, and encourages us to improve the testing to evaluate the effectiveness in learning.

Regarding the survey, Questions 1, 2, and 3 are related to the perception of the subject matter of learning using the serious game. The results (see Figure 7) indicate that the students perceive that *Classutopia* helps in the comprehension of the class diagram, in the understanding of how to construct a class diagram, and that *Classutopia* helps to understand the concepts used in the class diagram (the majority of the subjects agree or totally agree with the statements of Q1, Q2, and Q3). Regarding the ease of use of the game (Q4), there are two subjects that did not find it easy to understand how *Classutopia* works. However, more than 50% of the students agree or totally agree that it is easy to understand how the game works. Regarding the utility of the game, the majority of students agree that *Classutopia* allows to learn more easy than using texts (Q5), and they are totally agree that in general found *Classutopia* useful (Q6). The results of this survey allows to answer the research question since the students find it beneficial to use a serious game for learning conceptual modeling, in particular, for learning class diagrams.

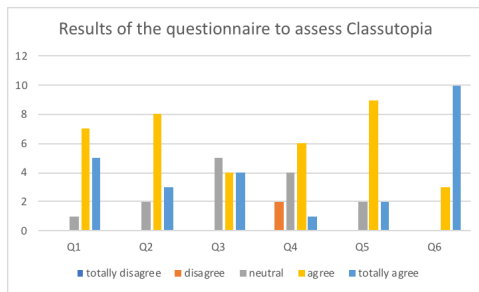


Fig. 7. Results of the questionnaire.

## V. CONCLUSIONS

Novel teaching techniques in computer courses are required, particularly for the advanced levels of Software Engineering, because in many cases, the emphasis of research has been placed on the most basic courses such as programming. The design, implementation, and use of a serious game to learn conceptual modeling in software engineering courses are presented in this study. From this work, it can be concluded that it is feasible to implement new teaching techniques with gamification and serious games for Software Engineering courses.

In the development of computer projects, modeling a correct class diagram is of vital importance for the reduction of problems and delays in deploying a software. In this work, we provide a solution for the problems caused by the misunderstanding of class diagrams and their semantic components. Thus, *Classutopia* has been presented, which is a serious game that supports the conceptual modeling learning, and it can be a good solution to correct the most common problems that students have to familiarize themselves with class diagrams, offering them a space free of pressure and consequences to learn and understand how to model and correct class diagrams.

In addition, there has been an exploratory empirical evaluation to verify the perception of students with respect of

this new method of teaching/learning in Software Engineering. Results indicate that *Classutopia* provides benefits for learning conceptual modeling design. We are aware that with a limited number of subjects, it is not possible to apply statistical techniques to increase confidence in the results. One of the limitations of the empirical assessment performed is the lack of evidence of the effectiveness of *Classutopia* in the learning process. Therefore, further work is referred to conduct experiments to evaluate the effectiveness of *Classutopia*. Finally, there are plans to improve the *Classutopia* graphics aspects that will help to enhance the gaming experience.

## ACKNOWLEDGMENT

This work was funded by CONICYT project ENSE RED1170020, 2017-2019.

## REFERENCES

- [1] OMG, Unified Modeling Language (UML) 2.4.1 Superstructure Specification, 2011.
- [2] D. Giordano, F. Maiorana, Object Oriented Design through game development in XNA, 3rd Interdisciplinary Engineering Design Education Conference (IEDEC), pp. 51-55, 2013.
- [3] J. Cabot, Common UML errors (I): Infinite recursive associations. <http://modeling-languages.com/common-uml-errors-i-infinite-recursive-associations/> (last access 14.03.18)
- [4] Guidelines for UML Diagram Development. <http://eng.umd.edu/~austin/nsf-crcd/uml-guidelines.html>(last access 14.03.18)
- [5] B. Marín, G. Giachetti, O. Pastor, A. Abran, Identificación de Defectos en Modelos Conceptuales utilizados en Entornos MDA, XII Ibero-American Conference on Software Engineering (CIbSE'2009), pp. 109-114, 2009.
- [6] C. Abt, Serious Games, University Press of America, 1987.
- [7] M. Zyda, From visual simulation to virtual reality to games, IEEE Computer, vol. 38, issue 9, pp. 25-32, 2005.
- [8] T. M. Connolly, E. A. Boyle, E. MacArthur, T. Hainey, J. M. Boyle, A systematic literature review of empirical evidence on computer games and serious games, Comput. Edu. Vol. 59, issue 2, pp. 661-686, 2012.
- [9] A. Calderón, M. Ruiz, A systematic literature review on serious games evaluation: An application to software project management, Comput. Edu. vol 87, pp. 396-422, 2015.
- [10] J. Vargas-Enríquez, L. García-Mundo, M. Genero, M. Piattini, Análisis de uso de la gamificación en la enseñanza de la informática, XXI Jornadas de la Enseñanza Univ. de la Informática (JENUI 2015), pp. 105-112, 2015.
- [11] J. Zhang, E. Caldwell, E. Smith, Learning the concept of Java inheritance in a game, 18th International Conference on Computer Games (CGAMES), pp. 212-216, 2013.
- [12] S. Sharma, J. Stigall, S. Rajeev, Game-theme based instructional module for teaching object oriented programming, International Conference on Computational Science and Comp. Intelligence (CSCI), pp. 252-257, 2015.
- [13] J. Livovský, J. Porubán, Learning object-oriented paradigm by playing computer games: concepts first approach, Central European Journal of Computer Science, vol. 4, issue 3, pp. 171-182, 2014.
- [14] Z. Li, L. O'Brien, S. Flint, Object-oriented Sokoban solver: A serious game project for OOAD and AI education, IEEE Frontiers in Education Conference (FIE) 2014.
- [15] R. Hunnicke, M. LeBlanc, R. Zubek, MDA: A formal approach to game design and game research, AAAI Workshop on Challenges in Game AI, 2004.
- [16] C. Wohlin, P. Runeson, M. Host, M. Ohlsson, B. Regnell, Experimentation in Software Engineering, Springer, 2012.



# Automatic Audience Focusing by Event Interestingness

Iaakov Exman, Yakir Winograd and Avihu Harush

Software Engineering Department

The Jerusalem College of Engineering – JCE - Azrieli

Jerusalem, Israel

iaakov@jce.ac.il, yakirwin@gmail.com, tchvu3@gmail.com

**Abstract**— Large social Networks have marketing potential to spread information about interesting events to suitable audiences. However, huge network sizes and varieties of information available are obstacles to reach the desired goal. This paper investigates the hypothesis of computable Interestingness as a criterion to focus on suitable audiences for any given event. Interestingness is calculated by combining two functions: Relevance and Surprise. A generic software tool has been developed as an experimental testbed to interact with any social network. Its inputs are the event characterization and audience candidates for the given event. Two results validate this work's hypothesis: first, audience candidates who actually visited the event site, have on the average a bigger computed Interestingness than the rest of the population; second and most important, computed Interestingness better differentiates event site visitors, actually interested in the Event, from non-visitors, while Relevance alone, does not distinguish so-well between visitors and non-visitors.

**Keywords:** *Interestingness; Automatic focusing; Event; Relevance; Surprise; Match; Mismatch; Randomization; Social Network; Software Architecture; Knowledge Discovery; Analytics; Marketing.*

## I. INTRODUCTION

Social Networks, by labeling their members with common interests, have the potential of efficiently marketing specific events. On the other hand, their huge sizes and proliferation of information types are impediments to reach the desired goals.

Our working hypothesis is that a computable Interestingness criterion focuses on suitable audience candidates for a chosen event, overcoming the sizes obstacle. Having a computable Interestingness, one automates its usage with a software tool. This tool is an experimental testbed for the working hypothesis.

The goal of this paper is to validate the working hypothesis by comparison of the calculated Interestingness with the behavior of audience candidates. The validation experiment consists of: 1) Send information items to audience candidates; 2) Compute the candidate's interestingness relative to the given event; 3) Compare it with the candidate action of visiting or not the Event Web site.

## II. RELATED WORK

We concisely review the literature related to Interestingness, its applications, and internet agents within social networks.

### A. Interestingness Concepts and Applications

Overviews of Interestingness measures for Data Mining and knowledge discovery are given by Geng et al. [6] and McGarry [11]. Interestingness approaches are described by Tuzhilin [13] in the Klosgen and Zytchow Handbook [9].

Criteria to determine interesting rules/patterns generated in data mining are discussed by Lenca et al. [10].

Exman, in 2009, [2] defined Interestingness as a product of relevance and surprise. This definition has been embodied in Web search software tools such as the one described in [4].

### B. Social Networks Applications

Social network bots, i.e. software robots, are ubiquitous, as seen in the book on Twitter and Society by Weller et al. [14]. Of particular interest is the Twitter Accounts chapter by Mowbray [12], describing Twitter marketing bots. Gentry et al. [7] analyze shop-bots, advising online shoppers about products and prices.

It is essential to distinguish bots from humans. Chu et al. [1] deal with this issue. Gilani et al. [8] also aim at bot recognition. Ferrara [5] reliably classify bots despite similar behavior to humans. Exman et al. [3] explored bot survivability within a human social net, as a kind of anti-Turing test.

## III. INTERESTINGNESS

Here we give generic and specific Interestingness definitions. Then, events and candidates are characterized.

### A. Interestingness Definitions

The assumptions behind the Interestingness definition are:

1. **Domain of interest choice is arbitrary** – one may express interest in pre-Columbian archeology or in Jazz music; any choice is a matter of personal taste;
2. **Unusual items attract more attention than average items** – unusual items should be given more weight than average ones, when computing interestingness.

*Interestingness* is a two function composition: the *Relevance* of an item to a domain chosen by one's personal taste and the *Surprise* caused by most unusual items, among the relevant ones. We simplify it to be just a commutative multiplication:

$$\text{Interestingness} = \text{Relevance} * \text{Surprise} \quad (1)$$

In this work an item is a candidate for a conference event, in a given domain. *Relevance* measures to what extent the candidate fits the event audience. *Surprise* measures to what extent the candidate for a conference event is outstanding, relatively to the average candidate for this event.

There exist several specific functions to calculate *Relevance* and *Surprise*. A well-known formula is *TfIdf* used in data mining. *Tf*, Term Frequency, expresses *Relevance*, based upon the chosen term frequency in a given document. *Idf*, Inverse Document Frequency, expresses *Surprise*, or rarity, i.e. inverse ratio of relevant documents relative to all examined documents.

In this work *match* and *mismatch*, respectively stand for *Relevance* and *Surprise*. These functions compare keyword sets for each Candidate *C* with the keyword set for Event *E*. *Match* calculates a similarity measure of the input sets, i.e. keywords appearing in the intersection  $\cap$  of these sets:

$$\text{Match} = C \cap E \quad (2)$$

The output is the number of intersection elements of *E* and *C*.

*Mismatch* calculates the sets' dissimilarity, viz. a symmetric difference  $\Delta$  between *E* and *C*. It is the union  $\cup$  of the relative complements of these sets:

$$\text{Mismatch} = C \Delta E = (C - E) \cup (E - C) \quad (3)$$

The final formula is normalized by a factor *NormF* to assure results independence of set sizes:

$$\text{Interestingness} = \frac{\text{Match} * \text{Mismatch}}{\text{NormF}} \quad (4)$$

### B. Characterization of Events and Candidates

The keyword sets characterizing an event are obtained from its Call-for-Papers after filtering stopwords, i.e. frequent words such as conjunctions and articles, "and", "the", which are not domain specific. Candidate characterization is similarly obtained, and schematically seen in Fig. 1.

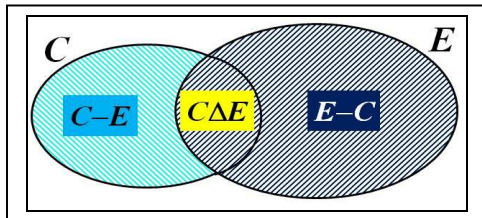


Figure 1. Schematic Match and Mismatch diagram – *E* is the Event set (dark blue). *C* is the candidate set (light blue). Match is the intersection *CΔE* (in yellow). Mismatch is the union between the relative complements *C-E* and *E-C*.

## IV. THE AUTOFOCUS SYSTEM SOFTWARE ARCHITECTURE

The AutoFocus software tool aims to automatically test the focus on targets within an event by computing *Interestingness*.

### A. Overall Experimental Client-Server System

The experimental system client-server architecture (in Fig. 2) enables server interaction with the remote user agent through the Web. The server interacts through a Restful API with the AutoFocus tool, each having its own database.

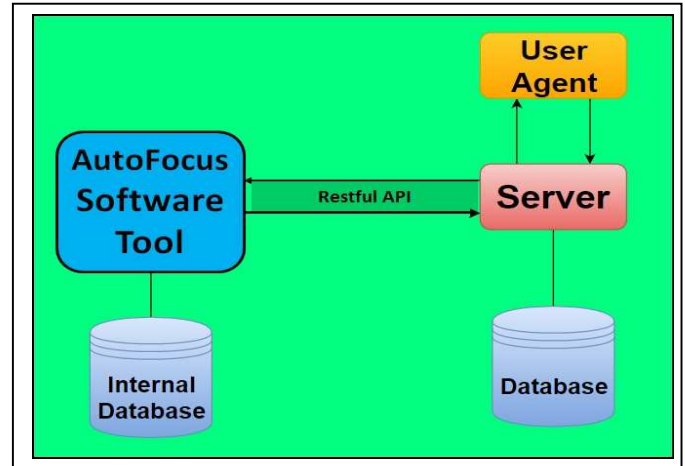


Figure 2. Overall Experimental system client-server Software Architecture – This system has three components: a *User client agent* and its *Server* and the system core, the *AutoFocus tool*. Both the *Server* and the *AutoFocus tool* have their own *Database*, and they communicate through a *Restful API*.

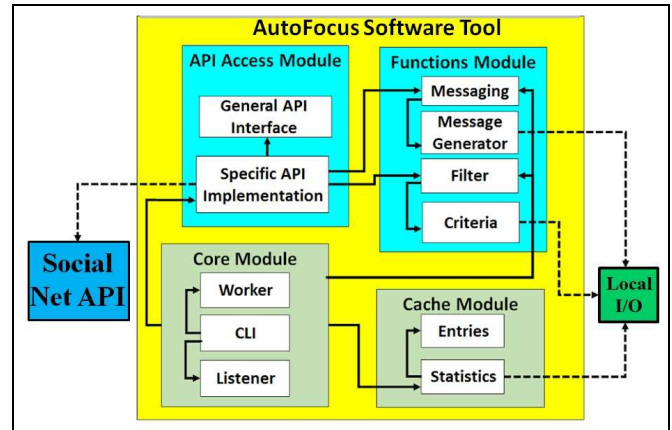


Figure 3. AutoFocus Software tool Architecture – The main sub-system (in yellow) has four modules. Two infrastructure modules: Core and Cache; two essential modules: generic API Access and the Functions module. Outside the main sub-system we emphasize the Social Net API for any specific Social Net.

### B. AutoFocus Tool Generic Architecture

The architecture goal is to clearly separate a generic API interface from any specific social net API. Assuming that social networks have much in common, one replaces any specific social net attached to the AutoFocus tool by any other one, with minimal or no modification of the generic API interface. The generic Functions Module, is also usable with any social net.

The AutoFocus architecture, in Fig. 3, is composed of infrastructure (Core and Cache Modules) and essential

functionalities (General API to access any social network and Functions Modules). The AutoFocus tool is implemented in Java. The Interestingness calculator is programmed in Python.

### C. Automation Criteria

Messages are automatically sent to social net members by some basic criteria: a) **ground frequency** (e.g. once in 24 hours) upon which actual communication is randomized; b) **random latency** with a lesser order of magnitude than the ground frequency (e.g. order of minutes); c) **message variation** specific message contents are sent only once to each target.

## V. RESULTS AND DISCUSSION

Starting from a small initial candidate set, the AutoFocus tool scans the net searching for new members related to previous candidates; the new members are added to the candidate list and the process continues recursively. Messages are actively sent and passively received, while calculating Interestingness values.

### A. Geographic Distribution Results

The countries distribution for a sample of AutoFocus contacts (received/sent messages) is seen in Fig. 4. These are: a) **Asia** – 8 countries, 8 contacts; b) **Europe** – 14 countries, 53 contacts; c) **North America** – Canada, USA, 34 contacts; d) **Other** – from Africa, Oceania and South America.

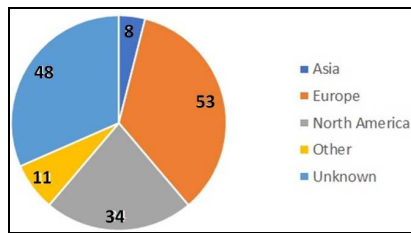


Figure 4. Geographical Distribution of Contacted Candidates – The majority of contacts are from Europe and North America. “Other” means a few countries in other continents. “Unknown” means unavailable country information.

### B. Interestingness vs. Event Site Visitors

Fig. 5 shows statistics for a larger sample of 959 candidates, “visitors” (those that visited the Event Site) vs. “non-visitors”.

	Visitors	Non-Visitors
<b>Candidate Percentage</b>	<b>9.59%</b>	<b>90.41%</b>
<b>Average Interestingness</b>	<b>0.255</b>	<b>0.154</b>
<b>Average Matched_Only</b>	<b>0.087</b>	<b>0.053</b>

Figure 5. Candidate Statistics for Visitors vs. Non-Visitors.

### C. The Importance of Surprise in Interestingness

Preliminary conclusions from the above results are:

- a- A significant number of candidates visited the Event Web site. Their average *Interestingness* is clearly higher than the average of those that did not visit the Event site;

- b- The average *Matched\_Only* does not distinguish so-well visitors from non-visitor candidates: *Surprise Interestingness* is significant.

Typical search techniques look for similarities with a pattern. The Relevance function does exactly this. This is feasible with a comparison standard. But, when searching something potentially interesting, and not sure about its existence, or without an available standard, using Relevance alone is infeasible.

The importance of *Interestingness* for searching – either in the Web or in other large data depositories – and for focusing on candidates for a certain Event resides in the *Surprise* function. This work’s experiment points out to the feature, that even when a standard ruler is available, Interestingness – including Surprise – affords advantages in order to focus on suitable items.

### D. Future Work & Main Contribution

Future work includes: time-axis distribution; measuring larger samples; usage of other Interestingness expressions such as Tfidf; more precise statistical criteria for analysis; experiments with other events.

The main contribution of this paper, besides AutoFocus tool generic development, is to test the independently computed Interestingness as a criterion to focus on candidates with real interest in a certain event, viz. Event Web site *visitor* candidates.

## REFERENCES

- [1] Z. Chu, S. Gianvecchio, H. Wang and S. Jajodia, “Who is Tweeting on Twitter: Human, Bot or Cyborg?”, in Proc. ACSAC’10, 26<sup>th</sup> Annual Computer Security Applications Conf. pp. 21-30, 2010.
- [2] I. Exman, “Interestingness – A Unifying Paradigm – Bipolar Function Composition”, in Proc. KDIR Int. Conf. on Knowledge Discovery and Information Retrieval, pp. 196-201, 2009.
- [3] I. Exman, N. Alfassi and S. Cohen, “Semantics of Social Network Frequencies for Turing Test Immunity”, in Proc. SKY’2012 Int. Workshop on Software Knowledge, pp. 79-84, 2012. DOI:
- [4] I. Exman, G. Amar and R. Shaltiel, R., “The Interestingness Tool for Search in the Web”, in Proc. SKY’2012 Int. Workshop on Software Knowledge, pp. 54-63, 2012.
- [5] E. Ferrara, O. Varol, C. Davis, F. Menczer and A. Flammini, “The rise of social bots”, Comm. ACM, Vol. 59, pp. 96-104, 2016.
- [6] L. Geng and H.J. Hamilton, “Interestingness Measures for Data Mining: A Survey”, ACM Computing Surveys, Vol. 38, (3), Article 9, 2006.
- [7] L. Gentry and R. Calantone, “A comparison of three models to explain shop-bot use on the web”, Psychology and Marketing, Vol. 19, pp. 945-956, 2012.
- [8] Z. Gilani, R. Farahbakhsh, G. Tyson, L. Wang and J. Crowcroft, “An in-depth characterization of Bots and Humans on Twitter”, 2017.
- [9] W. Klossgen and J.M. Zytow, (eds.), *Handbook of Data Mining and Knowledge Discovery*, Oxford University Press, Oxford, UK, 2002.
- [10] P. Lenca, P. Meyer, B. Vaillant and S. Lallich, “On selecting interestingness measures for association rules: user oriented description and multiple criteria decision aid”, European J. Operational Res., Vol. 183, pp. 610-626, 2008.
- [11] K. McGarry, “A survey of interestingness measures for knowledge discovery”, Knowledge Engineering Review J., 20 (1), 39-61, 2005.
- [12] M. Mowbray, “Automated Twitter Accounts”, Chapter 14 in ref. [21], pp. 183-194, 2014.
- [13] A. Tuzhilin, “Usefulness, Novelty, and Integration of Interestingness Measures”, chapter 19.2.2 in ref. [14], pp. 496-508, 2002.
- [14] K. Weller, A. Bruns, J. Burgess, M. Mahr and C. Puschmann, (eds.), *Twitter and Society*, Peter Lang Publishing, New York, NY, USA, 2014.

# Improvement of User Review Classification Using Keyword Expansion

Kazuyuki Higashi  
Graduate School of Information  
Science and Technology  
Osaka University  
Osaka, Japan  
Email: k-higasi@ist.osaka-u.ac.jp

Hiroyuki Nakagawa  
Graduate School of Information  
Science and Technology  
Osaka University  
Osaka, Japan  
Email: nakagawa@ist.osaka-u.ac.jp

Tatsuhiko Tsuchiya  
Graduate School of Information  
Science and Technology  
Osaka University  
Osaka, Japan  
Email: t-tutiya@ist.osaka-u.ac.jp

**Abstract**—Application users can submit reviews for downloaded applications. Recently, developers have received more and more user reviews. However, it is still difficult to extract beneficial comments from a large amount of reviews. Latent Dirichlet Allocation (LDA) is a promising way of topic modeling, which classifies documents according to implicit multiple topics. However, there is a gap between the documents that the developer wants to extract and the document extracted by LDA. In this paper, we propose a method to extract documents of each category, such as requirements descriptions or bug reports, more accurately. Our method first decomposes the topics. Then, the method uses the keyword list which is a set of semantically similar words collected by word2vec, to integrate the decomposed topics. We apply our method to the applications user reviews in Apple Store and demonstrate the validity of it. Our approach can help application developers to extract beneficial information.

## I. INTRODUCTION

With popularity of smartphones and tablets, mobile application stores, such as Apple Store, Google Play Store, and Windows Phone Store, have been growing. Over two million applications on the Apple Store exists, and those applications have been downloaded over 130 billion times [1].

These application stores allow users to submit reviews for downloaded applications in form of star ratings and text reviews. User reviews for downloaded applications are beneficial resources for developers because these reviews contain information, such as user requirements, bug reports, and evaluations of specific features. However, the amount of reviews is too large to extract beneficial information manually. Some popular applications receive hundreds of user reviews every day. From this background, it is needed to analyze user reviews more efficiently. In order to extract useful information from documents written in natural language, Latent Dirichlet Allocation (LDA) for topic modeling can be used. LDA is one of the most famous topic modeling techniques, which can classify the set of documents according to implicit multiple topics. When applying LDA to user reviews, LDA is good at classifying topics related to functions, such as stability and design. However, it is difficult to correctly gather requirements descriptions or bug reports in the same topic because they are

not functions. These reviews are often distributed into multiple topics. Therefore, we try to extract such crosscutting topics.

In this paper, we propose a method that allows to more accurately extract documents of crosscutting topics that developers need. Our approach is improving LDA by decomposing the topics and combining semantically similar topics. We apply this process to Facebook for iOS user reviews and compare it with the general LDA. The experimental results indicate that our approach can extract the documents more accurately than the general LDA.

This paper is organized as follows: Section II discusses related work. Section III gives the overview of our approach. Section IV gives the background of this study by providing the explanation of Latent Dirichlet Allocation (LDA). Section V explains our method of user reviews classification and keyword expansion. Section VI presents the results of experimental extraction from user reviews. Section VII discusses the feasibility of our approach, and Section VIII concludes the paper.

## II. RELATED WORK

Several studies analyzed reviews in application stores to give findings to application developers. Iacob et al. [2] evaluated user reviews about change requests and discovered that 23% of user reviews describe feature requests. Chen et al. [3] devised AR-MINER which is an approach to filtering and ranking informative reviews, and demonstrated that, on average, 35% of reviews contain informative content. These papers motivated our work because a noticeable percentage of user reviews contain useful information for developers.

Guzman et al. [4] proposed an approach that introduces rating of each function from words and user's sentiments by associating each other. Fu et al. [5] introduced a system that analyzes application reviews and identifies problems such as stability issues or cost by topic modeling. Palomba et al. [6] classified user reviews and grouped user reviews linking source code components. While these studies use Latent Dirichlet Allocation (LDA) for topic modeling and extract topics related to functions of the target application, our work, to the best of our knowledge, aims to improve LDA to accurately

classify documents according to crosscutting topics, such as requirements descriptions and bug reports.

There are some studies that combine LDA with word embedding, which is a set of feature learning techniques in Natural Language Process. Moody [7] devised lda2vec which is a model that learns word vectors jointly with Dirichlet-distributed latent document-level mixtures of topic vectors. Li et al. [8] devised TopicVec which is a generative model combining LDA and word embedding, with the aim of exploiting the word collocation patterns both at the level of the local context and the global document. For sets of short sentences such as user reviews, feature learning is difficult. Therefore, these models are not suitable for the purpose of this study.

### III. BACKGROUND

Latent Dirichlet Allocation (LDA) [9][10] is a topic model, which can classify documents written in natural language. LDA can be applied to various subjects such as news articles, feedback comments, and microblogging. In LDA, we assume that words are generated by topics and that those topics are mixed within a document. Figure 1 represents the graphical model of LDA. The variables in the figure correspond to the following concepts:

- $\alpha$ : hyper parameter about topic distribution
- $\theta$ : topic distribution for document
- $z$ : topic for word
- $\beta$ : hyper parameter about word distribution
- $\phi$ : word distribution for topic
- $w$ : word
- $K$ : the number of topics
- $M$ : the number of documents
- $N$ : the number of words in  $m$ -th document

Hyper parameter  $\alpha$  determines the topic distribution for document  $\theta$ , and the topic  $z$  is determined according to  $\theta$ . Another hyper parameter  $\beta$  determines the word distribution for topic  $\phi$ . Finally, the word  $w$  is determined according to  $z$  and  $\phi$ .

In order to construct the topic model that can classify the user reviews, LDA is used according to the following steps:

- **Step 0 (Preparation):** Give a set of documents ( $M$  and  $N$  are determined) and set the number of topics  $K$ .
- **Step 1:** Set a default topic for each word in all documents.
- **Step 2:** Select each word  $w$  from the documents.
- **Step 3:** Change the topic  $z$  for the word  $w$  according to the probability  $P$  shown in Eq. (1).
- **Step 4:** Repeat 2 and 3 until  $N_t^-$  and  $N_{mt}^-$  in Eq. (1) are converged.
- **Step 5:** Output  $\phi$  as the word distribution for topic and  $\theta$  as the topic distribution for a document.

$$P(z = t | Z^-, W, \alpha, \beta) \propto \frac{\beta + N_{tw}^-}{\beta V + N_t^-} (\alpha_k + N_{mt}^-) \quad (1)$$

where

- $Z^-$ : set of topics of all words excluding the word  $w$ .
- $W$ : set of all words in all documents.

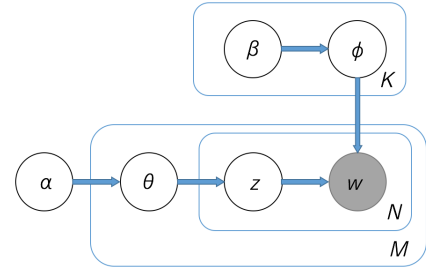


Fig. 1. Graphical model representation of LDA.

$N_t^-$ : the number of words in all documents whose topics are  $t$ .

$N_{tw}^-$ : the number of word  $w$  in all documents whose topic is  $t$ .

$N_{mt}^-$ : the number of words in selected document  $m$  whose topics are  $t$ .

$\alpha_k$ : the  $k$ -th (topic  $k$ 's) parameter  $\alpha$ .

$V$ : the number of words in all documents.

Inputting the set of documents, LDA can output  $\phi$ ,  $\theta$ , and  $z$  by constructing the topic model.

### IV. APPROACH

Figure 2 shows the schematic view of our approach using an example. Each circle represents a topic. The figure shows the classification of topics with the general LDA and our method. We try to extract topics related to *bug* by the general LDA. However, if we extract one topic, there are documents related to *bug* that can not be extracted. If we extract two topics, we extract the documents unrelated to *bug*. Therefore, it is necessary to decompose the topics. By classifying topics finely as shown and extracting topics in consideration of the meaning of words, documents related to *bug* can be extracted more accurately. In order to obtain the meaning of words, we use word2vec.

Word2vec [11][12] is an unsupervised learning algorithm for learning distributed representations of words in a vector space using a neural network model. Each word in the document can be learned from surrounding words. Distributed representations of words help to improve performance in natural language processing tasks. For example, spatial distance between words describes the similarity between the words semantically and syntactically.

Training learns representations for each word  $w_t$  (the  $t$ -th word in a corpus of size  $T$ ) so as to maximize the average log likelihood Eq.(2).

$$\frac{1}{T} \sum_{t=1}^T \log p(w_t | w_{t-c}^{+c}) \quad (2)$$

$c$  is the size of the training context (window size).  $w_{t-c}^{+c}$  is the set of words in the window of size  $c$  centered at  $w_t$ . Continuous bag-of-words (CBOW) architecture predicts the current word based on the context. CBOW defines  $w_{t-c}^{+c}$  as Eq.(3).

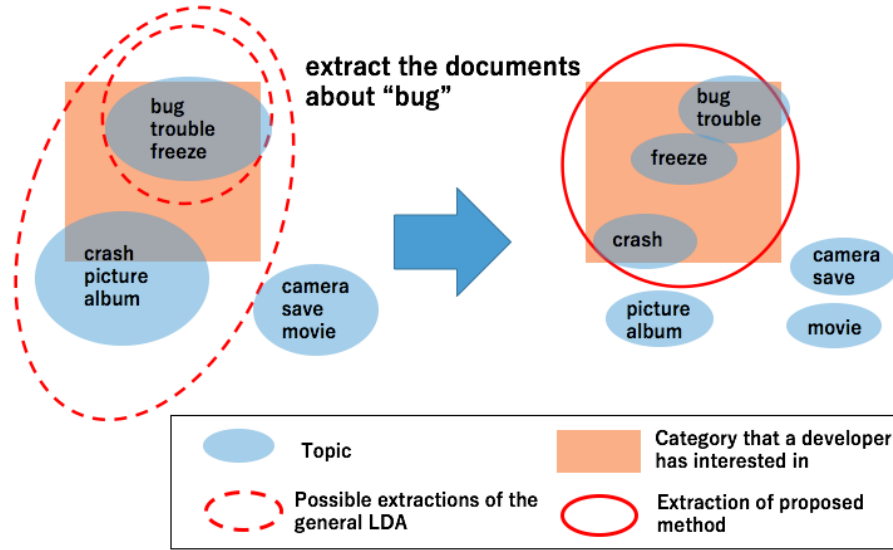


Fig. 2. Our approach is based on the decomposition of topics and binding decomposed topics using keyword expansion

$$p(w_t | w_{t-c}^{t+c}) = \frac{\exp(e'_w{}^T \cdot \sum_{-c \leq j \leq c, j \neq 0} e_{w+j})}{\sum_w \exp(e'_w{}^T \cdot \sum_{-c \leq j \leq c, j \neq 0} e_{w+j})} \quad (3)$$

where  $e_w$  and  $e'_w$  are the input and output vector representations of  $w$ .

## V. REVIEW CLASSIFICATION USING KEYWORD EXPANSION

In this section, we explain our method to classify reviews more accurately by using topic modeling. First, we collect the user reviews for a specific application and extract the title and text body from each review. Then, we preprocess the text data to remove the noise for topic modeling. Afterwards, we classify user reviews based on LDA.

We, in particular, classify the reviews more finely than in the general LDA. We can fragment reviews by setting larger value to the parameter *topic size* of LDA. In order to obtain the meaning of words, we collect semantically similar words by word2vec. We call this process keyword expansion. Finally, we select topics for each category we want to extract by using keyword list which is a set of semantically similar words.

Algorithm 1 explains the details of our method. We input the topic size, i.e., the number of topics to be generated, and keywords to the algorithm. The algorithm outputs documents belonging to the categories that the given keywords specify.

### A. Topic Modeling Using LDA

We use Latent Dirichlet Allocation (LDA) to classify user reviews. LDA is one of the most widely used topic models, to classify user reviews.

We use MALLET [13], a tool package for the topic modeling based on LDA. This tool also has the word tokenization and unnecessary word removal functions. We input user reviews to MALLET, and MALLET outputs following results by constructing the topic model based on LDA:

---

### Algorithm 1 Review Classification Using Keyword Expansion

---

```

input the number of topics larger than the general LDA
topic classification
for  $n$  in prepared keyword list do
  for top words in topics do
    calculate similarity between each keyword and top word
  end for
  sum similarity of each word
end for
add the top words of similarity to the keyword list
for topics do
  calculate the number of occurrences of the keyword list
   $O =$  sum the number of occurrences
  if  $O \geq$  threshold then
    output the documents in the topic
  end if
end for

```

---

- topic distribution for each review
- topic for each word
- word distribution for each topic (top words for each topic)

As shown in Figure 2, we intentionally obtain further segmentalized topics than those of the general LDA. For this purpose, we use a larger topic size than the size that we use in the general LDA modeling. As a result, we can obtain a number of smaller size topics than those of the general LDA modeling.

### B. Keyword Expansion

After the review classification, our method collects topics that meet with categories, which a developer is interested in. First, the developer prepares keywords that represent categories. Our method finds similar words to the given keywords.

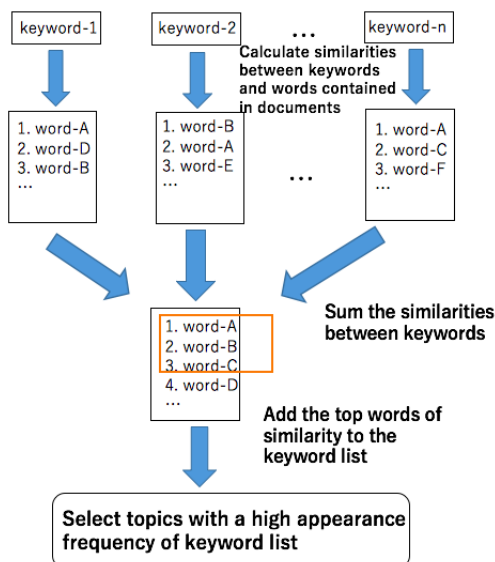


Fig. 3. Keyword expansion process

To obtain such similar words, our method uses word2vec, which can generate word vectors. By using word2vec, it is possible to calculate similarity between words and acquired similar words. Training data is full text of English Wikipedia [14] because the source covers various fields. After learning word vectors, when we input two words, the similarity between the words can be calculated. We use gensim [15], which is a package for python to learn word2vec model.

Figure 3 shows our keyword expansion process. When we find topics that belong to each category, we use keyword list obtained by keyword expansion. LDA can output topics for words and the number of occurrences of each word in each topic. We calculate the number of occurrences in each topic of the keyword list respectively. Then, the number of occurrences of each keyword is summed up for each topic, and topics whose sum is equal to or larger than the threshold are selected. Finally, we extract documents belonging to the selected topics.

## VI. EXPERIMENT

We compared our method with the general LDA to evaluate it. We applied our method to Facebook user reviews, and extracted documents of three categories: *requirements*, *bug*, *resource*. *Requirements* and *bug* are crosscutting categories, and *resource* is one of the general functions. *Requirements* type review is a review that requests changes or improvement of specific features, and desires to add a new function. *Bug* type review is a review about application problems, stability issues, bugs of specific function. *Resource* type review is a review about the effect of application use on hardware, such as battery consumption and memory usage.

### A. Dataset

We applied our approach to application user reviews to evaluate it. Apple Store is a well-known review platform. In

Title: Works, but need to stop turning off Music  
Body: While listening to music using the default music app on iOS 9.0.1 opening Facebook will pause the music no matter if the music is playing over the Built in Speaker, Headphones, or over Air-Play to an Apple TV. Please Fix. Very annoying.

Title: Causing my iPhone to freeze  
Body: App freeze When watching videos or scrolling down though my feeds. While frozen my iPhone power button won't work and once the time out causes ur screen to go to sleep mode, u won't be able to turn it on till about 3-4 mins. Anyone having this issue?

Title: Battery killer  
Body: This app is destroying my battery. Even with background refresh off and location off it still plows through battery with background activity.

Fig. 4. User reviews of each category

this experiment, we used 1200 user reviews of Facebook for iOS in Apple Store from August 5 to September 8, 2015. The reviews in Apple Store are composed of five parts: title, rating, author, date, and body text. We extracted titles and body texts from the reviews. The reviews shown in Figure 4 are reviews for Facebook. The first one is about *requirements* that the user wants to stop turning off music. The second one is about *bug* that freezes the device when using the application. The third one is about *resource*, claiming that the application drains battery with background activity. Among 1200 reviews, 270 reviews should be classified into the *requirements* category, 447 reviews should be classified into the *bug* category, 40 reviews should be classified into the *resource* category.

### B. Preparation

In order to classify documents more accurately, we pre-processed the documents. English documents contain very common words (e.g., “a”, “for”, “is”, and “that”), which are noisy to NLP activities. We removed these words as stop words from the documents. We used stop word list of MALLET, and modified it. Takahashi et al. [16] demonstrate that it is possible to make topics related to requirements likely to appear by removing several words related to the requirements from stop word list. Figure 5 represents the words that we excluded from the stop word list.

### C. User reviews classification process

First, we used the general LDA as the baseline method. In this method, we constructed topic models under the conditions that the number of topic is 20. We prepared keywords which

TABLE I  
KEYWORD EXPANSION RESULTS : TOP SIMILAR WORDS AND SIMILARITY FOR EACH CATEGORY

requirements		bug		resource	
want	1.41598253025	drop	0.910286822769	devices	1.07031276398
wish	1.34129323342	fix	0.880712643248	functionality	1.0634093351
hopefully	1.10542239509	glitches	0.826809114425	storage	1.0331266395
try	0.965955479289	overload	0.8121774242	cache	0.965408862186
help	0.944442417494	kill	0.76876851121	user	0.942083889095
sigh	0.920252507561	reset	0.765789424058	load	0.892174335253
feel	0.902822493972	trouble	0.763711373632	function	0.872954556545
will	0.883233011125	stuck	0.722607486066	software	0.836354576718

able, appropriate, appreciate, asking, ask, awfully, because, better, best, cannot, can, contains, containing, contain, considering, consider, currently, could, different, enough, except, help, hopefully, if, like, need, needs, necessary, new, normally, please, shall, should, toward, towards, tries, trying, try, unfortunately, useful, want, wants, will, why, would

Fig. 5. The words removed from stop word list.

are related to category. Topics are selected by keywords. The following keywords are prepared for this experiment.

- **requirements** : please, need, hope
- **bug** : bug, crash, freeze
- **resource** : battery, memory, data

We selected topics if these keywords are in the top words of each topic.

In our method, we applied LDA under the conditions that the number of topic is 40 to break down the topic more. Then, we expanded keywords by using word2vec of gensim.

We constructed word2vec model under the conditions that learning model is CBOW, the dimensions of the vectors is 400, the size of window is 5, and other conditions are default of gensim.

We calculated similarity between each keyword mentioned above and the top 20 words of each topic. The similarities of the three keywords are summed, and the top three words are added as new keywords. Table I lists the results of keyword expansion. After applying word2vec to the user reviews, we acquired the top similar words and the similarity between these words and the prepared keywords for each category. We acquired the following words by keyword expansion.

- **requirements** : want, wish, hopefully
- **bug** : drop, fix, glitches
- **resource** : devices, functionality, storage

Topics for each category are selected based on the number of appearances in each topic of the keywords in the keyword list obtained by the keyword expansion. Total number of appearances in each topic of all keywords is summed up and top topics are selected. If the total number of appearances was

less than 30, the topic was not selected. The number 30 was decided based on preliminary experiments.

#### D. Experimental results

Table II lists the results of applying our method. In order to evaluate the effectiveness of the proposed method, we check whether the extracted documents of each category actually correspond to the content of that category. We investigated the total number of documents in each category manually. After applying our approach, we counted the number of documents of extracted topics and the number of documents belonging to each category, out of the documents of the extracted topics. In order to evaluate our method, we used precision, recall, and F-measure as metrics. Precision is the rate of retrieved documents that are relevant to the category, and recall is the rate of the relevant documents that are successfully retrieved. F-measure is the harmonic average of the precision and recall.

Table II demonstrates that our method improves F-measure for all categories compared with the general LDA, especially *requirements* category.

## VII. DISCUSSION

In this section, we discuss our results and describe the limitations and threats to validity.

Our method improves F-measure for all categories compared with the general LDA, especially *requirements* category. User reviews representing *requirements* have many feature words, such as, "please", "want", "hope", "wish", and "need", and sometimes there is no specific word, such as, "Add the new button to save a picture!". Therefore, it is difficult to gather these reviews on the same topic. In the general LDA, unrelated documents are often mixed in the topic representing the *requirements*, resulting in a problem that precision is low. However, in our method, by preparing many keywords, it is possible to acquire the reviews of *requirements* more accurately. Our method also improves the F-measure of *bug* category. Since feature words representing *bug* are limited, precision and recall are higher than *requirements*. Our method improves the F-measure of *requirements* category slightly compared with the other two categories. There are few words representing *resource*, such as battery and memory, and documents tend to gather on the same topic.

Based on these results, our method is effective to extract crosscutting categories, such as *requirements* and *bug*. On the



TABLE II

DOCUMENT EXTRACTION RESULTS : #TOTAL NUMBER OF DOCUMENTS IN EACH CATEGORY (# $D$ ), #THE NUMBER OF EXTRACTED TOPICS(# $T$ ), #THE NUMBER OF DOCUMENTS OF EXTRACTED TOPICS (# $D_t$ ), #THE NUMBER OF DOCUMENTS BELONGING TO EACH CATEGORY OUT OF DOCUMENTS OF EXTRACTED TOPICS (# $D_{tc}$ ),  $Precision = D_{tc}/D_t$ ,  $Recall = D_{tc}/D$ ,  $F - measure = 2 * Precision * Recall / (Precision + Recall)$

		# $D$	# $T$	# $D_t$	# $D_{tc}$	Precision	Recall	F-measure
requirements	general LDA	270	4	350	117	0.334	0.433	0.377
	our method		6	265	122	0.460	0.452	0.456
bug	general LDA	447	4	389	272	0.699	0.609	0.651
	our method		8	446	311	0.697	0.696	0.697
resource	general LDA	40	1	48	26	0.542	0.650	0.591
	our method		1	34	22	0.647	0.550	0.595

other hand, we can not expect much improvement to extract the category of the general function such as *resource*.

In keyword expansion by word2vec, it is considered that similar words could be extracted with high accuracy because semantically similar words have higher similarity. However, there are still semantically similar words that we cannot extract. In order to expand keyword, we could extract more general similar words by using full text of English Wikipedia as training data of word2vec. By using user reviews itself as training data of word2vec, there is a possibility that more similar words in reviews can be extracted.

### VIII. CONCLUSIONS

In this paper, we proposed a method for extracting documents with contents required by developers with high accuracy. This method is based on topic classification by LDA and keyword expansion with word2vec. We applied this method to user reviews of a well-known application. Our experimental results demonstrated the validity of our method.

The results presented in this paper indicate some possible directions of further work and improvements. One of our primary on-going studies is further improvement of precision and recall rate of our extraction. We would like to establish a method for determining the appropriate number of topics. We also plan to define guidelines for keyword expansion. We will also conduct case studies using a large amount of user reviews to discover further findings for the extraction. We believe that automatic and sophisticated keyword expansion and topic classification help us to analyze user reviews efficiently.

### REFERENCES

- [1] "Techcrunch," <https://techcrunch.com/2016/06/13/apples-app-store-hits-2m-apps-130b-downloads-50b-paid-to-developers/>.
- [2] C. Jacob and R. Harrison, "Retrieving and analyzing mobile apps feature requests from online reviews," in *Proceedings of the 10th Working Conference on Mining Software Repositories*, ser. MSR '13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 41–44. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2487085.2487094>
- [4] E. Guzman and W. Maalej, "How do users like this feature? A fine grained sentiment analysis of app reviews," in *IEEE 22nd International Requirements Engineering Conference, RE 2014, Karlskrona, Sweden, August 25-29, 2014*, 2014, pp. 153–162. [Online]. Available: <https://doi.org/10.1109/RE.2014.6912257>
- [3] N. Chen, J. Lin, S. C. H. Hoi, X. Xiao, and B. Zhang, "Arminer: Mining informative reviews for developers from mobile app marketplace," in *Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE 2014. New York, NY, USA: ACM, 2014, pp. 767–778. [Online]. Available: <http://doi.acm.org/10.1145/2568225.2568263>
- [5] B. Fu, J. Lin, L. Li, C. Faloutsos, J. Hong, and N. Sadeh, "Why people hate your app: Making sense of user feedback in a mobile app store," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '13. New York, NY, USA: ACM, 2013, pp. 1276–1284. [Online]. Available: <http://doi.acm.org/10.1145/2487575.2488202>
- [6] F. Palomba, P. Salza, A. Ciurumelea, S. Panichella, H. Gall, F. Ferrucci, and A. De Lucia, "Recommending and localizing change requests for mobile apps based on user reviews," in *Proceedings of the 39th International Conference on Software Engineering*, ser. ICSE '17. Piscataway, NJ, USA: IEEE Press, 2017, pp. 106–117. [Online]. Available: <https://doi.org/10.1109/ICSE.2017.18>
- [7] C. E. Moody, "Mixing dirichlet topic models and word embeddings to make lda2vec," coRR, 2016.
- [8] S. Li, T.-S. Chua, J. Zhu, and C. Miao, "Generative topic embedding: a continuous representation of documents," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2016, pp. 666–675. [Online]. Available: <http://www.aclweb.org/anthology/P16-1063>
- [9] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Mar. 2003. [Online]. Available: <http://dl.acm.org/citation.cfm?id=944919.944937>
- [10] I. Porteous, D. Newman, A. Ihler, A. Asuncion, P. Smyth, and M. Welling, "Fast collapsed gibbs sampling for latent dirichlet allocation," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '08. New York, NY, USA: ACM, 2008, pp. 569–577. [Online]. Available: <http://doi.acm.org/10.1145/1401890.1401960>
- [11] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 3111–3119. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2999792.2999959>
- [12] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," coRR, 2013. <http://arxiv.org/abs/1301.3781>.
- [13] "Mallet homepage," <http://mallet.cs.umass.edu/>.
- [14] "Training word2vec model on english wikipedia by gensim," <http://textminingonline.com/training-word2vec-model-on-english-wikipedia-by-gensim>.
- [15] "gensim: models.word2vec-deep learning with word2vec," <https://radimrehurek.com/gensim/models/word2vec.html>.
- [16] H. Takahashi, H. Nakagawa, and T. Tsuchiya, "Towards automatic requirements elicitation from feedback comments: Extracting requirements topics using LDA," in *The 27th International Conference on Software Engineering and Knowledge Engineering, SEKE 2015, Wyndham Pittsburgh University Center, Pittsburgh, PA, USA, July 6-8, 2015*, 2015, pp. 489–494. [Online]. Available: <https://doi.org/10.18293/SEKE2015-103>

# A New Scheme for Citation Classification based on Convolutional Neural Networks

Khadija Bakhti<sup>1</sup>, Zhendong Niu<sup>1,2</sup>, Ally S. Nyamawe<sup>1</sup>

<sup>1</sup>*School of Computer Science and Technology*

*Beijing Institute of Technology*

Beijing, China

<sup>2</sup>*School of Computing and Information, University of Pittsburgh, Pittsburgh, PA, USA*

(bakhti.khadidja, zniu)@bit.edu.cn, nyamawe@udom.ac.tz

**Abstract**—Automated classification of citation function in scientific text is a new emerging research topic inspired by traditional citation analysis in applied linguistic and scientometric fields. The aim is to classify citations in scholarly publication in order to identify author’s purpose or motivation for quoting or citing a particular paper. Several citation schemes have been proposed to classify the citations into different functions. However, it is extremely challenging to find standard scheme to classify citations, and some of the proposed schemes have similar functions. Moreover, most of previous studies mainly used classical machine learning methods such as support vector machine and neural networks with a number of manually created features. These features are incomplete and suffer from time-consuming and error prone weakness. To address these problems, we present a new citation scheme with less functions and propose a deep learning model for classification. The citation sentences and author’s information were fed to convolutional neural networks to build citation and author representations. A corpus was built using the proposed scheme and a number of experiments were carried out to assess the model. Experimental results have shown that the proposed approach outperforms the existing methods in term of accuracy, precision and recall.

**Index Terms**—Citation Annotation, Citation Scheme, Deep Neural Networks, Citation Function Classification, Convolutional Neural Networks.

## I. INTRODUCTION

In the previous published research works the citation is categorized as a tool to calculate impact factor with an objective to know how the citation is used [1], [2]. Citation function classification is defined as the reason or motivation that why the authors cite others works in their literature, and the field of research concerned with classifying citations into classes based on the purpose behind the citations. Classification of citations could provide precise representation of the influence or the impact of a publication. For example, by considering only citations that are important to the citing paper and discarding citations that are perfunctory. The first step in citation function classification includes selecting a number of functions that citations can be categorized into, which is called a citation function classification scheme. When a scheme has been selected, a classification method is used to carry out the classification of citations.

Several citation function classification schemes have been created with a different number of functions and levels of granularity [3]. For example, [4] established a citation

scheme containing four dimensions with two functions in each dimension. Each dimension groups two related classes together; a citation can belong to one class only from one or more dimensions. Different names are used in the literature to represent specific purposes for citations such as “category”, “class”, “type”, “reason” and “facet”. We refer to the different names throughout the paper by the word “function”.

Manual citation function classification has been proposed, but subsequently automated classification became inevitable due to the large number of publications produced on a daily basis [3]. Automated citation function classification has been carried in the literature into two ways; the first way is the use of rule based methods where domain experts developed rules that were coded into computer programs to perform citation function classification [5]. The rules were created based on a set of human labeled citations where each citation was labeled with a function or label revealing the related purpose. The second way involves applying supervised machine learning techniques [6] where a set of citations were labeled by human annotators to build the training phase.

Previous studies on automated citation function classification commonly used rule-based and supervised machine learning methods [5], [7]. However, the rule-based techniques do not generalize well for citations that have never been seen by the domain experts. Therefore, multiple schemes have been proposed with different granularity varies from 35 to 3 functions [8]. However, there is no standard scheme established for citation function. Therefore, there is no way that a scheme can allow authors to frame their citations and how this framing can influence the use by future citers. [7] proposed a citation scheme for classifying citation function into six functions namely based on/supply, useful, weakness, contrast, acknowledge, hedges. [9] proposed a new scheme to annotate the citations which has seven functions: background, motivation, uses, extension, continuation, comparison, and future. Regarding these proposed schemes, there is no defined standard for citation classification schemes. However, the majority of functions like based on/useful and uses/extension have the same purpose; and similarities exist between functions could be difficult for an annotator to differentiate them for the future use. Moreover the usability of the functions proposed is limited and cannot be adopted for all the annotators from

different domains. Therefore, in this context we unify the functional common roles in several classifications and grouped together all similar functions in some categories which reflect the particular reason or motivation a citation is serving in the discourse. By focusing on the dimension of organic or perfunctory citations following [4] scheme, we divide the citations into five general functions. For this initial study, we have limited to use only the top level functions. We propose these functions mainly due to the following reasons: first of all, these proposed functions cover the most general and mutually exclusive citation functions for different domains; one could facilitate the annotation because it will be easy for annotators to have these functions separated and easily to use them later on. Secondly, it is easy to depict a typical scientific publication based on citations from these functions. Thus, our proposed strategy is valuable for the construction of further detailed citation function classification models with more refined functions. The proposed functions are: *Useful*, *Correct/Weakness*, *Contrast*, *Mathematical* and *Neutral*. The proposed approach can solve the limitations of supervised learning approaches such as the incomparable citation schemes used to label the training sets provided to the supervised algorithms and the high cost of annotating the training sets by humans.

Regarding the success of neural network methods in document classification models, [10] proposed a model for designing features from words representation in neural network. The citation function classification has recently become an active field to design new features for citation function classification based on neural network methods by identifying author's reasons for citing the literature. A deep learning approach based on convolutional neural network (CNN) algorithm with a specific layer for author information is proposed in our case to learn author's information (including author-id and name) for embedding word vector in the input layer. We will demonstrate that the proposed approach using CNN is able to solve the feature selection and representation issues automatically and achieve better results compared to other existing methods in citation function classification task. The model is tested on our corpus based on the proposed scheme and the output vector is classified into the proposed five functions.

The paper is structured as follows: section II responds to the state of the art on citation function classification. In section III, the proposed methodology is presented. In section IV and V, experiments and results are discussed. Finally, a conclusion and future work are drawn in section VI.

## II. STATE OF THE ART ON CITATION FUNCTION CLASSIFICATION

Citation function classification has been defined as the process of identifying the functions or purpose of quoting from other works [11]. In other words, it means the process of detecting the relationship between citing paper and cited paper. Many authors formulate this relationship as a citation schemes and reported several schemes to identify the

influence of cited papers on citing papers. For example, the citation scheme proposed in [12], where fifteen reasons were suggested to justify the quoting from previous work. Another citation function classification contains four dimensions which include conceptual or operational use, evolutionary, organic or perfunctory and confirmative or negation [4]. [13] reported seven argumentative zones as another classification scheme, namely: background, other, own, aim, textual, contrast, and basis according to the citation role in the author's argument. However, this scheme has some limitations, due to the large number of functions used which is time-consuming for citation annotation and cannot process a large data set of documents.

The automatic classification of citation functions can handle the schemes limitations. [14] used rule-based approach with cue words to reduce the citation functions into three categories: reference type B, C, and O. Another classification of the citation was proposed by [3] where they chose 116 articles in a random way from the Computation and Language E-Print Archive and classified twelve citation functions into four categories. In [8], authors used a semi-supervised learning method and the Naive Bayes (NB) as the main technique with features such as negation and cue words.

In [15], based on the implementation of the hybrid method algorithm, authors used discourse as their tree model and analyzed part of speech to find out citation relations regarding contrast and corroboration. [16] proposed an unsupervised bootstrapping algorithm, which led to the categorization of two concepts: application and technique. [8] used the Support Vector Machine (SVM) algorithm with linear kernel method and established a faceted classification of citation links in networks that are functional, perfunctory, and ambiguous cases. In [11], authors classified the purpose and polarity of a citation using the SVM algorithm, along with the trained classification model and linear kernel to the ACL Anthology.

Recently, citation function scheme has been created using clustering techniques found to be useful to address the annotation difficulty in previous schemes. [17] proposed an approach based on contains semantic and syntactic-based models. Their models employed multiple similarity methods to calculate the similarity between citations sentences, each cluster of similar sentences considered as a citation function.

Moreover, authors in [18] explored these issues by selecting the relevant verb in a citation sentence. They labeled each citation sentence by using semantic role labeling and then proposed six rules to extract and select the best verb that represents the citation sentence. Their rules were evaluated using four test sets and their results are reasonable. To this end, we propose here a different concept based on the deep neural networks model to address citation function classification task.

## III. METHOD

In this section, we first introduce our methodology for dataset creation and then describe the CNN based model for citation function classification. Each step will be presented below.

### A. Dataset selection

The citation corpus was built from ACL Anthology network <sup>1</sup>(ANN). ACL Anthology is an academic repository that contains full-text articles with associated meta-data. Hundred papers were chosen as a corpus size for citation sentences extraction. Our choice of these papers follows a number of previous works [6], [7]. We have used parsing rules to extract citation sentences followed by regular expressions for data cleaning, and non-citation sentences were excluded. After this step, 8700 sentences were obtained and passed as inputs for citation annotation process. As expected from previous works [7], [9], we found some citation functions were infrequent. Therefore, we attempted to build frequent functions *mathematical*, *correct*, *follow* and *neutral*, by basing on keywords used towards extracting citing citations. For example, the word statistical for the *mathematical* function. In the citation annotation process, three PhD students worked separately as annotators to manually annotate the citation sentences using the proposed scheme. The annotators not only focused on labeling word in the citation sentence, but also read the entire sentence and the whole context where the sentence is located. Then made a decision on the function of the citation and determine function of each citation by choosing from the five functions described in Table I. To test annotation reliability, we measured inter-annotator agreement between three annotators, we used  $\kappa$  coefficient as proposed in [19]. We used a small section of the corpus about 800 citations to analyze them according to their function. Inter-annotator agreement was  $\kappa = 0.76$  with parameter  $n = 5$  and  $N = 800$ . The results is quite high given the fact that Kappa value of 0.76 is considered as stable [3]. Table II illustrates the distribution of the dataset.

### B. Model architecture

Fig.1 shows our proposed model architecture based on CNN method. In this model, we have a citation sentence consisting of  $n$  words as  $\{w_1, w_2, w_3, w_4, w_i, \dots, w_n\}$  where  $w_i$  means the  $i^{th}$  word in the sentence  $\in R^d$  and the  $d$  is dimensional word vector corresponding to the word. The output of word

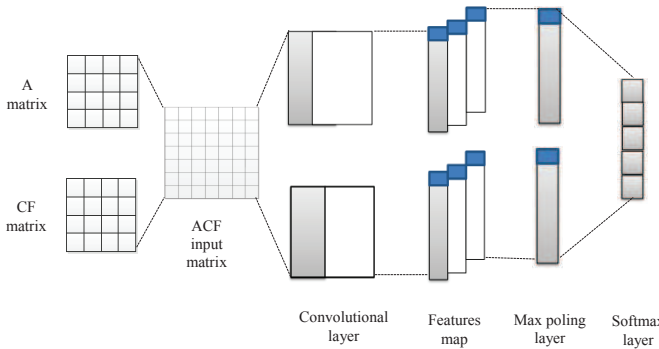


Fig. 1. Architecture of citation function classification based on CNN model.

<sup>1</sup><http://clair.eecs.umich.edu/aan/index.php>

TABLE I  
THE PROPOSED CITATION FUNCTIONS FOR CITATION CLASSIFICATION.

Functions	Description
useful	The citation sentence is classified as use if the cited work utilized or followed data, method, tools from the citing work.
contrast	Is reserved for correct of previous research, addressing by the authors such as error, weakness from cited paper.
mathematical	Describe the comparison between the cited own work and other works, the result can be positive or negative.
correct	The cited work base on tools, results, statistical tables, algorithms from citing paper.
neutral	Expression of author using own language marked as no useful interpolation or the description of specific method, concept.

TABLE II  
THE DISTRIBUTION OF THE DATASET.

Functions	The number of citation sentences	Ratio
useful	2195	24.81%
contrast	1800	20.68%
mathematical	1846	21.21%
correct	1700	19.54%
neutral	1195	13.73%
<b>Total Sentences</b>	<b>8700</b>	<b>100%</b>

vectors gets a real-valued vector known as word embedding. The word embedding is a powerful technique to capture the semantic and syntactic of words and also it could be useful to extract features from the sentence. Therefore, following [10], [20] strategy of word vectors representation, we used word2vec to build our CF matrix as depicted in Fig.1. In addition, we proposed to exploit author's information such as personal demographic data (id-author, name) to build a matrix A. The model takes as inputs the citation sentences and the author that cites the paper with relative information. Then we link all the authors with their citation sentences, which have the same function from our functions.

In our model, Author Citation Function (ACF) is a matrix that combines the representation of citation sentences and the author's information as presented in equation (1).

$$ACF = A \cdot CF \quad (1)$$

Finally, the ACF is passed to the CNN method to classify citation into our proposed functions.

In the CNN, the convolutional process consists of applying filters  $W \in R^{h \times d}$  in a window of  $h$  words in the sentence of length  $n$   $\{w_{1:h}, w_{2:h+1}, \dots, w_{n-h+1:n}\}$ . We have chosen multiple convolutional filters with varying filters window size from 3 to 5, and applying these filters using non-linear activation function (In our model we used wide Rectified Linear Unit (Relu) as the activation function, [21]) for each window of words within the citation sentence to produce a new feature  $p_i$  of size  $n - h + 1$ . A feature  $p_i$  is generated from a window of words  $w_{i:i+h-1}$ . Let consider the following ex-

ample illustrating the non-linear activation function operation given as:

$$p_i = f(W \cdot w_{i:i+h-1} + b) \quad (2)$$

where  $b \in R$  is the bias,  $f$  is the non-linear activation function. The max-pooling operation, [22] is then applied. We used max-pooling because it is widely used, and the idea is to take the maximum value  $p_{max}$  from the feature map as the most important feature among one map  $P$ .

$$p_{max} = \max\{P\} = \max\{p_1, \dots, p_{n-h+1}\} \quad (3)$$

### C. Function classification

To perform citation function classification, our classifier used citations with functions. The performance of the classifier can be affected by over-fitting problem, which could come from the weakness of the neural net. We have employed the dropout regularization to prevent over-fitting problem of the hidden units in the classifier. In the classification stage, we feed the final feature map to the softmax layer. We chose the softmax because it is commonly used for classification problem, which gives a probability of the sample belongs to each label (class). The outputs of softmax layer can be interpreted as conditional probabilities. Equation (4) shows the softmax function formula.

$$Softmax_i = \frac{e^{x_i}}{\sum_{j=1}^L e^{x_j}} \quad (4)$$

where  $L$  is number of labels (we have five functions as labelling (classes)) and  $x_j$  is the weight vector of the  $L^{th}$  label.

## IV. EXPERIMENTS

Experiments were carried out using the dataset described in section III.A in order to test our approach by applying it into classification of citation function. The results from the experiments were then compared.

### A. Experimental Setting

First, let address some hyper-parameters used. For each filter size as a window, we chose 3, 4, 5 respectively. We enable the dropout in training and disable it in evaluating the model and set 0.5 as a dropout rate. We used 10-fold cross validation for training and evaluating our model. We also apply the loss function classifier to correct and minimize errors that our network makes [23]. For evaluating the model, we calculate the accuracy using the standard accuracy formula described in [24].

### B. Baseline methods

We compared the proposed ACFNN model with the following state of the art methods for citation function classification:

- N-gram+SVM: this method uses n-gram and train classifier with SVM [7].
- Word2vec+SVM: this method considers each function as a separate feature and train classifier with SVM [9].
- Word2vec +Naive Bayes: this method uses vector and train classifier with Naive Bayes [9].

TABLE III  
ILLUSTRATION OF ACCURACY MEASURE.

The method	Acc(%)
N-gram+SVM (Hernandez)	56.8
Word2vec+SVM (Jurgens)	57.1
Word2vec+Naive bayes (Jurgens)	55.4
Cue phrases+IBK algorithm (Toufel)	52.2
CNN (no A)	58.2
ACFNN	62.7

- Cue phrases or meta-discourse: this method uses cue phrases and train classifier with IBK algorithm [3].
- CNN (no A): in this method, we remove matrix based representations (A) from ACF and train the model with CNN.

### C. Results and discussion

In this sub-section, we report the details of the experimental results as presented in the following.

1) *Accuracy measure*: The results of the experiments are presented in Table III. As output, the final vector is a fine grained classification into five functions: *useful*, *contrast*, *mathematical*, *correct* and *neutral*.

Comparing the results of the proposed model tested along with the baseline methods, the results indicate the performance calculated in accuracy (Acc) by incorporating different feature sets with the batch size of 25, obtaining 62.7% as a best performance achieved using the proposed (ACFNN) model. Comparing (ACFNN) with CNN (no A), we can clearly observe that our model achieves the highest accuracy (62.7%), followed by the CNN (no A), which has an accuracy of 58.2%. The results show that the proposed model improves the classification accuracy by 4.5%, and this illustrates that author’s information can improve the impact of the importance of their integration in the citation function process to handle the problem of citation function classification task.

The baseline’s approaches n-gram features (SVM), word vector features (SVM, Naive Bayes), and cue phrases (IBK algorithm) suggest that word2vec features with deep neural networks models (word2vec+CNN) dual an improvement to oriented methods for a better classifier for citation function. Furthermore, the capacity in exploiting information is proven by the CNN workflow by scanning the combination of words sequentially and retaining the sequential information to attract a pool operation, which can bridge the information space at both ends of the citation sentence. Thus, it is evident that the CNN can handle the problem of manual features extraction.

2) *Performance evaluation*: Experiments were conducted to evaluate the performance of the proposed ACFNN. In comparing the performance of the proposed ACFNN approach against the baseline methods, we used precision, recall and f-measure metrics. In using precision and recall evaluation metrics, labels are mapped into a binary scale (relevant versus not relevant). We also considered learning elements as “not relevant/not classified” and “relevant/classified”. The description of precision and recall metrics is shown in Table IV. Precision

TABLE IV  
PRECISION AND RECALL METRICS.

	Classified	Not Classified
Used	True Positive (tp)	False Negative (fn)
Not Used	False Positive (fp)	True Negative (tn)

is the ratio of relevant instances selected by the classifier to the number of instances selected. A learning element is considered non-relevant if the classifier ignores it.

$$Precision = \frac{\text{Correctly classified instances}}{\text{Total classified}} = \frac{tp}{tp + fp} \quad (5)$$

Recall is the ratio of relevant instances selected to the number of relevant instances.

$$Recall = \frac{\text{Correctly classified instances}}{\text{Relevant instances}} = \frac{tp}{tp + fn} \quad (6)$$

where relevant instances is the number of learning elements classified as relevant by the classifier.

F-measure is the harmonic mean of precision and recall, F-measure uses both precision and recall to correctly assess the efficacy of the classification.

$$F - \text{measure} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad (7)$$

The results of citation functions classification in 10-fold cross-validation are given in Table V. The results are conducted in three overall measures: Precision, Recall and F-measure of five functions. Precision for all the functions is above 0.58. To test the contribution and success of the proposed functions, we used Macro-F which is the mean average of F-measure of all five functions. In the case of Macro-F, regarding the reported result; we can see that the classification yield higher values in the functions such as *useful* and *mathematical* than other functions. The distribution of the citations functions as shown in Table II are: 24.81% *useful*, 20.68% *contrast*, 21.21% *mathematical*, 19.54% *correct* and 13.73% *neutral*. We found that total number of *useful* and *mathematical* citations is higher than the other citations, this empirically confirm that the authors are likely to use, follow or extend (*useful*) works from the cited works ( $p=0.59$ ) as well as they are more focusing on mathematical concept (*mathematical*) such as methods, statistical tables, results from the citing work ( $p=0.59$ ). In the *contrast* function, as we know that the authors start the state of the art with an objective (compare) the previous works. In addition to this, in the *correct*, the authors address the errors and weakness of previous works and suggest solutions to correct them, as shown in the Table V ( $p=0.58$ ). Finally, the

TABLE V  
RESULTS OF CITATION FUNCTION CLASSIFICATION.

	Precision	Recall	F-measure
useful	0.62	0.59	0.60
contrast	0.58	0.57	0.57
mathematical	0.61	0.58	0.59
correct	0.60	0.57	0.58
neutral	0.58	0.56	0.57
Macro-F = 0.58			

citations which do not belong to any of the above citations are tag as not useful description (*neutral*). The analysis of the functions indicates that there is higher negative correlation between all functions, this leads to a conclusion that these proposed functions outperform the state-of-art citation schemes described in section I (no similarities between them). Thus, it is evident that the proposed five functions can cover the most general functions and increase the performance of the classification task. Fig.2 illustrates the performance of the proposed model in terms of precision into batch size of 25 in comparison with the baseline methods. The experiment was repeated in different number of iterations.

Comparing results from Fig.2, it is observed that the proposed model provides best performance in terms of precision than the other methods with batch size of 25 regardless of the number of iterations.

It is evident that as shown in Fig.3 the proposed model outperforms the baseline methods in terms of recall. The experiment was repeated for the different number of iterations.

## V. DISCUSSION

The results conducted using our corpus have shown the effectiveness of the proposed model. We can see that our model (ACFNN) significantly outperforms existing methods such as SVM, Naive Bayes and traditional CNN in terms of accuracy, precision and recall. The reason is that since the CNN can absolutely capture the semantic content of citations and select number of features needed automatically. In addition, using the concept of word embedding, we can acquire richer features automatically. Our results suggest that word2vec+CNN concentrate on weakness of prior works and had the largest impact on the performance. With the help of the authors information in concatenation with the citations and feed them to CNN, making our model more efficient and outperforms the baseline CNN by 4.5% in accuracy. As we stated the limitations of previous works in section II, there is no standard scheme up to date and it is difficult to distinguish between the functions with close similarities. Regarding the high frequency of usage frequent functions shown in Table II, we believe that our proposed scheme can handle the problem

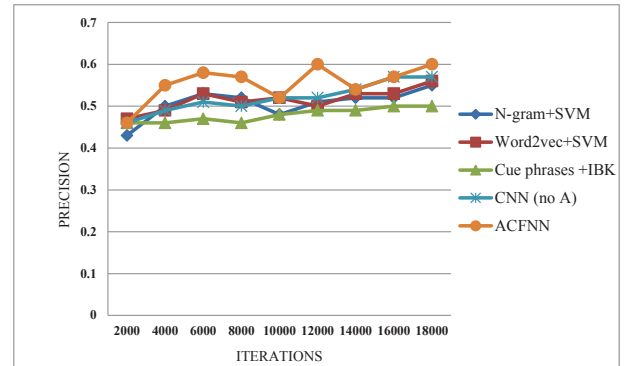


Fig. 2. The performance of the methods in term of precision.

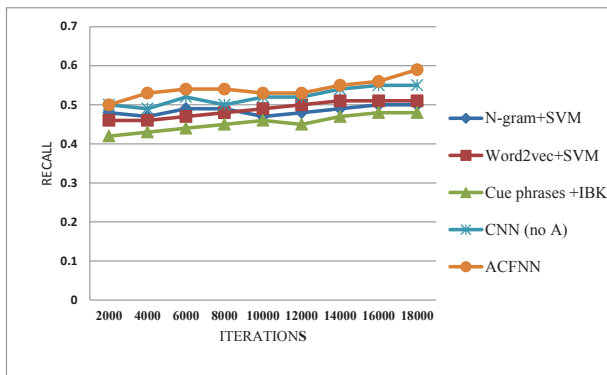


Fig. 3. The performance of the methods in term of recall.

of annotation for citation function. In addition, these functions can cover the most general and mutually exclusive citation functions for different domains. Moreover, these functions remain an important line for the future use since will be easy for the annotators to separate them later.

## VI. CONCLUSION

In this paper, we have presented an approach that uses the CNN based model combined with author's information to classify citation sentences into five functions. Experimental results show that the proposed method is able to identify authors' reasons semantically. Moreover, combining citations with authors' information achieves best performance in our corpus. Therefore, our proposed scheme is able to handle the weakness of the citations annotation and can be used in different domains. The proposed model reveals that CNN can outperform the shallow classification for citation function classification task. Valuable information can be extracted using the data citation function, which will have a real interest to help in the search of high-quality papers. Our future work is to explore other deep learning approaches such as Long Short-Term Memory Networks (LSTM) which is a type of recurrent neural network (RNN).

## ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China (No. 61370137), the Ministry of Education-China Mobile Research Foundation Project (No. 2016/2-7) and the 111 Project of Beijing Institute of Technology.

## REFERENCES

- [1] Jingqiang Chen and Hai Zhuge. Summarization of scientific documents by detecting common facts in citations. *Future Generation Comp. Syst.*, 32:246–252, 2014. 10.1016/j.future.2013.07.018.
- [2] Abdallah Yousif, Zhendong Niu, John K Tarus, and Arshad Ahmad. A survey on sentiment analysis of scientific citations. *Artificial Intelligence Review*, pages 1–34, 2017.
- [3] Simone Teufel, Advait Siddharthan, and Dan Tidhar. Automatic classification of citation function. In *EMNLP 2007, Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, 22-23 July 2006, Australia*, pages 103–110

- [4] Michael J Moravcsik and Poovanalngam Murugesan. Some results on the function and quality of citations. volume 5, pages 86–92. CA,1975.
- [5] Mohammad Abdullatif. Making the h-index more relevant: A step towards standard classes for citation classification. In *Workshops Proceedings of the 29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013*, pages 330–333, 2013. 10.1109/ICDEW.2013.6547476.
- [6] Myriam Hernández Alvarez and José M. Gómez. Citation impact categorization: For scientific literature. In *18th IEEE International Conference on Computational Science and Engineering, CSE 2015, Portugal, October 21-23, 2015*, pages 307–313.
- [7] Myriam Hernández Álvarez, José M Gómez, Patricio Martínez-Barco, et al. Annotated corpus for citation context analysis. 2016.
- [8] Han Xu, Eric Martin, and Ashesh Mahidadia. Using heterogeneous features for scientific citation classification. In *Proceedings of the 13th conference of the Pacific Association for Computational Linguistics*, 2013.
- [9] David Jurgens, Srijan Kumar, Raine Hoover, Dan McFarland, and Dan Jurafsky. Citation classification for behavioral analysis of a scientific field. 2016.
- [10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.
- [11] Amjad Abu-Jbara, Jefferson Ezra, and Dragomir R. Radev. Purpose and polarity of citation: Towards nlp-based bibliometrics. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, USA*, pages 596–606, 2013.
- [12] Eugene Garfield. Citation indexes for science. a new dimension in documentation through association of ideas. *International journal of epidemiology*, (5):1123–1127, 2006.
- [13] Simone Teufel, Jean Carletta, and Marc Moens. An annotation scheme for discourse-level argumentation in research articles. In *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*, pages 110–117. Association for Computational Linguistics, 1999.
- [14] Hidetsugu Nanba and Manabu Okumura. Towards multi-paper summarization using reference information. In *IJCAI*, pages 926–931, 1999.
- [15] Adam Meyers. Contrasting and corroborating citations in journal articles. In *RANLP*, pages 460–466, 2013.
- [16] Chen-Tse Tsai, Gourab Kundu, and Dan Roth. Concept-based analysis of scientific literature. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 1733–1738. ACM, 2013.
- [17] Mohammad Abdullatif, Yun Sing Koh, and Gillian Dobbie. Unsupervised semantic and syntactic based classification of scientific citations. In *Big Data Analytics and Knowledge Discovery - 17th International Conference, DaWaK 2015, Valencia, Spain, September 1-4, 2015, Proceedings*, pages 28–39, 2015.
- [18] Mohammad Abdullatif, Yun Sing Koh, Gillian Dobbie, and Shafiq Alam. Verb selection using semantic role labeling for citation classification. In *Proceedings of the 2013 workshop on Computational scientometrics: theory & applications*, pages 25–30. ACM, 2013.
- [19] Jean Carletta. Assessing agreement on classification tasks: the kappa statistic. *Computational linguistics*, 22:249–254, 1996.
- [20] Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL (1)*, pages 1555–1565, 2014.
- [21] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.
- [22] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 2493–2537, 2011.
- [23] Ahmad Parsian and Nader Nematollahi. Estimation of scale parameter under entropy loss function. *Journal of Statistical Planning and Inference*, 52:77–91, 1996.
- [24] Dan Jurafsky. *Speech & language processing*. Pearson Education India, 2000.

# Learning API Suggestion via Single LSTM Network with Deterministic Negative Sampling

Jinpei Yan, Yong Qi, Qifan Rao, Hui He

School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, China  
Emails: yjp2013@stu.xjtu.edu.cn, qiy@xjtu.edu.cn, asd5510@stu.xjtu.edu.cn, huihe@xjtu.edu.cn

**Abstract**—Modern programming relies on a large number of fundamental APIs, but programmers often take great effort to remember names and the usage of APIs when coding, and repeatedly search the related API documents or Q&A websites. To improve the programming efficiency, we present a Java API suggestion approach called APIHelper which learns API sequence pattern via the Long Short-Term Memory (LSTM) network, then provides API suggestion based on the program context. Previous related works use statistical methods based on Hidden Markov Model (HMM), which require establishing one specific model for each class. We propose Deterministic Negative Sampling (DNS) to make API suggestion for a large number of Java classes by one single end-to-end LSTM. To verify this approach, we make API suggestion for 50,000 Java classes and evaluate it with top-K accuracy. Results show that APIHelper outperforms other prior works both on accuracy and computation efficiency.

**Keywords**—API suggestion; Long short-term memory; Negative sampling

## I. INTRODUCTION

Modern programming languages continue to evolve and introduce more and more high level APIs/methods/functions, while increasing third-party libraries are available for programmers to use. The constantly optimized and rich APIs allow programmers to achieve demo codes easily for different kinds of requirements, but give programmers a great challenge to memorize all these APIs. Programmers often use multiple programming languages, and each programming language involves a large number of grammar rules and APIs. Some programming languages, like Java, have a large number of very complex and long API names, which may be very similar. A simple example, Java offers a lot of classes for I/O operations, such as *FileInputStream()*, *ByteArrayInputStream()*, *CharArrayReader()*, *InputStream()*, *StringReader()*, *StringWriter()*, *PrintStream()*, *PrintWriter()*, *BufferedInputStream()*, *BufferedOutputStream()*, *BufferedReader()*, *BufferedWriter()*.

For this problem, some current IDEs are integrated with API tip tools, such as the one in Eclipse (shown in Fig.1). But these kinds of tools have a common problem that only shows all API methods in the alphabetical order containing the current Java class. Since a Java class often has a lot of methods, programmers still need to look over the list to find the right one in a bunch of similar method names.

To this end, we put forward a new approach of API suggestion called APIHelper to complete APIs pattern learning for full Java classes through the Long Short-Term Memory (LSTM) network. Our approach has two advantages over



Fig. 1. Eclipse IDE tools integrated with API tips.

the statistics-based approach. First, LSTM can capture more long-term contextual patterns or relationships, resulting in a higher prediction accuracy. Second, the mainstream statistical approaches [1], [2] are based on Hidden Markov Model (HMM), which require building separate HMMs for each single Java class. Since the existing Java classes are huge and keep increasing, the scalability of HMM-based method is seriously challenged. Instead, we propose a method called Deterministic Negative Sampling (DNS) to model all Java classes and APIs in one single LSTM network, which is more flexible and scalable.

Our main contributions are summarized as follows:

- We propose a new API usage pattern learning method based on LSTM, which learns context features from Java API sequences to make API suggestion.
- We build an API suggestion approach called APIHelper. It makes use of one single LSTM network combining with a proposed DNS method for all Java classes&APIs learning and predicting.
- We collect 18,000 Java project codes from Github for the API suggestion experiment. Results show that APIHelper has a better classification accuracy and computational efficiency compared with HMM and N-gram.

## II. RELATED WORK

**API Mining/Usage Learning.** These prior works are closely associated with API suggestion. Xie et al. [3] first proposed the method to mine API usage patterns from source code called MAPO. UP-Miner [4] was proposed to improve the efficiency on API sequences mining compared to MAPO. They used a probabilistic graph to describe API sequences and introduce a set of N-gram features of API call sequences for clustering distance metric calculation. Recently, Fowkes et al. [5] proposed a tool named PAM which is a near parameter-free probabilistic algorithm to output a list of API call patterns.



PAM significantly outperforms both MAPO and UPMiner. Similar work was proposed by Nguyen et al. [6] which also used the graph to represent API sequences. They built a tool called GROUM, a vector-based approximation approach, to dig out the object usage pattern by finding the isomorphic subgraph for anomaly detection.

Other works try to use NLP techniques to represent and mine API usage patterns. For example, Nguyen et al. [7] proposed API2VEC which represents APIs with a dense vector. They came up with a series of rules to extract API sequences from Java source codes and used API2VEC to do the Java to C# code translation work for evaluation. Gu et al. [8] used natural language to represent APIs and extracted them from Javadoc annotations for code blocks to collect  $\langle API\ seq, annotation \rangle$  pairs as the dataset and mine the correspondence.

**API Suggestion/Recommendation.** Currently, these are some strategies for API suggestion/recommendation. The first strategy is focused on sequential pattern mining. Nguyen et al. [1] designed an API suggestion tool specifically for Android mobile development. They used GROUM to extract the API call graph from the Android source code, then simply traversed the graph to generate the API sequences and process it using a special HMM model for each Java class. On this basis, further improvements were made by Pham et al. [2], who analyzed directly the Android applications to find the usage for APIs. They first decompiled the Android application to get bytecodes, then transferred them into Control Flow Graph (CFG), finally built HMMs to mine API usage patterns. Raychev et al. [9] implemented a tool called SLANG for API prediction. They tried N-gram, Recurrent Neural Network (RNN) and the combination of both methods to extract the API call sequence, and used N-gram or RNN for training.

The second strategy is focused on frequent subgraph or itemset mining. Zhong et al. [10] used code search engines and code snippets to extract API call sequences. These extracted APIs are clustered according to the distance metric which reflects the similarity between class names and API names. They mined the most frequent API calls using SPAM for each cluster. Nguyen et al. [11] presented a tool named LIBSYNC to guide the developer to update APIs so as to fit the third-party library update. They used several graph-based techniques to describe changes in the APIs.

Besides, Niu et al. [12] mined API usage patterns without relying on frequent-pattern mining, but automatically extracted usage patterns by clustering the data based on the co-existence relations between object usages. Tetsuo Yamamoto [13] instead took a simply and lightweight method, the author designed a specific algorithm based on rules to give a method call suggestion according to the context.

### III. OUR METHODOLOGY

#### A. Overview

The main idea is to use the LSTM network to learn the semantic and context information from Java source codes, the embedding representation of Java classes&APIs (methods) and the context features from API call sequences, respectively. We treat API suggestion as a multi-class classification problem.

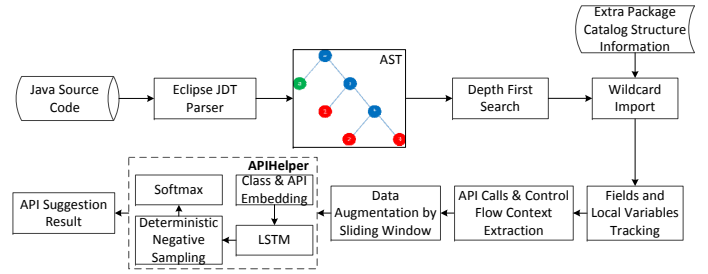


Fig. 2. The core flowchart of proposed method.

Given an API call sequence as an input, LSTM uses a softmax output layer to predict next API call and outputs probabilities of all candidate API calls. The core flowchart of our approach is shown in Fig.2. We first extract API sequences from Java source codes. Then APIHelper embeds Java classes and APIs into dense vector representations. As a result, it transfers the API sequence into a vector data stream as LSTM’s input for training. Here API suggestion can be regarded as a “predict next word” task. Given an API call sequence as one input, LSTM tries to predict next API call.

#### B. API Sequence Extraction

We mainly extract the API call information from source codes. For this information, we extract Java method invocations and Java class instance creations. Concretely, we first obtain the corresponding Abstract Syntax Tree (AST) with Eclipse JDT parser tool from Java source codes. Then we traverse the AST tree through Depth First Search (DFS) to extract nodes for Java classes and API calls, and track all fields and local variables’ method calls. Thus we can resolve the fully qualified name of an API call according to the field or local variable. In this phase, we also need to parse the import statements to get fully qualified class names to identify all Java classes explicitly imported. Note that in some codes, wildcards can be used to load all classes in a high level package path, which will cause the fully qualified name of some classes cannot be resolved. Under this situation, we can obtain the package catalog structure information to get a fully qualified path through the relevant Java documents. After that, some method names cannot be fully resolved, so we filter them in the data preprocessing phase. It should be noted that the superclass method, method override, constructor call, and class conversion expression do not take into account for simplification in our scenario.

#### C. Learning API Usage Pattern with LSTM

Programming languages have some similarities to natural languages, so here we use language models for API suggestion. For a Java API sequence, since Java is an object-oriented language and all methods are encapsulated in an object belonging to one specific class, we can define an API sequence as  $S = \{c_1, a_1, c_2, a_2, \dots, c_T, a_T\}$ , where  $c_t, t \in (1, T)$  represents the class to which the  $t$ th object belongs, and  $a_t$  represents the corresponding API called by the object. Then we can use the statistical language model to define the occurrence probability of this API sequence. According to the Bayesian algorithm,

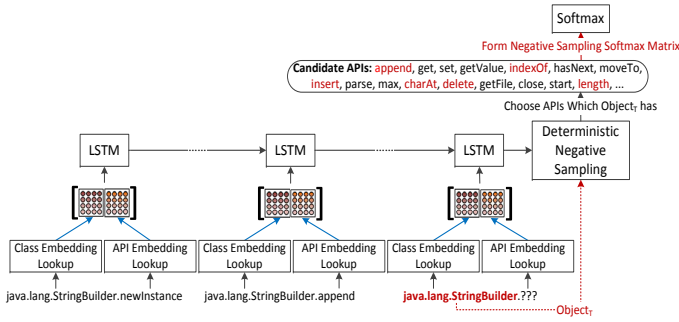


Fig. 3. The neural language model for API suggestion by the LSTM network.

the probability that occurrence probability of a sequence  $S$  equals to the combination of the probability that each “word” appears, so  $P_w(S)$  can be expanded to:

$$P_w(S) = P_w(a_1) * \prod_{t=2}^T P_w(a_t | c_t, a_{t-1}, c_{t-1}, \dots, a_1), \quad (1)$$

where  $w$  is a parameter learned by the statistical language model and updated through training model. For the API sequences in the training dataset, we maximize the occurrence probability:

$$\operatorname{argmax}_w P_w(c_1, a_1, c_2, a_2, \dots, c_t, a_t). \quad (2)$$

For the API suggestion task, we choose  $a_t$  to maximize the occurrence probability of current API sequence, and calculate the maximum likelihood probability as the objective function in the training phase which gets the highest probability when  $a_t$  equals to the true value  $y$ :

$$\operatorname{argmax}_w P_w(a_t = y | c_1, a_1, c_2, a_2, \dots, c_t). \quad (3)$$

However, the actual Java API sequence often has a long-term context, like follows:

```

FileWriter writer = new FileWriter("XX.csv");
{... a bunch of codes ...};
writer.close();

```

The bunch of codes in the middle part may be very long. Therefore, we need a way to learn longer context information to suggest API `close()`. The neural language model is one good solution which is widely used in NLP tasks in recent years. In this paper, we use LSTM to build a neural language model. It is a powerful deep neural network for temporal data mining and learning. As a variant of RNN, LSTM takes a bunch of previous API sequences as context to predict next API call, which actually gives a prediction category from all candidate API calls as classification. The overall model part that APIHelper uses for API suggestion is shown in Fig.3.

Specifically, the whole model contains three parts: embedding layer, LSTM layer, and classification layer. First, the embedding layer will embed these input API sequences into dense vectors that actually learn some semantic and logic information from Java classes and API calls. These dense vectors will be used as the input for several LSTM units to learn the context features. Then, LSTM sequentially takes the elements in the API sequence as an input. Each step, LSTM will update

Constant Error Carousel (CEC) and “gates” to selectively store information from previous inputs and calculate them with the input of current time node. In this way, after receiving the complete API sequence input, LSTM can learn the context information in the sequence to generate a dense vector for the final prediction. Then this dense vector is regarded as the highly abstract feature information to be added to the softmax output layer, which outputs the prediction result. The formula is as follows:

$$P_w(a_t = \hat{y} | c_1, a_1, c_2, a_2, \dots, c_t) = \frac{\exp(W_{\hat{y}}^T h_t + b_{\hat{y}})}{\sum_{k'=1}^K \exp(W_{k'}^T h_t + b_{k'})}, \quad (4)$$

where  $W_{\hat{y}}^T$  represents the part of weight matrix  $W$  corresponding to the prediction category  $\hat{y}$ , and  $h_t$  is the final dense vector calculated by LSTM forward propagation along the time series. The softmax layer uses a nonlinear transformation to calculate the conditional probability for outputting the final prediction result. Meanwhile, it uses maximum likelihood probability as the objective function to train model parameters.

It should be noted that, unlike dealing with natural language, we have conducted some specialized training strategies for API sequence learning. As mentioned earlier, each of Java classes follows a specific API call. A straightforward approach to digitalize these elements is directly mapping them using one same dense embedding matrix. But this approach is not good for several reasons. First of all, these two elements are not the same type. Using one embedding matrix would mix them up and lose this boundary between two elements. Second, Java language contains a large number of classes and APIs, and needs a very large embedding matrix to represent them, which would greatly slow down the training speed. And most importantly, LSTM is not effective due to the gradient vanishing when dealing with a very long API sequence. Therefore, we propose a new approach using two embedding matrixes  $W_c^e$  and  $W_a^e$  for representing Java classes and APIs,

$$e_t^c = W_c^e [c_t], e_t^a = W_a^e [a_t]. \quad (5)$$

Considering that each Java class must be followed by an API call, we come up with a strategy called Embedding Concatenation (EC), which uses  $[e_t^c, e_t^a]$  to concatenate the Java class embedding in the current timestep and the corresponding API call embedding as the input to LSTM, that is  $x_t = [e_t^c, e_t^a]$ .

Since one embedding vector can represent one Java class and one API call, and LSTM can deal with a *Class&API* unit in one timestep, the length of the API sequence that LSTM needs to process is actually reduced by nearly half. It allows LSTM to handle and learn more long-term context information and eases the gradient vanishing problem.

The last API call  $a_T$  of an API sequence is to be predicted, so the last timestep input is received by LSTM which contains only the Java class information  $c_T$ . Here we use an identifier  $e^{pred}$  to remind LSTM that it is the section to make a prediction, that is  $x_T = [e_T^c, e^{pred}]$ .

#### D. Deterministic Negative Sampling

The concept of Negative Sampling (NS) comes from a training strategy for word vectors in the NLP area. Some famous models using negative sampling are Skip-gram with

Negative Sampling (SGNS) and CBOW with Negative Sampling (CBOW-NS). The main purpose of negative sampling is to improve computational efficiency. For a multi-class classification problem, machine learning models usually use the softmax layer to output multi-class prediction probabilities:

$$P_w(a_t = \hat{y} | context) = \frac{\exp(W_{\hat{y}}^T h_t + b_{\hat{y}})}{\sum_{k'=1}^K \exp(W_{k'}^T h_t + b_{k'})}. \quad (6)$$

As we can see the softmax layer outputs probabilities by a normalization operation, which is usually called the Cross Entropy (CE) error function in machine learning. The computation cost increases linearly with the number of categories. So the cost of CE is  $|V| + 1$ , where  $V$  represents the number of vocabularies to be predicted, and the cost of NS is  $|K| + 1$ . The speed up ratio is  $K/V$  (NS is much faster). Since in each NLP word classification task, there is only one positive category (the word to be predicted) and all other words are negative. So instead of updating weights through whole negative vocabularies, NS only samples some negative words for weight updating.

APIHelper tries to build one single LSTM network for all API suggestions. However, the biggest challenge is that the number of candidate APIs to be predicted is so huge (for common Java classes, there are more than 30,000 candidate APIs in total). From above we know the softmax is inefficient when dealing with such a big multi-class classification. Therefore we propose DNS to solve this. The main difference between DNS and NS is that DNS is a deterministic sampling strategy rather than a random sampling. The deterministic sampling is from the specific constraints for API calls that every API call must come from one specific Java class. Considering the API we need to predict must belong to the current Java class, we can use the last Java class  $c_T$  to limit the range of candidate APIs to be predicted. There are around 5 to 108 APIs for a common Java class, so the LSTM network does not need to do a prediction in the entire candidate APIs set. Instead, it first picks out APIs subset for current Java classes, and then makes a prediction based on that. Meanwhile, the LSTM network computes softmax errors and does weight updating only for negative API samples in this subset.

For the specific implementation of DNS, current mainstream deep learning frameworks (such as Tensorflow, Caffe, Theano, etc.) only provide a random negative sampling function. For example, Tensorflow simply provides a random negative sampling error function: `tf.nn.sampled_softmax_loss()`, which computes the cross entropy over the subset of candidate classes. Since it cannot achieve the function of DNS, we devise a special LSTM training module to implement DNS in Tensorflow. Specifically, in each iteration, we introduce an additional Tensorflow place holder DNS to reduce the prediction probability manually for APIs not belonging to current Java classes. Then LSTM calculates the cross entropy error for this fixed output probability:

$$dns_{1*N} = [b_{API_1}, b_{API_2}, b_{API_3}, \dots, b_{API_N}], \text{ where} \\ b_{API_n} = \begin{cases} \tau & API_n \in \text{current Java class } C_T \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

$$logtis_{1*N} = [p_{API_1}, p_{API_2}, p_{API_3}, \dots, p_{API_N}], \quad (8)$$

where *logtis* represents the logistics probability of the original LSTM prediction. *logtis* adds *dns* as bias, that is  $logtis = logtis + dns$ , and calculates the loss for back propagation, shown as: `tf.nn.softmax_cross_entropy_with_logits(logits=logtis, labels= ...)`. It can be seen as passing the prior knowledge to the LSTM network by *dns*. By manually reducing the output probabilities of some unrelated candidate APIs, their cross entropy error is very low. This in turn lets the LSTM network focus on learning to classify or distinguish APIs belonging to current Java classes during error back propagation.

## IV. EXPERIMENT AND RESULTS

### A. Dataset and Data Preprocessing

We crawled 18,000 Java projects from Github and extracted corresponding Java source codes for a total of 16GB data. To ensure a high quality of Java codes dataset, we only collect Java projects with over 100 stars. More stars means more popular project and higher programming quality.

For example, APIHelper learns the most popular Java APIs, as follows: *java.\**, Java foundation classes; *javax.\**, Java foundation classes extension; *android.\**, Android related Java classes; *org.apache.\**, all top level Java project from Apache software foundation; *com.google.\**, all Java project provider by Google; *org.springframework.\**, one of the most popular structure for Java web development.

Above Java classes and APIs cover daily function needs for programmers. However, these Java libraries also contain some rarely used Java classes and APIs. Thus we do the frequency counts for APIs and use UNK to represent Java classes that occur less than 20 times and Java APIs less than 10 times in our dataset. This greatly reduces the size of embedding matrix for LSTM. Only 50,000 Java classes and 60,000 Java APIs are reserved, so the hyperparameter *num\_emb* is reduced from original 540,000 to 110,000.

The total number of candidate APIs to be predicted is 20,000, but the frequency of these APIs varies a lot. The most frequent API is *newInstance* which occurs more than 5,000,000 times in our dataset, while the least popular API only appears 50 times. Hence we use data augmentation strategy to rebalance the distribution, which achieves the maximum augmentation ratio up to 40 times.

### B. Model Setup and Training Details

In experiment, APIHelper is trained by GPU. We use NVIDIA's GTX980 GPU and related software repositories including CUDA 7.5, cuDNN V4, Python 2.7.6, Numpy 1.8.2, Scipy 0.13.3, and Tensorflow 0.9.0. APIHelper uses a two-layer LSTM network, and each layer contains 128 neuron units. For the design of LSTM structure, we use ReLU as activation function and use Dropout with dropping probability 0.5 to ease overfitting problem. To ensure the LSTM network convergence, we use orthogonal weight initialization with a range of  $\pm 0.04$ , add batch normalization layers, and set gradient regularization factor to 10 (it is used to control gradient expansion). APIHelper adopts an optimizer based on

the Adam optimization algorithm, sets the initial learning rate of  $\eta = 5e - 03$ , and takes sequence length of 60 as the input and batch size of 228. Also, we set the maximum number of training epoches (=150,000). Since we use early stop strategy, usually the training phase stops after 5 complete rounds.

Each element of an input API sequence consists of two parts: Java class embedding with vector length of 150 and API embedding with vector length of 200. We propose EC to concatenate two-part embedding vectors together along the horizontal axis,  $x_t = [e_t^c, e_t^a]$ . So the input API sequence  $[x_1, x_2, x_3, \dots, x_T]$  can be represented as  $\{[e_1^c, e_1^a], [e_2^c, e_2^a], [e_3^c, e_3^a], \dots, [e_T^c, e^{pred}]\}$ , where  $e^{pred}$  is the API to be predicted. The embedding lookup matrix is initialized randomly, and it uses  $0.1 * \eta$  to adjust learning rate (otherwise model will get seriously overfitting during training).

### C. Results and Evaluation

We use 18,000 Java projects and 10-fold cross validation to evaluate the performance of LSTM network used in APIHelper, as well as the effect of using EC and DNS to verify whether they are valid. The results are shown in Table I.

TABLE I  
THE PERFORMANCE FOR DIFFERENT LSTM NETWORKS

Methods	Accuracy(%)	top-5 Accuracy(%)
Standard LSTM	40	79
LSTM + EC	43	81
LSTM + EC + DNS (APIHelper)	<b>53</b>	<b>90</b>

We use accuracy and top-5 accuracy as evaluation indicators. Here top-5 accuracy represents the prediction accuracy when the model can provide top-5 predicted APIs. As we can see from Table I, both EC and DNS give an improvement of prediction accuracy. Specifically, the standard LSTM model serves as the baseline method which regards each Java class or API as one input, while EC merges and concatenates them into one long vector as another input. EC enables LSTM to learn API sequence in a more efficient way and capture more long-term contextual information. So top-5 accuracy raises from 79% to 81% when using EC.

Apart from this, DNS has the most benefit to LSTM performance, raising top-5 accuracy from 81% to 90%. For each input API sequence, DNS builds one corresponding softmax layer which narrows down the candidate APIs to predict according to the last input Java class. Hence LSTM only needs to choose one prediction API belonging to the last input Java class instead of all Java APIs. This greatly reduces the search space of LSTM network, making it more efficiently during training and easier to converge. Most importantly, this makes it possible to predict massive Java APIs (give a API suggestion for massive Java classes) by using one single LSTM network, because it simplifies the original multi-class classification problem with hundreds of thousands of categories into hundreds of categories classification.

In addition, we compare APIHelper with the current two mainstream methods, HAPI [2] and N-gram. Original task of HAPI is for Android API learning. It first extracts Android bytecodes from .apk file through dex parser tool (e.g. *baksmali*)

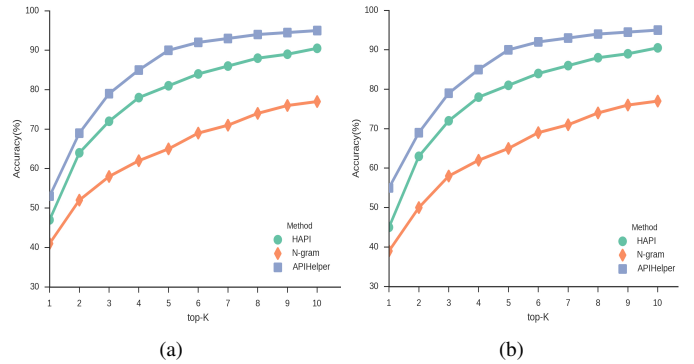


Fig. 4. (a) The results of API suggestion for predicting next API. (b) The results of API suggestion for filling API hole.

and converts them to CFG, then traverses CFG to generate API sequences for training HMM (HMM takes the sequence as the input). For each Java class, HAPI trains one specific HMM model and takes two experiments for evaluation, which are called “predict next API” and “fill API hole”. The difference is the latter task not only takes the context before the API to be predicted, but also the context after it. The experiment takes a 63GB dataset for 2,700 Java classes with 17,000,000 API sequences. Each Java class has around 6,000 API sequence samples and the average length of an API sequence is 4. HAPI trains 1,200 HMM models on 2,700 Java classes, with averaging eight hidden states per HMM.

The goal of HAPI is to solve Android API learning, which is a subtask compared with Java API suggestion, but APIHelper needs to predict APIs for 18,000 Java project codes which contain all Android related APIs and many other Java APIs. Thus, in order to better evaluate and compare APIHelper with HAPI, we reproduce the HAPI method and conduct a comparative experiment with the same task on our dataset.

Also, we make the experiment of 3-gram for API suggestion or recommendation as a baseline. We first filter out low frequency 3-gram combinations, then use the Random Forest (RF) model for feature importance analysis, from which a 43,000-dimensional feature vector is extracted. Finally, we use these features with softmax regression to achieve multi-class classification and get the API suggestion result. As we can see from the results shown in Fig.4(a) and Fig.4(b), APIHelper outperforms other two methods. The performance of HAPI is somehow lower than the original results in their paper. One reason is that our task is to make an API suggestion or recommendation at any randomly chosen location rather than at the end of a well-extracted API sequence. Thus sometimes the context is relatively insufficient.

### D. Computation Efficiency

We analyze the computational efficiency and computation resource consumption of all methods, and separately calculate the storage consumption, training time and API suggestion time-consuming for comparison. The evaluation results are shown in Table II. As we can see, the training time consumption for standard LSTM model takes 2.4 hours, which is obviously slower than HAPI. Besides, the slowest method

TABLE II  
THE EXPERIMENT RESULTS FOR DIFFERENT METHODS

Methods	Accuracy(%)	top-5 Accuracy(%)	Training Time(h)	Disk Space Consumption(MB)	Suggestion Time(ms)
Standard LSTM	40	79	2.4	328	23.3
LSTM + EC	43	81	1.5	263	13.5
3-gram	43	68	1.4	182	172
HAPI [2]	48	81	1.2	<b>18.5</b>	14.4
<b>APIHelper</b>	<b>53</b>	<b>90</b>	<b>0.5</b>	263	<b>13.2</b>

is 3-gram model, mainly because 3-gram takes a lot of time for feature extraction. As mentioned before, 3-gram counts all occurrences of 1-gram, 2-gram and 3-gram in the dataset, which means doing millions of counts as features. After that the N-gram statistical method uses a simple classifier which costs a little time for training.

However, after using EC and DNS, APIHelper only needs half an hour to complete the training process. The main reasons are as following: first EC almost reduces the input sequence length by half for LSTM. EC concatenates class and API embeddings as  $[e_t^c, e_t^a]$ , so LSTM timestep takes more information for each one timestep. Then DNS further speeds up the training phase because the back propagation calculation only needs to be done on some softmax nodes instead of all of them. This greatly accelerates the weight updating speed for LSTM network. At last, coupled with GPU-accelerated deep neural network computation, APIHelper takes the shortest time for training. For HAPI [2], thought training a single HMM model for a Java class takes less time, HAPI needs to train thousands of HMMs for all Java classes. Overall, it is less efficient than APIHelper. Since the model training phase is often offline, the more critical indicator is the time-consuming in API suggestion phase, which reflects the response speed of API suggestion system. In the prediction phase, the LSTM network is the fastest (13.2ms) which only needs to do one forward propagation to get the prediction result.

Finally, we compare the memory and disk consumption. Since data storage is critical for all methods, here the main storage consumption we evaluate is the model storage for storing model parameters. It can implicitly indicate the memory consumption because usually a system loads all model parameters into memory for fast computation. The evaluation results show APIHelper and N-gram consume more storage than HAPI. APIHelper needs more storage because LSTM network is complex and contains mass parameters. However, all three methods consume less than 500MB of storage space, which we think is not a bottleneck for current computer resources in a server with GPU computation.

## V. CONCLUSIONS

In this paper, we explore a new LSTM-based API suggestion approach to improve programming efficiency. To this end, we construct a prototype implementation called APIHelper, which uses what we called deterministic negative sampling to build one single end-to-end LSTM network to make API suggestion for tens of thousands of Java APIs. While experiments show that APIHelper can effectively provide API suggestions according to the API contexts and has better performance in

terms of suggestion accuracy, computational efficiency and scalability compared with previous works.

For the limitation of APIHelper, it has to rely on class object information for API suggestion, so currently it can only be applied to strongly typed languages. In the future work, we will plan to explore APIHelper making API suggestion for weakly typed languages like Python, which is a more challenging task.

## ACKNOWLEDGMENT

This work is partially supported by the National Natural Science Foundation of China under Grant No. 61672421 and the Shaanxi Provincial Natural Science Foundation under Grant No. 2017JM6109.

## REFERENCES

- [1] T. T. Nguyen, H. V. Pham, P. M. Vu, et al. Recommending API usages for mobile apps with hidden markov model. In Proc. of 30th IEEE/ACM International Conference on Automated Software Engineering (ASE), 795-800, 2015.
- [2] H. V. Pham, P. M. Vu, T. T. Nguyen. Learning API usages from bytecode: A statistical approach. In Proc. of the 38th International Conference on Software Engineering (ICSE), 416-427, 2016.
- [3] T. Xie and J. Pei. MAPO: mining API usages from open source repositories. In Proc. of the International Workshop on Mining Software Repositories (MSR), 54-57, 2006.
- [4] J. Wang, Y. Dang, H. Zhang, K. Chen, T. Xie and D. Zhang. Mining succinct and high-coverage API usage patterns from source code. In Proc. of the 10th Working Conference on Mining Software Repositories (MSR), 319-328, 2013.
- [5] J. Fowkes, C. Sutton. Parameter-free probabilistic API mining across GitHub. In Proc. of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE), 254-265, 2016.
- [6] T. T. Nguyen, H. A. Nguyen, N. H. Pham, et al. Graph-based mining of multiple object usage patterns. In Proc. of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering (ESEC/FSE), 383-392, 2009.
- [7] T. D. Nguyen, A. T. Nguyen, H. D. Phan, et al. Exploring API embedding for API usages and applications. In Proc. of the 39th IEEE/ACM International Conference on Software Engineering (ICSE), 438-449, 2017.
- [8] X. Gu, H. Zhang, D. Zhang, et al. Deep API learning. In Proc. of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE), 631-642, 2016.
- [9] V. Raychev, M. Vechev, E. Yahav. Code completion with statistical language models. In Proc. of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), 419-428, 2014.
- [10] H. Zhong, T. Xie, L. Zhang, J. Pei and H. Mei. MAPO: Mining and recommending API usage patterns. In European Conference on Object-Oriented Programming (ECOOP), 318-343, 2009.
- [11] H. A. Nguyen, T. T. Nguyen, Jr. G. Wilson, et al. A graph-based approach to API usage adaptation. In Proc. of the ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA), 302-321, 2010.
- [12] H. Niu, I. Keivanloo, Y. Zou. API usage pattern recommendation for software development. The Journal of Systems and Software, 129: 127-139, 2017.
- [13] T. Yamamoto. Code suggestion of method call statements using a source code corpus. In Proc. of the 24th Asia-Pacific Software Engineering Conference (APSEC), 2017.

# Adaptive software search toward users' customized requirements in GitHub

Jinze Liu, Zhixing Li, Tao Wang, Yue Yu and Gang Yin  
College of Computer Science  
National University of Defense Technology  
Changsha, Hunan, China  
jinze\_liu@qq.com, {lizhixing,taowang2005,yueyu,yingang}@nudt.edu.cn

**Abstract**—Because of a tremendous growth of Open Source Software (OSS) scale and the diversity of users' requirements, users now face the problem of finding OSS that meets their expectations in a huge number of OSS resources. However, current GitHub-provided search service has a shortage in adapting to user needs. When facing diverse users' requirements, it cannot always return satisfactory results. In this paper, we provide a more efficient search service for OSS on GitHub. We first design a multi-dimensional measurement model for OSS, which forms a corresponding metric system and quantitative measurement method. Then we propose a ranking algorithm based on fuzzy synthetic evaluation in order to implement an adaptive metric ranking method that is oriented to user requirements. We verify that our work is useful by setting up experiments. The experiment results show that compared with GitHub-provided search service (searching by “Best Match” & searching by “Most Stars”), the effectiveness of our method improved by 97.6% and 13.8% respectively, which means our method returns search results which meet users' expectations more, and has high self-adaptive ability.

**Keywords**—Open Source Software; Search; Multi-dimensional Measurement; Fuzzy Synthetic Evaluation; GitHub

## I. INTRODUCTION

With the rise of the open source movement, OSS has made tremendous growth and the global OSS resources have become an Internet-scale repository. Especially in recent years, the rapid development of Internet technology has greatly enhanced the influence of OSS around the world. Many well-known OSS hosting platforms have emerged and the most successful platform among them is GitHub. GitHub is a social programming and code hosting platform [1], officially launched on April 10, 2008, after which countless open source projects began to migrate to this platform, and the number of hosted software showed the situation of “explosive” growth. As of December 2016, GitHub has hosted more than 35 million projects and attracted more than 14 million developers to participate in open source activities [2].

OSS resources are already a huge and diverse ecosystem. At the same time, the users' requirements of OSS is also diverse. For example, some users like the popular OSS currently. Some of them are more concerned about whether the authors of OSS have high development enthusiasm and make good maintenance of their work. And others may look for open-source software that still maintains high activity. Therefore,

users now face the problem of finding OSS that suits their diverse requirements in a huge number of OSS resources.

In a web search, as stated by studies [3] [4], people often prefer results located on the first page, or at most, the first three pages. Later search results people tend to no longer be concerned about, so a good search service needs to make more important results come in front. For this reason, GitHub provides ranking indicators based on the text matching, popularity, etc. of OSS, but the results are not satisfactory. For example, if the text matching degree is used as a ranking indicator, GitHub only considers the matching between the software name or description, and the users' search keywords, thus the quality of the software is not guaranteed. Due to the low threshold of creating OSS in open source community and the different ability of authors of OSS, there are a large number of OSS with low quality in GitHub. So if the search service does not consider the quality of OSS, the search results will have very limited help to users. If we choose popularity as the ranking indicator, GitHub ranks only by the number of “Star” of the repository of OSS. Other attributes of OSS are not taken into account at all, which makes the search results still less helpful.

Although GitHub provides an advanced search service, allowing users to propose multiple search criteria, from the point of actual use, GitHub simply combines multiple search criteria together. For example, if users choose both text matching degree and popularity as ranking indicators, the final search results are only based on the search results of text matching degree, and delete the results which popularity do not meet the requirements. So if what users want is being more interested in the software which is more popular among all the related software, the search results provided by GitHub still cannot give them a satisfied answer. All in all, GitHub currently offers a weak search service. On the one hand, it provides users with less choice of ranking indicators. On the other hand, it cannot adapt to complex users' requirements. When users select an indicator to rank, other indicators will be ignored, and this often does not meet users' expectations.

Therefore, we provide a more efficient search service for OSS on GitHub by a multi-dimensional measurement model and a ranking algorithm based on fuzzy synthetic evaluation. In this paper, we first build a GitHub OSS information database that contains information about the attributes of OSS

such as name, description, number of “Watch”, situation of “Pull Request” (PR), and so on. Second, we get preliminary search results by keywords matching. Third, we extract ranking indicators and design a multi-dimensional measurement model according to users’ requirements. Fourth, we propose and implement a ranking algorithm based on fuzzy synthetic evaluation. This algorithm calculates the synthetic evaluation score of OSS according to the weight of OSS attributes, so that we can get final search results from the order of evaluation scores. At last, by setting up experiments with different users’ customized requirements scenarios, we compare our search results with those of GitHub-provided search service and verify that our method returns search results which meet users’ expectations more, and has high self-adaptive ability.

The key contributions of this study include the following:

- We analyze the key factors when users choose OSS from four dimensions: popularity of software, collaborative development of software, development attitude of software author and activeness of software, and propose a detailed quantitative measurement method.
- We allow users to set their personal search requirements and provide users with enhanced, easier-to-use search service.
- We propose and implement an OSS ranking algorithm based on fuzzy synthetic evaluation, so that the OSS can obtain the corresponding evaluation score according to different users’ customized requirements so as to make the search results more in line with users’ expectations, greatly improving the self-adaptive ability of the search service.

The rest of this paper is organized as follows. Section II introduces related research on software ranking and fuzzy mathematics theory. Section III elucidates the approach of our study. Section IV elaborates our experimental process and results. Section V concludes this paper and introduces future work.

## II. RELATED WORK

### A. Software Ranking

In the field of software engineering, researchers evaluate software quality through software evaluation models, and then rank the software based on software quality. Researchers have proposed Capgemini maturity model [5], Navica maturity model [6], OpenBRR model [7], QSOS model [8], and SQO-OSS model [9] and so on.

In the field of open source community, open source communities typically use the feedback from community users to rank the OSS. For example, GitHub provides the function of ranking OSS according to the “Most Star” indicator. The higher the number of “Star” of a software, the more popular it is in the community. SourceForge uses the information which is collected from users about the download and comment situation of OSS to calculate the popularity of them, and then rank the software. OpenHub has designed a button named “I use it” for each OSS, which users can click to mark this

software was used. Then the platform ranks by the number of users of them.

However, these ranking methods only consider a single ranking indicator, not comprehensive enough to meet more complex users’ customized requirements.

### B. Fuzzy Synthetic Evaluation

Fuzzy synthetic evaluation is one of the most widely used methods for multi-index synthetic evaluation [10]. With the help of the theory of membership degree of fuzzy mathematics, it turns qualitative evaluation into quantitative evaluation, which can solve the problem of fuzzy and hard to quantify. The basic steps of fuzzy synthetic evaluation shows in Table I [11].

TABLE I: Basic Steps of Fuzzy Synthetic Evaluation

1. Determine the evaluation indicator set and evaluation level set of the evaluation target.
2. Determine the weight of each evaluation indicator and its membership vector in order to get the fuzzy evaluation matrix.
3. Make fuzzy operation between the fuzzy evaluation matrix and the weight set of evaluation indicators.
4. Normalize the calculation results in step 3 and get the evaluation results.

## III. APPROACH

The goal of our work is providing users with OSS search service which has high self-adaptive ability. As shown in Figure 1, we first get development history data of OSS from GitHub to build an information database. Then we get preliminary search results by keywords matching. Third, we extract ranking indicators and design a multi-dimensional measurement model according to users’ requirements. Fourth, we get final search results by a ranking algorithm based on fuzzy synthetic evaluation. In the following sections, we will elaborate each step in detail.

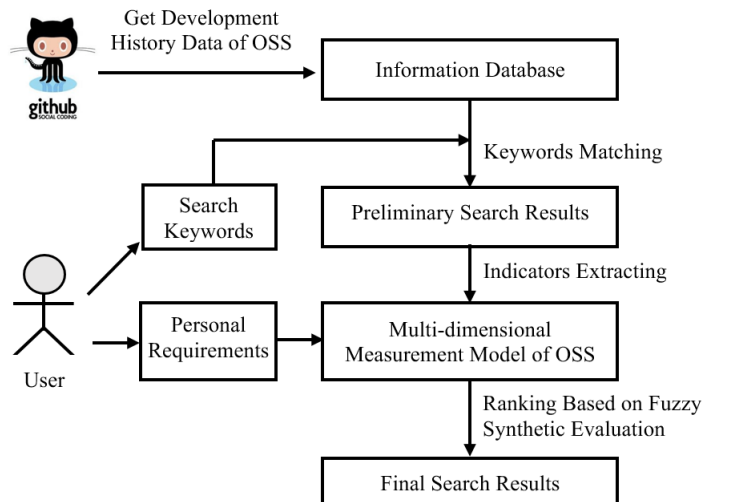


Fig. 1: Overall Framework of Our Method

### A. Database Building

**Dataset collection.** In this paper, we use the “GHTorrent” project, which monitors the GitHub public event timeline, to get GitHub open source data [2]. Whenever an event occurs on GitHub, the project retrieves its content and all its dependencies and stores them, then releases the data regularly for users to download.

**Dataset cleaning.** GitHub allows users to “Fork” the origin OSS project to create a new branch based on the main branch of the project. The main branch and the “Fork” branch actually refer to the same project. However, there are multiple records stored in the database. Therefore, in order to avoid duplication of processing, all “Fork” branches need to be removed and the corresponding main branch should be reserved.

### B. Preliminary Search Based on Keywords Matching

In this section, we get all OSS related to users’ search keywords as preliminary search results by matching keywords with the name and description of software. First, we use WordNet to make synonym expansion of keywords in order to improve the recall rate. Second, we set whether software name or description contains keywords as the standard of whether software is related to keywords.

### C. Multi-dimensional Measurement Model for OSS

By collecting lots of posts of IT Q & A communities (such as Stack Overflow etc.) and analyzing the community users’ comments on their needs for OSS, a multi-dimensional measurement model is presented for OSS. As shown in Figure 2, this model includes four dimensions, and each dimension has one or more indicators. Each indicator is described by one or more online attributes through the Github open source project repository.

**Popularity of software.** In general, the software of higher popularity degree often means more famous and has higher quality. So an indicator named *software popularity degree* is introduced to describe how popular is an OSS.

In GitHub, if developers in the community are interested in an OSS project, their first choice is to “Watch” the software. Then, whenever the software has any dynamic information, developers can receive the corresponding data at the first time, which like following celebrities interested in Facebook.

Therefore, the number of “Watch” of an OSS can describe its *software popularity degree*, the higher the number of “Watch”, on behalf of the developers concern more about this software. This indicator can be calculated as Equation 1.

$$P_{popularity}(j) = Nor(watch(j)) \quad (1)$$

where  $Nor()$  is normalization and can be calculated as bellow:

$$Nor(watch(j)) = \frac{watch(j) - \min(watch)}{\max(watch) - \min(watch)} \quad (2)$$

**Collaborative development of software.** GitHub is an open source collaborative development community. For the software the developers are interested in, developers can work with the related authors. Developers who involved in collaborative

development are called as contributors to this OSS. So an indicator named *software coordination degree* is introduced to describe the enthusiasm of contributors to participate in the collaborative development of an OSS project.

In GitHub, the development mechanism based on “Fork→Pull Request” is widely used [12]. If developers want to participate in the collaborative development of a software, they can clone the main branch of this software to developers’ local branch through “Fork”. After that, they can implement some new features or fix bugs based on their personal repository cloned from the latest version of project repository. When their work is finished, the patches are packaged as a PR submitted to GitHub and reviewed by software authors, which indicates that developers request that their work be merged into the main branch of the original project.

Therefore, the number of PR of an OSS can describe its *software coordination degree*, the higher the number of PR, on behalf of the more requests are sent and the higher enthusiasm of contributors to participate in the collaborative development. This indicator can be calculated as Equation 3.

$$P_{coordination}(j) = Nor(pull\_request(j)) \quad (3)$$

**Development attitude of software author.** When developers want to participate in a collaborative development of OSS, they do not prefer those OSS developed by authors who are indifferent to their work. When they encounter problems during the development, the authors will not actively communicate with them and solve the problem, thus affecting the user experience. This dimension describes how positive is an author in his work.

The number of submitted PR per day on GitHub is enormous, leading to the update iterative efficiency of OSS largely depends on whether PR can be reviewed in a timely manner. Study [13] showed that the review of PR is a very important way for distributed software development community like GitHub to maintain the quality of code of OSS.

Therefore, two indicators including response degree of PR ( $P_{response}$ ) and response speed of PR ( $S_{response}$ ) are introduced, which can be calculated as Equation 4.

$$\begin{aligned} P_{response}(j) &= Nor\left(\frac{pull\_request_{response}(j)}{pull\_request(j)}\right) \\ S_{response}(j) &= Nor\left(\frac{1}{T_{latency}(j)}\right) \\ &= Nor\left(\frac{1}{Mid_{1 \leq i \leq M}\left(T_{closed}^i(j) - T_{opened}^i(j)\right)}\right) \end{aligned} \quad (4)$$

where  $P_{response}$  is the ratio of the number of responded PR to the total number of PR.  $S_{response}$  is the reciprocal of the medium number of PR evaluation latency, which is the time difference between the PR opened and closed.

**Activeness of software.** The less active OSS often means that its author has not updated it for a long time and has even abandoned its development. So we use the reciprocal of the



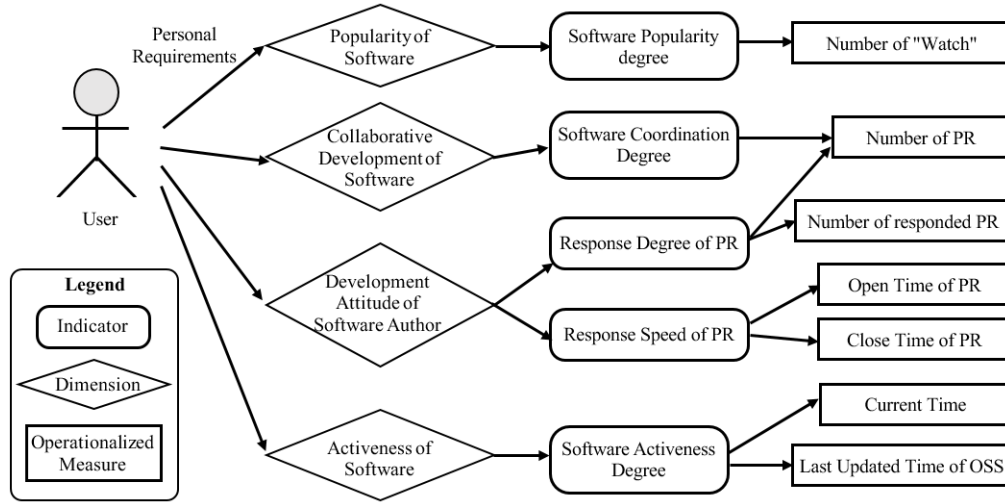


Fig. 2: Multi-dimensional Measurement Model for OSS

time difference between current time and the last updated time of OSS to describe an indicator named *software activeness degree*, which can be calculated as Equation 5.

$$P_{activeness}(j) = Nor \left( \frac{1}{T_{current} - T_{updated}(j)} \right) \quad (5)$$

For those dimensions above, users can choose one or more of them and set the corresponding importance order. For example, if users want to find OSS in great quality, they would choose *popularity of software* as the first important dimension. Then we determine the weight of dimensions according to their requirements. After many experiments, we select a set of weight values that have the better search performance, and the results shows in Table II.

TABLE II: Weight of Dimensions

Number of dimensions	The order of importance			
	First	Second	Third	Fourth
4	0.4	0.3	0.2	0.1
3	0.5	0.3	0.2	
2	0.7	0.3		
1	1.0			

As mentioned before, there are two indicators (*response degree of PR* and *response speed of PR*) to describe the dimension: *development attitude of software author*. So the weight of both indicators will be the half of the weight of this dimension.

#### D. Ranking Based on Fuzzy Synthetic Evaluation

We build a fuzzy synthetic evaluation model of OSS to rank the preliminary search results by using evaluation score of software.

**Evaluation indicator set and evaluation level set.** In this paper, the evaluation indicator set  $U$  is:

$$U = \{u_1, u_2, u_3, u_4, u_5\} \quad (6)$$

where  $u_1$  represents *software popularity degree*,  $u_2$  represents *software coordination degree*,  $u_3$  represents *response degree of PR*,  $u_4$  represents *response speed of PR* and  $u_5$  represents *software activeness degree*.

We choose “excellent”, “good”, “general”, “bad” and “awful” as the evaluation level set  $V$ :

$$V = \{v_1, v_2, v_3, v_4, v_5\} \quad (7)$$

where  $v_1, v_2, v_3, v_4, v_5$  represents “excellent”, “good”, “general”, “bad” and “awful” respectively.

**Membership function.** We use the triangle function as membership function in our evaluation model.  $\mu_{v_1}, \mu_{v_2}, \mu_{v_3}, \mu_{v_4}, \mu_{v_5}$  is respectively the membership function of “excellent”, “good”, “general”, “bad” and “awful”.

$$\begin{aligned} \mu_{v_1}(x) &= \begin{cases} 4x - 3 & x \in [0.75, 1] \\ 0 & \text{others} \end{cases} \\ \mu_{v_2}(x) &= \begin{cases} 4x - 2 & x \in [0.5, 0.75] \\ -4x + 4 & x \in (0.75, 1] \\ 0 & \text{others} \end{cases} \\ \mu_{v_3}(x) &= \begin{cases} 4x - 1 & x \in [0.25, 0.5] \\ -4x + 3 & x \in (0.5, 0.75] \\ 0 & \text{others} \end{cases} \\ \mu_{v_4}(x) &= \begin{cases} 4x & x \in [0, 0.25] \\ -4x + 2 & x \in (0.25, 0.5] \\ 0 & \text{others} \end{cases} \\ \mu_{v_5}(x) &= \begin{cases} -4x + 1 & x \in [0, 0.25] \\ 0 & \text{others} \end{cases} \end{aligned} \quad (8)$$

**Fuzzy evaluation matrix.** For each indicator, we get its single factor evaluation according to the membership function.

Then we get the fuzzy evaluation matrix bellow:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} & r_{15} \\ r_{21} & r_{22} & r_{23} & r_{24} & r_{25} \\ r_{31} & r_{32} & r_{33} & r_{34} & r_{35} \\ r_{41} & r_{42} & r_{43} & r_{44} & r_{45} \\ r_{51} & r_{52} & r_{53} & r_{54} & r_{55} \end{bmatrix} \quad (9)$$

where  $r_i = (r_{i1}, r_{i2}, r_{i3}, r_{i4}, r_{i5})$  represents the single factor evaluation of  $u_i$ .

**Evaluation indicator weight set.** In this paper, the evaluation indicator weight set  $W$  is:

$$W = \{w_1, w_2, w_3, w_4, w_5\} \quad (10)$$

where  $w_i$  is the weight of indicator  $u_i$ .

**Evaluation model and evaluation score.** We get the fuzzy evaluation set  $S$  bellow:

$$\begin{aligned} S &= W * R \\ &= (w_1, w_2, w_3, w_4, w_5) * \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} & r_{15} \\ r_{21} & r_{22} & r_{23} & r_{24} & r_{25} \\ r_{31} & r_{32} & r_{33} & r_{34} & r_{35} \\ r_{41} & r_{42} & r_{43} & r_{44} & r_{45} \\ r_{51} & r_{52} & r_{53} & r_{54} & r_{55} \end{bmatrix} \\ &= (s_1, s_2, s_3, s_4, s_5) \end{aligned} \quad (11)$$

where  $*$  is fuzzy synthesis operator and there are four common operators:  $M(\wedge, \vee)$ ,  $M(\bullet, \vee)$ ,  $M(\wedge, +)$  and  $M(\bullet, +)$ . In this paper, we use  $M(\bullet, +)$ :  $s_j = \sum_{i=1}^5 w_i \bullet r_{ij}$

Finally we get the evaluation score bellow:

$$Score_k = \frac{\sum_{i=1}^n c(v_i) \bullet s_i^k}{\sum_{i=1}^n s_i^k} \quad (12)$$

where  $c(v_i)$  is the quantified value of each evaluation level, and can be expressed as bellow:

$$\{c(v_1), c(v_2), c(v_3), c(v_4), c(v_5)\} = \{5, 4, 3, 2, 1\} \quad (13)$$

#### IV. EXPERIMENT

##### A. Experimental Setup

Stack Overflow is the most popular developer community of asking and answering questions and it is a platform which reflects the real requirements of developers. As shown in Table III, we summarize several hot software types by analyzing the tags in Stack Overflow.

TABLE III: Hot Software Types in Stack Overflow

Software type	Number of tags
database	138070
machine learning	20603
web crawler	6560

So we choose “database”, “machine learning” and “web crawler” as the search keywords in the experiment.

In addition, we also summarize two representative users’ customized requirements:

**Requirement I. Users want to find OSS resources for software reuse.** As for software reuse, users need OSS in

better quality and more mature. When the OSS has a high degree of popularity and remains active, the quality of such OSS can be better guaranteed. Therefore, *popularity of software* is the first important dimension and *activeness of software* is the second important dimension.

**Requirement II. Users want to find OSS which is suitable for collaborative development.** For this need, users need OSS of which contributors participate in passionately. These software authors communicate with contributors frequently, timely, and actively maintain software. Therefore, *collaborative development of software* is the first important dimension and *development attitude of software author* is the second.

##### B. Evaluation Metrics

In user study, in order to compare the effectiveness of our search service and GitHub-provided search service, volunteers will be asked to evaluate whether the top 10 OSS that in the search results offered by our method and GitHub (include searching by “Best match” and searching by “Most stars”) is what they are looking for. The evaluation is a Likert-type scale with a more detailed expression for each choice [14]. The respondents are asked to choose one of three candidate response items, that is, our evaluation process is a three-point Likert scale. Table IV describes these three candidates.

TABLE IV: Likert Scale Response Categories

Scale	Response category
3	perfect expectations
2	general expectations
1	few expectations

To do the evaluation, twenty individuals with different backgrounds were invited to assess the result. Among them, nine are master students, six are Ph.D. students and five engineers with at least three years software development experience.

At last, we believe that software which is more in line with users’ requirements should be ranked in more front. So according to the rank of software, we use *weighted\_Likert\_score* to describe the effectiveness of search results.

$$weighted\_Likert\_score = \sum_{i=1}^{10} scale_i * weight_i \quad (14)$$

For the software ranked in the first, its weight is 1.0. For the software ranked in the second, its weight is 0.9. And for the last, the weight is 0.1. So the full marks of *weighted\_Likert\_score* is 16.5.

##### C. Experimental Results

Volunteers will evaluate OSS separately according to Requirement I and Requirement II.

As for Requirement I, after volunteers tried out these OSS, they evaluated whether the OSS was mature and whether the quality reached their expectations.

As for Requirement II, after participating in the collaborative development of these OSS, volunteers assessed the

enthusiasm of other contributors and whether the software authors had good communication with them.

Figure 3 shows the evaluation results of user study in Requirement I.<sup>1</sup>

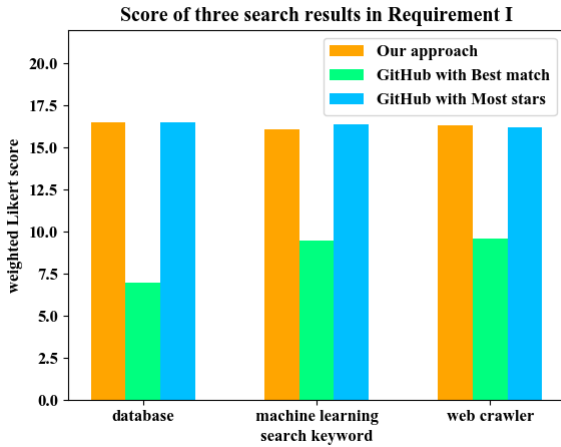


Fig. 3: Score of three search results in Requirement I

Figure 4 shows the evaluation results of user study in Requirement II.

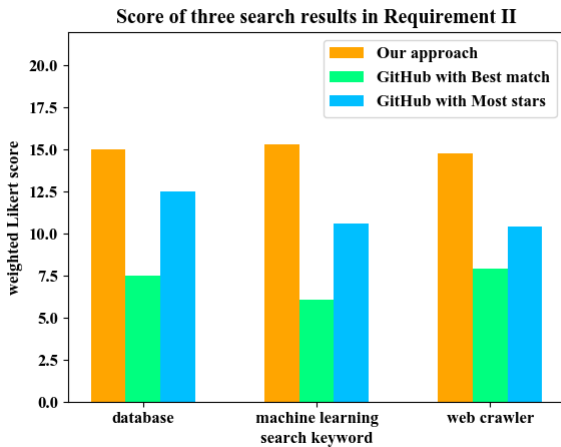


Fig. 4: Score of three search results in Requirement II

Through Figure 3 and Figure 4 we conclude the following conclusions:

- Whatever users' requirements are, searching by "Best match" has limited help with users.
- Searching by "Most stars" has better performance in finding software in high quality than finding suitable software for collaborative development. Because it does not take other attributes into account.
- Our method shows good results under different users' customized requirements and reflects a high self-adaptive ability.

- Compared with searching by "Best match", the average score of our method increased by 97.6%.
- Compared with searching by "Most stars", the average score of our method increased by 13.8%.

## V. CONCLUSION AND FUTURE WORK

In this paper, we first introduce a problem that users have to find OSS that suits their diverse requirements in a huge number of OSS resources and analyze why the current GitHub-provided search service is weak. Then we propose our own approach, which is providing a more efficient search service for OSS on GitHub by a multi-dimensional measurement model and a ranking algorithm based on fuzzy synthetic evaluation. At last, we verify that our method returns search results which meet users' expectations more, and has high self-adaptive ability.

The software data hosted on GitHub is huge today and will continue to grow rapidly. So in the future, we plan to optimize our algorithm and improve its efficiency.

## REFERENCES

- [1] A. Begel, J. Bosch, and M.-A. Storey, "Social networking meets software development: Perspectives from github, msdn, stack exchange, and topcoder," *IEEE Software*, vol. 30, no. 1, pp. 52–66, 2013.
- [2] G. Gousios, "The ghtorrent dataset and tool suite," in *Proceedings of the 10th working conference on mining software repositories*. IEEE Press, 2013, pp. 233–236.
- [3] C. D. Manning, P. Raghavan, H. Schütze *et al.*, *Introduction to information retrieval*. Cambridge university press Cambridge, 2008, vol. 1, no. 1.
- [4] A. Aula, P. Majaranta, and K.-J. Räihä, "Eye-tracking reveals the personal styles for search result evaluation," in *IFIP Conference on Human-Computer Interaction*. Springer, 2005, pp. 1058–1061.
- [5] F. Duijnhouwer and C. Widdows, "Open source maturity model. capgemini expert letter," *EU QualOSS project (grant number: 033547, IST-2005-2.5. 5)*, 2003.
- [6] B. Golden, *Succeeding with open source*. Addison-Wesley Professional, 2005.
- [7] A. Wasserman, M. Pal, and C. Chan, "The business readiness rating model: an evaluation framework for open source," in *Proceedings of the EFOSS Workshop, Como, Italy, 2006*.
- [8] R. Semeteys, "Method for qualification and selection of open source software," *Open Source Business Resource*, no. May 2008, 2008.
- [9] B. Russo, E. Damiani, S. Hissam, B. Lundell, and G. Succi, *Open Source Development, Communities and Quality: IFIP 20th World Computer Congress, Working Group 2.3 on Open Source Software, September 7-10, 2008, Milano, Italy*. Springer Science & Business Media, 2008, vol. 275.
- [10] U. Höhle and S. E. Rodabaugh, *Mathematics of fuzzy sets: logic, topology, and measure theory*. Springer Science & Business Media, 2012, vol. 3.
- [11] X. Xue and X. Yang, "Seismic liquefaction potential assessed by fuzzy comprehensive evaluation method," *Natural hazards*, vol. 71, no. 3, pp. 2101–2112, 2014.
- [12] Y. Yu, H. Wang, G. Yin, and C. X. Ling, "Reviewer recommender of pull-requests in github," in *Software Maintenance and Evolution (ICSM), 2014 IEEE International Conference on*. IEEE, 2014, pp. 609–612.
- [13] Y. Yu, H. Wang, V. Filkov, P. Devanbu, and B. Vasilescu, "Wait for it: Determinants of pull request evaluation latency on github," in *Mining software repositories (MSR), 2015 IEEE/ACM 12th working conference on*. IEEE, 2015, pp. 367–371.
- [14] S. Jamieson *et al.*, "Likert scales: how to (ab) use them," *Medical education*, vol. 38, no. 12, pp. 1217–1218, 2004.

<sup>1</sup>Complete experimental results can be found at: <https://www.trustie.net/projects/4009/boards>

# A Non-Functional Requirements Recommendation System for Scrum-based Projects

Felipe Ramos<sup>§</sup>, Alexandre Costa<sup>§</sup>, Mirko Perkusich<sup>¶</sup>, Hyggo Almeida<sup>§</sup> and Angelo Perkusich<sup>§</sup>

<sup>§</sup>Intelligent Software Engineering (ISE) Group, Federal University of Campina Grande,  
Campina Grande, Paraíba, Brazil, 58429-140

{feliperamos, antonioalexandre}@copin.ufcg.edu.br,  
hyggo@dsc.ufcg.edu.br, perkusic@dee.ufcg.edu.br

<sup>¶</sup>Federal Institute of Paraíba, Monteiro, Paraíba, Brazil, 58500-000  
mirko.perkusich@ifpb.edu.br

**Abstract**—Agile software development focuses on quick delivery and flexibility to change. Despite being effective in delivering quality functional requirements, agile practices tend to neglect non-functional requirements until the later stages of software development. This work focuses on Scrum, the most popular agile method, and presents a non-functional requirements recommendation system to support Scrum practitioners on their early identification. The solution is based on instrumenting the Scrum process to extract useful data and the use of collaborative filtering and item recommendation. To evaluate the recommendations, we conducted off-line experiments with data collected from 12 Scrum practitioners through a survey. The data was analyzed using 10-fold cross-validation. As a result, our proposed solution showed a recall rate of up to 81%, which indicates that it is a promising approach to recommend non-functional requirements given a set of functional requirements identified by project stakeholders.

**Keywords**—Non-functional Requirements; Scrum; Recommendation System; Agile Software Development.

## I. INTRODUCTION

Agile software development (ASD) methods, such as Scrum and XP, have gained strength with the acceptance of the fact that uncertainty is part of software development [4], emerging as an alternative to keep up with the high competitiveness and volatility of the software market. Unlike traditional approaches, which rely on detailed processes and extensive planning, ASD focuses on the rapid delivery of business value to customers. Moreover, ASD supports requirement changes at any stage of the development process [4]. However, as well as in traditional approaches, Requirements Engineering (RE) is a crucial process for the success of agile projects.

ASD methods, in contrast to traditional plan-driven processes, follow an incremental and iterative development process. Even though they are efficient in delivering quality functional requirements (FRs) [7], non-functional requirements (NFRs) are often overlooked until the later stages of software development [10], which might increase the costs [10] and probability of failure [2].

This work focuses on Scrum, the most popular agile method. In Scrum, the requirements are usually managed by a person with a business-oriented profile (i.e., the Product Owner), which tends to focus on FR, neglecting NFRs. [13].

Most existing studies related to applying NFRs processes to ASD [10], [6], [5] do not consider Scrum's artifacts, events

and roles, reducing their applicability. The studies [14], [13], [3] that focus on Scrum present processes to complement it and consider, for instance, modeling NFRs as a “done” criteria or a constraint story [13].

Our goal is to automate the definition of NFRs given historical data of Scrum projects, supporting the team on identifying them early in the process. For this purpose, in this paper, we present a NFRs recommendation system to support Scrum practitioners. It is based on collaborative filtering and item recommendation and supported by an instrumentation of the Scrum process to enable the collection of quality data for the recommendations.

This paper is organized as follows. Section II presents previous works on NFRs applied to the ASD. Section III presents the proposed solution. Section IV presents the design of the validation process. Section V discusses the results; and Section VII presents our conclusions and future work.

## II. RELATED WORK

There are several studies regarding the management of NFRs on ASD [3], [5], [6], [7], [8], [10], [11], [13], [14], [15].

In a study including a series of papers, authors presented solutions for the capture, definition and prioritization of NFRs in ASD. First, authors proposed a NFR modeling framework that is tailored for agile processes, called Non-Functional Requirements Modeling for Agile Processes (NORMAP) [6]. In addition, a simulation tool was proposed for modeling non-functional requirements for semi-automatic agile processes (NORMATIC) [7], which supports the more general NORMAP methodology. In [5], authors proposed a methodology for elicitation, reasoning and validation of NFRs in agile processes (NERV), which showed better results in comparison to the NORMAP framework. NERV is a lightweight methodology to address NFRs early in ASD. In the study [10], the authors proposed to use NFRs metadata from software requirements artifacts - documents and images - as an extension of previous works [6] and [5]. Finally, in [12], authors investigated the prioritization of requirements based on the framework proposed in [11].

Some studies conducted research on the topic of NFRs in the context of Scrum [3], [13], [14].

Bourimi et al. [3] proposed the Agile Framework For Integrating Non-functional requirements Engineering (AFFINE) with the goals of: (1) conceptually considering NFRs early in the development process, (2) explicitly balancing end-users' with developers' needs, and (3) having a reference architecture to support NFRs. The authors introduced the role of an NFR stakeholder. Although the proposed solution presents contributions, the method is based only on a conceptual effort of the early consideration of NFRs.

On the other hand, Sabry and El-Rabbat [13] discussed about architectural refactoring framework and techniques for achieving required levels of NFRs through the formalization of Spikes and Definition of Done (DoD) within Scrum practices.

Sachdeva and Chung [14] proposed a novel approach to handle non-functional requirements of security and performance in Scrum-based projects involving big data and Internet of Things (IoT). In their approach, authors proposed to consider security as system functionalities (set of user stories) and performance as spikes and acceptance criteria of user stories.

Although previous studies focused on the early definition of NFRs in ASD, they based their approaches on conceptual reinforcement or on the automatic capture of NFRs from project documents, which may not always be available at the beginning of agile projects. In this study, we focus on the early definition of NFRs, but unlike previous works, we propose the use of historical (considering Scrum roles, artifacts and events) to generate recommendations of NFRs.

### III. PROPOSED SOLUTION

In this section, we present the NFR recommendation system, which aims to support Scrum practitioners in the early definition of NFRs.

In Figure 1, we present an overview of the recommendation process, which is based on the Scrum instrumentation, presented as follows. The activities take place during Sprint Planning meetings: (1) the Scrum Team, with the support of the Scrum Architect (SA), details product backlog items using Semi-structured User Stories (SUS) (2). The SUSs is used by the recommendation system (3) to generate recommendations of NFRs for each FR; (4) The Product Owner (PO) and the Development Team evaluate the recommendations, accepting or rejecting them. If recommendations are accepted, the information about recommended NFRs is stored. If they are rejected, negative feedback is stored.

#### A. Scrum Instrumentation

We instrument the Scrum framework by adding two new elements, as shown in Figure 2. We added a new role, the **Scrum Architect** (see Figure 2 (1)), and a new artifact based in tags and categories to represent product backlog items: **Semi-structured User Story** (see Figure 2 (3)). We detail the new role and artifact as follows.

##### 1) Semi-structured User Story

In Scrum, user stories are generally used to represent product backlog items. They are composed of three aspects: written description, conversations about their details and a list of acceptance criteria. They usually have the following format:

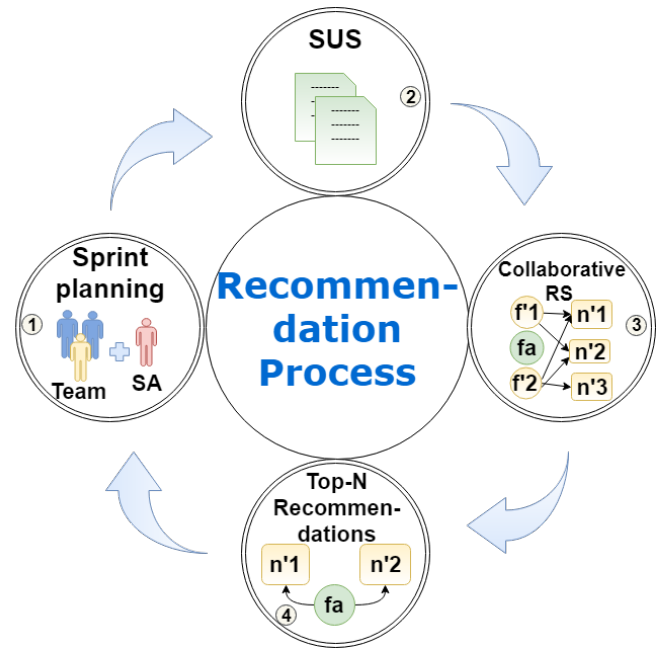


Fig. 1: An overview of the recommendation process.

As a [type of user], I want [some goal] so that [some reason].

This level of information is not enough to retrieve useful data for the recommendation system. Therefore, to ensure the traceability of requirements among different projects, we propose the SUS (see Figure 3).

The SUSs are developed during Sprint Planning meetings and, compared to traditional user story, they contain the following additional information:

- **FR category:** represents a predefined category that classifies a user story based on its goal (high-level FR). For example, “Login”, “Alarm and Notifications”, “Report Visualization”, etc.;
- **Technology tags:** represent labels related to the technologies necessary for the development of a user story. For example, programming language (e.g., java, python, etc.), database (e.g., mongo, sql, etc.), etc.;
- **Associated NFRs:** set of quality attributes that are associated with the functional requirement represented by the user story. Each NFR presents the following information:
  - **NFR category:** represents a predefined category, which classifies the non-functional requirement (e.g., security, performance, privacy, etc.);
  - **NFR statement:** represents the condition of the NFR that must be met to consider the associated functional requirement done. For example, “SSL encryption” for a given FR.

In Figure 3, we present an example of a SUS, in which FR category is “Login” (1), technology tags are “android”, “java” and “mongo” (2), and presents an associated NFR (3) of “Security” (4) with “SSL encryption” as statement (5).

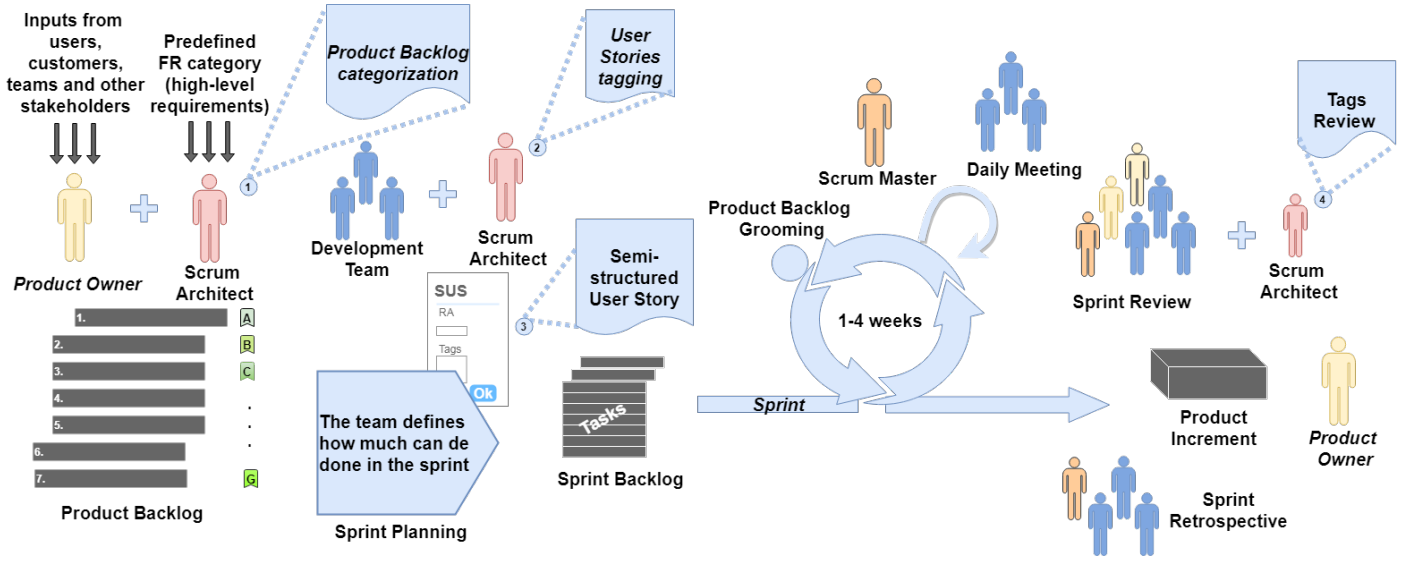


Fig. 2: An overview of the instrumented Scrum process.

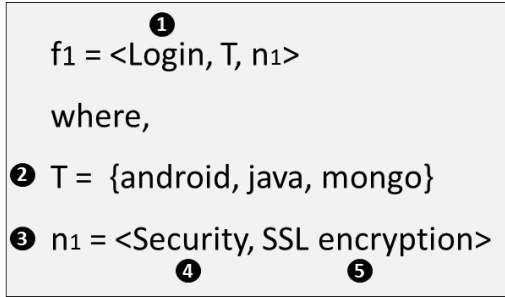


Fig. 3: An example of a Semi-structured User Story.

## 2) Scrum Architect

The SA is responsible for the following three activities: i) categorize product backlog items and gather project information (see Figure 2 (1)); ii) assign tags to the SUSs (see Figure 2 (2)); and (iii) review previously assigned tags (see Figure 2 (4)). The activities are executed at the beginning of the project, during the initial definition of the product backlog and, continuously, in the Scrum Planning and Review meetings at each Sprint.

During the initial definition of the project backlog, the SA is responsible to assign a predefined category for each product backlog item defined by the PO. The decision on the category of each item should be defined in common agreement between PO and SA by face-to-face communication. If a specific requirement can not be classified in any of the predefined categories, the SA can create a new category, insert it into the dataset, and reuse it (i.e., reuse-driven approach). By categorizing product backlog items, we aim to classify requirements of same purpose, and later, to ease the information retrieval. Additionally, the SA is responsible for filling the project profile with information such as project category (e.g., web, mobile, embedded, desktop, etc.), project domain (e.g., health, banking, etc), project goal (e.g., product or prototype), project architecture (e.g., client-server, MVC, multilayered, etc.), and

assign technology tags to the project, which represent the basic technologies needed in the software product development (e.g., programming language, database, etc.).

During Sprint Planning meetings, the SA is responsible for obtaining the remaining information of the SUSs by following the discussions between the PO and the Development Team, ensuring that all product backlog items are categorized and technology tags are properly assigned. Since it is during the Sprint Planning meeting that the Development Team defines all the tasks that must be performed to complete each selected user story for a Sprint, the SA has the chance to accurately obtain the information on which technologies will actually be used to complete each one of them, based on direct feedback from the Development Team.

Finally, during the Sprint Review meeting, the SA is responsible for reviewing the tags assigned to user stories, i.e., to identify if any tag should be removed from a user story representation and/or if new tags should be assigned to it.

## B. NFR Recommendation

We address the definition of NFRs as a recommendation problem. Thus, by adapting the generic definition of recommendation problems presented by Adomavicius and Tuzhilin [1], we have the following: let  $F$  be the set of all FRs and let  $N$  be the set of all possible NFRs that can be recommended. Let  $u$  be an utility function that measures the usefulness of a NFR  $n$  to a FR  $f$ , i.e.,  $u: F \times N \rightarrow T$ , where  $T$  is a totally ordered set. Then, for each FR  $f \in F$ , we want to choose such NFR  $n' \in N$  that maximizes the utility of the FR  $f$ . More formally:

$$\forall f \in F, \quad n'_f = \arg \max_{n \in N} u(f, n). \quad (1)$$

As follows, we present the main components of the NFR recommendation system:

- **Data Collector:** collects information from SUSs and project profiles to generate FR and NFR profiles;
- **Profile Manager:** generates FRs and NFRs profiles based on information collected by the Data Collector;
- **Recommender:** analyzes FRs and NFRs profiles to generate customized NFRs recommendations.

We detail each one of the three components in the following subsections.

### 1) Data Collector

The data collection task consists of extracting information from data sources to represent the elements of the recommendation system to generate their profiles. In our case, the elements are represented by FRs and NFRs. Before extracting the information through the Data Collector, we need to define the features capable of generating representative profiles of the elements in the problem domain. For this purpose, we elicited the knowledge of 3 scrum experts with experience between 4 and 10 years. They identified 5 features of software projects that may influence the definition of NFRs: project category (e.g., web, mobile, embedded, desktop, etc.), project domain (e.g., health, banking, etc.), project goal (e.g., product or prototype), project architecture (e.g., client-server, MVC, multilayered, etc.), and technologies (e.g., java, mongo, etc.).

For data collection, information about the FRs are retrieved from the SUSs and project profiles. More specifically, for each FR, information about FR category (high-level FR) and project profile are collected. For each NFR, NFR category and statements are extracted from the SUSs. Finally, information about the association between FRs and NFRs are collected.

### 2) Profile Manager

After collecting the data through the Data Collector, FR and NFR profiles are generated in the Profile Manager component. The profile of a FR  $f \in F$  is composed by its FR category in addition to the 5 features of project  $p$  which it belongs to. In Table I, we present examples of FR profiles.

TABLE I: Examples of FR profiles.

ID	Proj. Categ.	Domain	Goal	Arch.	Tech. tags	FR categ.
$f_1$	Mob.	Home Auto.	Product	MVC	android, java, mongo	Login
$f_2$	Mob.	Home Auto.	Product	MVC	android, java, sqllite	Alarm and notif.
$f_3$	Mob.	Educat.	Product	MVC	android, jsoup, java, retrofit	Login
$f_4$	Web	Busin. info. sys.	Product	Client-server	nodejs, angular, webstorm	Login
$f_5$	Web	Busin. info. sys.	Prototype	multilayer	nodejs, angular, bootstrap	Status Vis.

NFR profiles present two features, i.e., NFR category and NFR statement. In Table II, we present examples of NFR profiles.

TABLE II: Examples of NFR profiles.

ID	NFR statement	NFR category
$n_1$	SSL encryption	Security
$n_2$	retrieved result must be paginated	Performance
$n_3$	access functionality with less than X clicks	Usability

Finally, FR profiles are complemented with the analysis of co-occurrences between FRs and NFRs based on information from the SUSs. The result of this analysis is a binary matrix. In Table III, we present an example of the matrix, in which the assigned value is 1, if a FR  $f \in F$  has considered a NFR  $n \in N$  or 0, otherwise.

TABLE III: Example of binary matrix that represents the co-occurrence between FRs and NFRs.

	$n_1$	$n_2$	$n_3$
$f_1$	1	0	1
$f_2$	0	0	1
$f_3$	1	0	1
$f_4$	1	0	0
$f_5$	0	1	1

After generating FR and NFR profiles, it is possible to generate customized recommendations of NFRs to FRs.

### 3) Recommender

The proposed recommendation system is based in the following characteristics:

- **memory-based collaborative filtering (neighborhood based technique) [9]:** recommendations are generated from the analysis of historical data on the co-occurrence between FRs and NFRs. To calculate the utility  $u$  of a NFR  $n$  for a FR  $f$ , we evaluate the relation of  $n$  with the  $k$  FRs (from previous projects) most similar to  $f$ ;
- **recommendation of good items [9]:** the proposed recommendation system suggests a list with the  $j$  NFRs best suited to a given FR  $f$ .

To generate NFR recommendations, we carry out two activities: (i) estimate neighborhood using the k-Nearest Neighbors algorithm (kNN); and (ii) generate item recommendations.

For (i), we aim to identify the set of FRs  $\hat{F}$  which includes the most similar FRs to a FR  $f_a$ , where  $f_a$  is the FR we intend to generate recommendations, called **target FR**. We use the kNN method to perform this task, since it returns the  $k$  nearest neighbors of an input element. Before applying the kNN, we perform a pre-filtering in the dataset to retrieve just those FRs of the same category of the target one. Then, the returned list of FRs is used as input of the kNN for the similarity calculation.

Before calculating similarities between FRs, we need to represent each FR profile through an  $m$ -dimensional feature vector, which enables the use of a similarity metric. We generate feature vectors based on the information extracted from the target FR profile, where each vector position refers to its features and is filled with a value (i.e., 0 or 1) according to the following condition: receives the value 1 if the feature is present in the FR profile, or 0, otherwise. In Table IV, we present an example of the vectors generated based on features extracted from a target FR  $f_a$ . In this example, it is possible to see that requirement  $f'_1$  shares the same features of  $f_a$ , since its feature vector is filled only with values equal to 1. On the other hand, the functional requirement  $f'_3$  differs from  $f_a$  in terms of “project domain” (i.e., Home Automation) and technology tag “mongo”.

To calculate similarities among the feature vectors of  $f_a$  and pre-filtered FRs, we use the similarity based on Manhattan

TABLE IV: Examples of feature vectors generated based on features extracted from a target FR  $f_a$ .

	Mobile	Home Auto.	Product	MVC	android	java	mongo
$f_a$	1	1	1	1	1	1	1
$f'_{1}$	1	1	1	1	1	1	1
$f'_{3}$	1	0	1	1	1	1	0
$f'_{4}$	0	0	1	0	0	0	0

Distance (Equations 2 and 3), which is given by the sum of the differences between the values of the two input vectors of same dimension  $m$ . For example, the distance (Equation 2) between  $f_a$  and  $f'_{3}$  is equal to 2, whereas the similarity (Equation 3) between them is 0.71 (see Table IV).

$$d(f_a, f') = \sum_{i=1}^m |f_{a_i} - f'_{i}|, \quad (2)$$

$$sim(f_a, f') = 1 - \frac{d(f_a, f')}{m}. \quad (3)$$

Therefore, to identify the  $k$  nearest neighbors of target FR  $f_a$ , we only have to sort in descending order the list of pre-filtered requirements by the calculated similarity, and return the first  $k$  items from that list. At the end of the process, we have the set with FRs  $f' \in \hat{F}$  most similar to  $f_a$ .

The second activity is to generate item recommendations. More formally, we intend to recommend NFRs  $n' \in N$  that maximize the utility of a target FR  $f_a$ . Thus, the value of the unknown utility  $u_{f_a, n'}$  (Equation 4) for target FR  $f_a$  and NFR  $n'$  is computed as an aggregate of the amount of co-occurrence between the  $k$  neighbors of  $f_a$  and  $n'$ , weighted by the similarity among  $f_a$  and its neighbors, where  $u_{f', n'}$  returns 1 if NFR  $n'$  was considered in the development of neighbor FR  $f'$ , or 0, otherwise.

$$u_{f_a, n'} = \sum_{f' \in \hat{F}} sim(f_a, f') \times u_{f', n'}. \quad (4)$$

Finally, we generate recommendations for target FR  $f_a$  as a list of NFRs  $n' \in N$  sorted in descending order by the utility calculated for  $f_a$  and each NFR  $n'$ , where  $u_{f_a, n'} > 0$ .

#### IV. VALIDATION

To validate our approach, we executed an offline experiment following the instructions presented by Gunawardana and Shani [9], to answer the following research question: is it possible to automate the definition of NFRs in scrum-based projects based on historical data?

##### A. Experiment Goal

As mentioned before, the proposed recommendation system is based on collaborative filtering and item recommendation approaches. In this context, the most suitable metrics for evaluating our RS in an offline evaluation setup are related to precision and recall [9].

Therefore, the main goal of the experiment is to analyze the proposed RS for the purpose of NFRs recommendation

with respect to the *precision and recall metrics* from the point of view of a dataset generated with information provided by Scrum practitioners in the context of Scrum.

##### B. Data Collection Procedure

To perform an offline experiment, it is necessary to use a dataset that corresponds as faithfully as possible to the real data of the problem domain. Thus, we conduct a survey with Scrum practitioners. To guarantee the reliability of the collected data, we only requested information that can be collected with the proposed instrumented Scrum.

The survey was responded by 12 Scrum project managers with 4 to 10 years of experience, who are mostly members of the same software company. At the end of the data collection, we gathered a dataset with the following attributes:

- 31 different software projects profiles, each one with its 5 features described;
- 31 different types of FR categories (e.g., “Login”, “Status visualization”, etc.);
- 47 different types of NFR statements (e.g., “SSL encryption”, “retrieved result must be paginated”, etc.);
- 130 functional requirements instances, where each instance represents a FR of a software project and a set of associated NFRs.

##### C. Offline Experimental Evaluation

In our experiment, we have two independent variables as input source and two dependent variable as output information. The first independent variable is represented by the number  $k$  of neighbors considered in the generation of recommendations, with 5 levels (1, 3, 5, 7 and 10). The second independent variable is represented by the number  $j$  of recommended NFRs that are considered to calculate the metrics, with 6 levels (1, 3, 5, 7, 10 and  $|\hat{N}|$ , where  $\hat{N}$  represents the full list). Our two dependents variable are represented by the precision and the recall of the NFR recommendation achieved through a run based on a set of independent variables.

To perform the experiment we use the 10-fold cross-validation method, which randomly splits the dataset into 10 independent parts, and each part is used once as test set and the remaining as training set. Therefore, we separate the whole data into 90% for training and 10% for testing. Additionally, we repeat the execution for each set of independent variables. Thus, we have an amount of 300 runs.

##### D. Threats to Validity

The data collection was not done continuously during the Sprints as indicated in the Scrum instrumentation, which is a threat to internal validity. The size of the dataset is a threat to external validity, because it is required to have a large dataset to validate memory-based recommendation systems to represent different cases of the domain. We plan to address both threats in future work.



## V. RESULTS AND DISCUSSION

In Figure 4, we present the scatter plot of the two evaluated metrics (i.e., precision and recall), which summarizes the results obtained in the experiments. For example, the averages of precision and recall for the round with  $k = 10$  and  $j = 5$  are 44% and 73%, respectively.

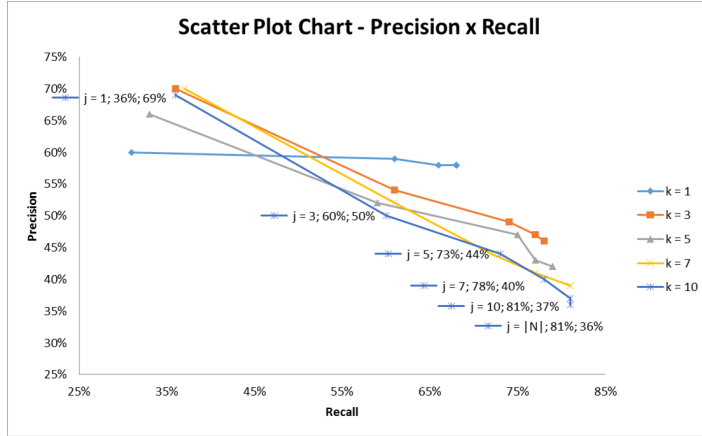


Fig. 4: Results obtained in the experiments.

Considering that the domain of NFRs recommendation is not sparse, since a FR is generally associated to a small number of NFRs, we conclude the results of the offline experiments are promising, since we observe rates of up to 81% of recall in the recommendations ( $k = 7$  and  $j = 10$ ), i.e., we correctly recommended 8 out of 10 NFRs raised in the dataset. Moreover, we observe recall rates greater than 70% and precision rates of up to 49% based on lists of recommendations with a size  $j$  equal to 5 ( $k \in \{3, 5, 7, 10\}$ ), which can be easily handled by Scrum Teams at Sprint Planning meetings and support the early definition of NFRs. We also notice that the ideal number of neighbors  $k$  is greater than or equal to 3.

Finally, we conclude that it is possible to automate the definition of NFRs in scrum-based projects based on historical data. Furthermore, the precision observed in experiments can be improved with a larger dataset, since the proposed recommendation system is memory-based and the quality of recommendations depends on the representativeness of the dataset.

## VI. CONCLUSION AND FUTURE WORK

In this paper, we proposed a NFRs recommendation system to support Scrum practitioners to consider NFRs early in the process. The proposed solution is divided into two main steps, a Scrum instrumentation and a recommendation system. The instrumentation contributes to the data collection process in Scrum-based projects and can be used for different application domains. The NFRs recommender uses historical data from Scrum-based projects, which was not seen in previous works in this research field. Therefore, our work can be used as baseline for future works that intend to investigate this subject.

The offline experimental evaluation showed the feasibility of automating the definition of NFRs through historical data of Scrum projects. We observed an average recall rate of up to 81%, which is promising. Although we observed values of

less than desired precision, we believe that the results can be improved with a larger dataset.

For future work, we intend to keep the data collection process to increase the dataset and improve its representativeness, and then, replicate experiments. Additionally, we intend to carry out a case study with running Scrum-based projects to evaluate recommendations in online environments.

## ACKNOWLEDGMENT

The authors would like to thank CAPES for supporting this work.

## REFERENCES

- [1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE TKDE*, 17(6):734–749, 2005.
- [2] V. Bajpai and R. P. Gorthi. On non-functional requirements: A survey. In *2012 IEEE Students' Conference on EECS*, pages 1–4, March 2012.
- [3] M. Bourimi, T. Barth, J. M. Haake, B. Ueberschär, and D. Kesdogan. AFFINE for enforcing earlier consideration of NFRs and human factors when building socio-technical systems following agile methodologies. *Lecture Notes in Computer Science*, 6409 LNCS:182–189, 2010.
- [4] T. Dingsyr, S. Nerur, V. Balijepally, and N. B. Moe. A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software*, 85(6):1213 – 1221, 2012. Special Issue: Agile Development.
- [5] D. Domah and F. J. Mitropoulos. The nerv methodology: A lightweight process for addressing non-functional requirements in agile software development. In *SoutheastCon 2015*, pages 1–7, April 2015.
- [6] W. M. Farid. The normap methodology: Lightweight engineering of non-functional requirements for agile processes. In *Proceedings of the 2012 19th APSEC - Volume 01*, APSEC '12, pages 322–325, Washington, DC, USA, 2012. IEEE Computer Society.
- [7] W. M. Farid and F. J. Mitropoulos. Normatic: A visual tool for modeling non-functional requirements in agile processes. In *2012 Proceedings of IEEE Southeastcon*, pages 1–8, March 2012.
- [8] A. Firdaus, I. Ghani, D. N. Abg Jawawi, and W. M. N. Wan Kadir. Non functional requirements (NFRs) traceability metamodel for agile development. *Jurnal Teknologi*, 77(9):115–125, 2015.
- [9] A. Gunawardana and G. Shani. A Survey of Accuracy Evaluation Metrics of Recommendation Tasks. *The Journal of Machine Learning Research*, 10:2935–2962, 2009.
- [10] R. R. Maiti and F. J. Mitropoulos. Capturing, eliciting, predicting and prioritizing (cepp) non-functional requirements metadata during the early stages of agile software development. In *SoutheastCon 2015*, pages 1–8, April 2015.
- [11] R. R. Maiti and F. J. Mitropoulos. Capturing, eliciting, and prioritizing (cep) nfrs in agile software engineering. In *SoutheastCon 2017*, pages 1–7, March 2017.
- [12] R. R. Maiti and F. J. Mitropoulos. Prioritizing Non-Functional Requirements in Agile Software Engineering. *Proceedings of the SouthEast Conference on - ACM SE '17*, pages 212–214, 2017.
- [13] A. E. Sabry and S. S. El-Rabbat. Proposed framework for handling architectural nfr's within scrum methodology. In *Proceedings of the International Conference on SERP*, page 238. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2015.
- [14] V. Sachdeva and L. Chung. Handling non-functional requirements for big data and iot projects in scrum. In *2017 7th International Conference on Cloud Computing, Data Science Engineering - Confluence*, pages 216–221, Jan 2017.
- [15] T. Suryawanshi and G. Rao. A Survey to Support NFRs in Agile Software Development Process. 6(6):5487–5489, 2015.

# Analysis of Security Failure-Tolerant Requirements

Michael Shin

Texas Tech University  
Department of Computer Science  
Lubbock, Texas, USA  
michael.shin@ttu.edu

Don Pathirage

Texas Tech University  
Department of Computer Science  
Lubbock, Texas, USA  
don.pathirage@ttu.edu

Dongsoo Jang

Texas Tech University  
Department of Computer Science  
Lubbock, Texas, USA  
dongsoo.jang@ttu.edu

*Abstract* - This paper describes an approach to analyzing security failure-tolerant (SFT) requirements that are specified by means of SFT use cases, along with security use cases and application use cases for application systems. The SFT requirements are analyzed with the analysis model that consists of the static model and dynamic model. A meta-modeling approach is taken to specify the static and dynamic models for analysis of SFT requirements. Threats are identified in the analysis of SFT requirements, and SFT countermeasures against the threats are specified in the analysis model. An online shopping system is used for illustrating our approach.

*Keywords* – *Security Failure-Tolerant Requirements; Analysis of SFT Requirements; Static Model; Dynamic Model; Threat; Meta-Model*

## I. INTRODUCTION

Security services seem to be unbreakable and they can protect security assets in applications from attacks. However, this appearance is not a reality. Security services, such as authentication, encryption or non-repudiation, are incorporated into applications in order to achieve security goals of the applications. Although the applications are designed by means of security services, they are still vulnerable because the security services can always be broken down as attack skills are getting crafty [1, 2].

Several approaches [3, 4, 5, 6, 7, 8, 9, 10] have been developed to make applications secure in software development. Most of the approaches have focused on specifying and designing applications using security services in order to build secure applications. Security requirements are specified with Unified Modeling Language (UML) [11, 12] and its extended notation [3, 4], separately from application requirements [7]. Secure software architecture is designed by means of secure connectors [8, 9, 10, 13] that encapsulate security services. However, less attention has been paid to the tolerance of broken security services.

Security failure-tolerant (SFT) requirements in our previous research [14] are specified to make applications tolerable when security services are breached. SFT requirements are modeled as SFT use cases, along with application use cases and security use cases, against the threats identified in application use cases. The SFT approach aims at reducing the possibility of security damage to security assets in the applications from the breaches of security services.

This paper is an extension of our research [14] by analyzing SFT requirements specification using the analysis model that represents the static and dynamic views of applications. The research in [14] specifies SFT requirements using SFT, security and application use cases against threats. This paper describes the analysis of the use cases and finds new threats, which are not identified in SFT requirements specification, and models the threats in the analysis model. In addition, this paper attempts to develop security and SFT countermeasures against the threats.

This paper is organized as follows: Section 2 describes related work for our research. Section 3 describes SFT requirements specification. Section 4 describes the meta-model for the analysis model of SFT requirements. Section 5 describes the analysis of SFT requirements in terms of the static and dynamic models. Section 6 describes the validation of our approach. Section 7 describes conclusions and future work.

## II. RELATED WORK

Related work focuses on threat modeling, secure software development, and efforts to mitigate security failures so that applications become more secure. There is some research on security, but the research does not provide an adequate solution for developing secure software systems that tolerate the failures of security services.

**Threat Modeling.** Threats in a system have been modeled by several approaches, which include attack trees [1], data flow diagrams [15, 16], and UML-based modeling [17, 18, 19, 20, 21]. Attack trees in [1] provide an approach to modeling and analyzing the threats of systems, and the threats are analyzed in terms of attackers' capabilities. The design models in the research [15, 16] are specified with data flow diagrams, and the threats to the models are identified and analyzed using scenarios of each function in a system. Several threat-modeling approaches, such as misuse cases [17, 18, 19], abuse cases [20] and HAZOP (Hazard and Operability Analysis) [21], have been developed for object-oriented software systems. The approaches model threats using the use case model in UML and specifies security requirements against them.

The misuse case model [17, 18, 19] extends the use case model to misuse cases, along with actions that systems should take to support security requirements. An inverse of a use case is a misuse case, which is a negative requirement of a system that should not occur. The scenario of each possible attack is modeled using a misuse case. A use case description is analyzed to identify misuse cases and their actors.

**Secure Software Development.** Some research for developing secure software has been done in terms of secure requirements and design. The studies in [3, 4] proposed a UML-based modeling language for the model-driven development of secure, distributed systems. The research in [22] illustrates an ontology-based approach that uses predefined pattern-based templates to aid requirements engineers in the formulation of security requirements. Security patterns in [5, 6] address a broad range of security issues that should be taken into account in the stages of the software development lifecycle.

In earlier work by a coauthor in [7], an approach is described to model complex applications by modeling application requirements and design separately from security requirements and design using the UML notation. In later work by a coauthor in [8], an approach is described for modeling the evolution of a non-secure application to a secure application in terms of the requirements and software architecture. The recent work by coauthors in [9, 10] proposes the design of reusable secure connectors using a component-based approach in which reusable secure connectors are structured into reusable security components and communication components.

**Mitigation of Security Failures.** Security failures can be mitigated by several approaches, such as layered security (defense in depth) [23], intrusion tolerance [24], and self-protection [25]. Layered security [23] addresses multiple facets of a security on a network. It is made up of multiple layers of complementary security technologies, so that all the technologies work together to provide the required level of protection.

The study in [24] presents the systematical notion of intrusion tolerance by rearranging the concepts and design of intrusion tolerance. Also the work in [26] presents a way to combine preventive maintenance with an existing intrusion tolerance system to improve the system security.

Self-protection [25] is a part of autonomic computing in which a self-protection component controls the security of a system without human interaction. To defend a system against malicious attacks or cascading failures, a self-protection system automatically prevents the attacks or failures.

### III. SPECIFICATION OF SFT REQUIREMENTS

SFT requirements [14] are specified against threats to application systems. The threats can be identified by considering security assets described in the use case description. The threats are represented using the use case notation in the use case model. A threat use case may not have a specific actor because an attacker can be any malicious persons or parties. A threat point [14] is defined in the use case description for an application use case where a security asset is contaminated if a security service is broken and there is not any SFT service to protect the asset. The threat to *make order request* use case in the online shopping system [27] is modeled in Fig. 1a in which the *release ID and password* threat threatens the *make order request* use case at the *ID and password* threat point.

The requirements of security services for an application system are specified with security use cases [7, 14] separately from application use cases. When an application system requires security services, the security use cases are extended

from the application use cases at extension points. An extension point is a location in an application use case where a security use case extends an application use case if the system requires the security service. An application use case designates an extension point in the use case description where a security use case extends the application use case. The security use case for the *make order request* application use case is depicted in Fig. 1b in which the *check keystroke logging* security use case is extended from the *make order request* application use case if the system requires the *check keystroke logging* security service.

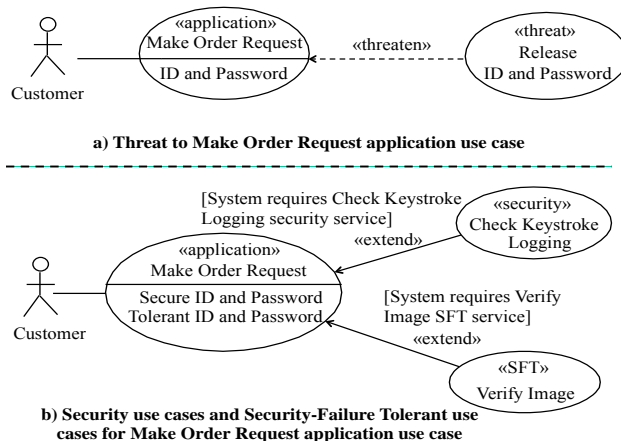


Fig. 1 Threat and Security use case and SFT use case for Make Order Request application use case

SFT requirements for security services are modeled using SFT use cases [14], which tolerate breaches of security services. By careful separation of concerns, SFT requirements are captured in SFT use cases separately from security use cases and application use cases. A SFT use case extends an application use case at an extension point if the system requires the SFT service. The *verify image* SFT use case (Fig. 1b) extends the *make order request* application use case at the *tolerant ID and password* extension point if the system requires the SFT service. The *verify image* SFT use case verifies that an image selected by a customer is matched with the image that the customer registered in the system. Even though the customer ID and password are released to an attacker due to failure of *check keystroke logging* security use case, the attacker would have to know of the customer's image registered in the system in order to make a malicious purchase order.

### IV. META-MODEL FOR ANALYSIS OF SFT REQUIREMENTS

SFT requirements are analyzed from the static model for defining structural relationships between classes and dynamic model for defining how objects participate in use cases. SFT requirements are specified using the use cases that describe the requirements of SFT applications. The static model is developed using the class diagram in UML that determines the classes supporting each use case of SFT requirements and relationships between classes. The dynamic model is developed using the communication diagram in UML that describes the sequences of message communication between objects for each use case.

The static and dynamic models for SFT requirements analysis are depicted in Fig. 2 using a meta-model, which is extended from the meta-model for the class diagram and communication diagram in UML. A meta-model describes the meta-classes and relationships between the meta-classes. Any class diagrams and communication diagrams instantiated from the meta-model for SFT requirements analysis should follow the meta-classes and relationships between the meta-classes defined in the meta-model. The meta-model (Fig. 2) is simplified by representing the underlying meta-classes and their relationships associated with SFT requirements analysis.

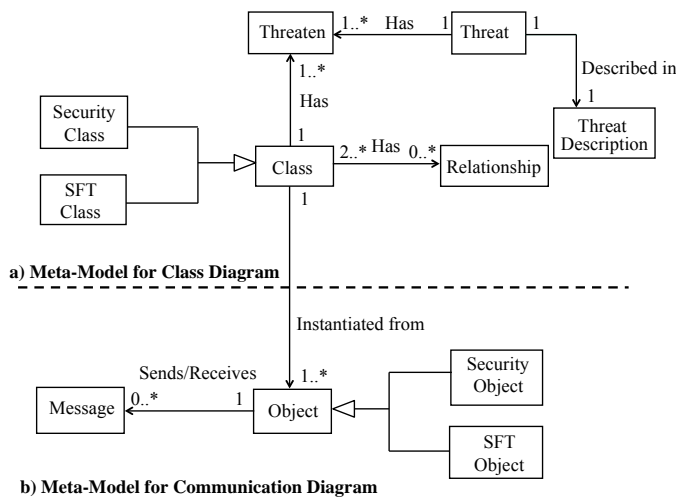


Fig. 2 Meta-model for SFT Requirements Analysis

The meta-model for the class diagram (Fig. 2a) of SFT requirements consists of class meta-class and relationship meta-class. There is a relationship meta-class between class meta-classes. A class meta-class is specialized to security class meta-class or SFT class meta-class. A threat meta-class threatens a class meta-class through a threaten meta-class. A threat meta-class is described in a threat description meta-class. Similarly, the meta-model for the communication diagram (Fig. 2b) of SFT requirements is represented by means of object meta-class and message meta-class in which an object meta-class may send a message meta-class to or receive a message meta-class from other object meta-classes. An object meta-class can be specialized to a security object meta-class or SFT object meta-class. An object meta-class in the meta-model for the communication diagram is instantiated from a class meta-class in the meta-model for the class diagram.

## V. ANALYSIS OF SFT REQUIREMENTS

### A. Static Modeling of SFT Requirements

The static model for SFT requirements defines the structural relationships between application classes, security classes and SFT classes. Application classes support application use cases, whereas security classes are involved in realizing security use cases. SFT classes are needed to implement SFT use cases. The structural relationships between application, security, and SFT classes depict the static view between the classes. The static model for *make order request* application use case (Fig. 1b) is depicted in Fig. 3 using the class diagram, which includes *Customer Interface*, *Customer Account* and *Delivery Order*

classes (Fig. 3) for *make order request* application use case (Fig. 1b), *Keystroke Logging Checker* security class (Fig. 3) for *check keystroke logging* security use case (Fig. 1b), and *Image Verifier* SFT class (Fig. 3) for *verify image* SFT use case (Fig. 1b). The *Customer Interface* class checks keystroke logging attack using *Keystroke Logging Checker* security class, and it verifies the image selected by a customer using *Image Verifier* SFT class.

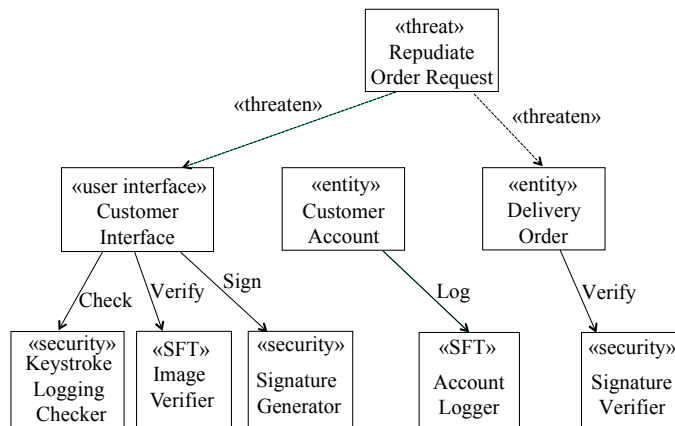


Fig. 3 Static model for Make Order Request application use case

### B. Dynamic Modeling of SFT Requirements

The dynamic model for SFT requirements determines how security objects and SFT objects participate in the sequence of message communication between application objects. Security objects are invoked by application objects if the application objects require security services. Also, application objects invoke SFT objects when they need SFT services in order to tolerate the breaches of security objects. Security objects and SFT objects are represented on the communication diagram along with application objects.

The dynamic model for *make order request* application use case is depicted in Fig. 4, which describes how security and SFT objects are integrated into the sequence of message communication between application objects. The *Keystroke Logging Checker* security object checks malicious keystroke logging software (messages M1.1 and M1.2 in Fig. 4) when a customer initiates an order service (message M1 in Fig. 4). If there is no keystroke logging software installed, *Image Verifier* SFT object verifies whether the image selected by a customer is matched with the customer's image stored in the system (messages M1.3 through M1.8 in Fig. 4). A customer's order request is sent to *Purchase Order Manager* business logic object (messages M2 and M3 in Fig. 4), which requests a customer's account information from the *Customer Account* entity object (message M4 in Fig. 4). The *Purchase Order Manager* business logic object requests the authorization of a customer's credit card payment from a bank via *Bank Interface* object when it receives a customer's account information from the *Customer Account* entity object (messages M5 through M6 in Fig. 4). If the bank approves a customer's credit card payment (message M7 in Fig. 4), *Purchase Order Manager* business logic object stores an order in the *Delivery Order* entity object

(messages M8 and M9 in Fig. 4) and sends a confirmation email to a customer via *Email* entity object (message M10 in Fig. 4).

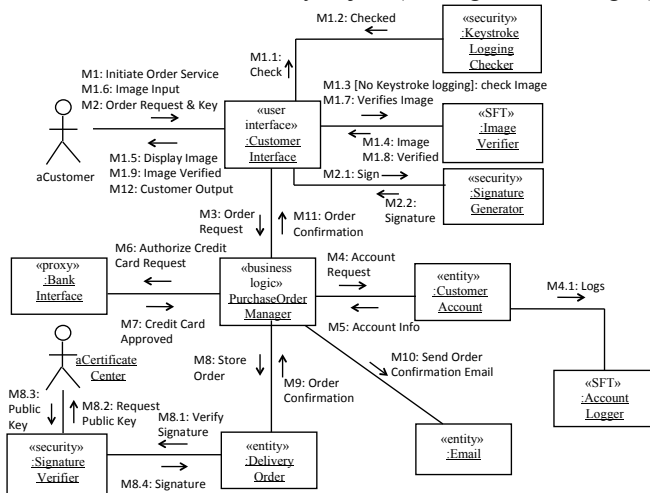


Fig. 4 Dynamic model for SFT Make Order Request application use case

### C. Threat Modeling of SFT Requirements Analysis

Threats in SFT requirements specification focus on the security assets that should be protected from attacks. A security asset can be a security-relevant input to an application, secure data maintained in an application, and a system itself on which an application is running [17]. A security-relevant input to an application is a user’s input to the application or an input from an external system or device to the application in which the inputs require security. An account identification (ID) or password entered by a user to an application can be an example of a security-relevant input to an application. Secure data stored in an application can be a target of an attack. An example of secure data can be the credit card information maintained by an online shopping system or a patient’s medical records stored in a healthcare system. Also, a system on which an application is running should be a security asset when the system’s availability affects the application’s availability.

New threats are found in the analysis of SFT requirements in terms of confidentiality, integrity, or non-repudiation of messages, which can be sent from an object to another. While an object communicates a message with another object, the message might require confidentiality. Also, some messages should not be tampered with in an interaction between objects. In addition, when an object sends a message to another, it might need to prove who sends the message, so as to protect the non-repudiation security.

New threats can be identified by examining the message sequence between objects described in the communication diagrams for SFT requirements analysis. Each use case is refined by means of a communication diagram, which represents the business logic using objects and message sequence between objects. Some messages involved in the business logic might need to be secure from the application perspective. The *order request* (message M2 in Fig. 4) is created by a customer and is sent to the *Delivery Order* entity object through the *Purchase Order Manager* business logic object (messages M3 and M8 in Fig. 4) after the customer’s credit card payment is approved by

a bank. The customer might deny the *order request* later due to this or that reason after the customer made the *order request*. The *order request* message is under a repudiation threat from the application perspective.

The new threats identified in an analysis of SFT requirements are modeled in the static model, which describes application classes, security classes, and SFT classes. A threat is represented by means of the class notation with the stereotype «threat», and a threat class has a «threaten» dependency relationship with a class (Fig. 2). The *Repudiate Order Request* threat is modeled in the static model (Fig. 3) in which the *Repudiate Order Request* threat threatens the *Customer Interface* class and the *Delivery Order* entity class. This means that the *Customer Interface* object and the *Delivery Order* entity object in the dynamic model (Fig. 4) are under the *Repudiate Order Request* threat.

Each new threat is defined in a short threat description that describes threat name, description, security asset, security goal, security class, and SFT class. The security class and SFT class are the security countermeasure against the new threat. An alternative to a short threat description, a threat can be analyzed and specified in detail in terms of threat attributes, threat effect, and security concern [2]. The following is the short threat description for the *Repudiate Order Request* threat:

- Threat Name: Repudiate Order Request
- Description: Customer can repudiate the order request.
- Security Asset: Order Request
- Security Goal: Non-repudiation of order request
- Security class:
  - Signature Generator security class
  - Signature Verifier security class
- SFT class: Account Logger SFT class

As the analysis model of SFT requirements reveals a new threat, a security service and its SFT service against the threat are incorporated into the analysis model. The security and SFT services are modeled through classes in the static model. A digital signature security service is taken against the *Repudiate Order Request* threat, being implemented using the *Signature Generator* security class (Fig. 3), which signs a customer’s order request using the customer private key, and the *Signature Verifier* security class (Fig. 3), which verifies the order request signed by the customer using the customer’s public key. Also, an *Account Logging* SFT service is created against the *Repudiate Order Request* threat, which is mitigated by the *Account Logger* SFT class (Fig. 3) to record a customer’s access to the customer account.

Security and SFT objects against a threat identified in the analysis of SFT requirements are incorporated into the dynamic model. The *Signature Generator* security object signs a customer’s order request using the customer private key (messages M2.1 and M2.2 in Fig. 4) before the *order request* is sent by the *Customer Interface* object to the *Purchase Order Manager* business logic object. The signed order request is verified by the *Signature Verifier* security object (messages M8.1 through M8.4 in Fig. 4) using the customer public key obtained from a certificate authority before the *Delivery Order*

entity object creates a new order request. Also, the *Account Logger* SFT object logs a customer’s access to the *Customer Account* entity object (message M4.1 in Fig. 4) for proving the customer’s transaction later.

## VI. VALIDATION

### A. Implementation

The dynamic model for the *make order request* application use case (Fig. 4) was implemented along with the *keystroke logging checker*, *signature generator*, *signature verifier* security objects, as well as *image verifier* and *account logger* SFT objects. The *make order request* application use case was implemented on an online hosting sever to generate real world results. The *keystroke logging checker* security object (Fig. 4) was implemented as an antivirus agent, which was developed in C# for this paper and ran on a customer’s local machine once every 24 hours. The antivirus agent updates the security status of the customer’s local machine in an online database. The antivirus agent checks if the client’s machine installed anti-keystroke logging software, such as Symantec Endpoint Protection. The online shopping system connects to the online database and checks the security status of the client’s machine, which has been updated by the *antivirus agent*. If the database indicates that the client’s machine is secure from keystroke logging attack, the system allows the customer to proceed with the order. If the system detects that the client’s machine does not have an *antivirus agent* or does not have up to date anti-keystroke logging software, it displays an appropriate warning message and stops the *make order request* use case. However, no SFT service was implemented to protect the database. We assumed that the database would remain secure for the simplicity of our approach.

The *Image Verifier* SFT object (Fig. 4) accesses an image repository in the online shopping system and displays a random set of four images, including a customer’s personal image. The system verifies that the image selected by the customer is matched with the customer’s personal image stored in the system. If the customer chooses a wrong image, the system prompts a different set of images. If the customer selects an incorrect image consecutively twice, the customer’s account is locked.

The *signature generator* and *signature verifier* security objects were implemented for the non-repudiation security

service. The *signature generator* security object computes a signature for an order request using a *secure hash algorithm 1* (SHA1) to generate a hash value, followed by an encryption of the hash value using the customer’s private key. The *signature verifier* security object verifies that the signature is correct for the order request using the customer’s public key.

The *account logger* SFT object (Fig. 4) was implemented to tolerate the breach of a digital signature. The *account logger* SFT object logs the customer’s activities, including the transaction time, customer’s information, and order details. The log data is used later to confirm the validity of the order request.

### B. Performance Analysis of SFT Requirements

This section describes the performance analysis of our approach to see how much performance overhead occurs due to SFT use cases that are specified for application and security use cases. The computational performance of our approach was measured using three approaches: (1) with *standalone application object approach*, for running the application use case with no security; (2) with *security object approach*, for running the application use case with security objects; (3) with *security object and SFT object approach*, for running the application use case with security objects and SFT objects.

The performance was evaluated by measuring the average time taken to complete the execution of three approaches, which were run 20 times each to calculate the average execution time, so that the performance evaluation would not be dependent on a few exceptional running times. The average execution time was calculated by measuring the run time of the program per session, but it excluded the time that a customer interacted with the system. Also, we assumed that the *antivirus agent* was already installed on the client’s machine. Table 1 shows the average execution time of the approaches and performance comparison.

The second column of Table 1 shows that the average execution time is 1.33 seconds for the *make order request* application use case (Fig. 4). The third column of Table 1 shows that the average execution time for the *make order request* application use case with *security objects* (Fig. 4) is 1.57 seconds. The fourth column shows that the corresponding average execution time for application use cases with both security and SFT objects takes 1.64 seconds.

Table 1. Average execution time of approaches and performance comparison

Application use case	With standalone application object approach	With security object approach	With security object and SFT object approach	Time difference between with standalone application object approach and with security object approach	Time difference between with standalone application object approach and with security object and SFT object approach	Time difference between with security object approach and with security object and SFT object approach
Make order request use case (Fig. 4)	1.33 s	1.57 s	1.64 s	0.24 s ≈ 18%	0.31 s ≈ 23%	0.07 s ≈ 3%

The fifth column of Table 1 indicates that there is the time difference between the *with standalone application object approach* and the *with security object approach*. Time difference for the *make order request* application use case is 0.24 seconds (17 %) because the *with security object approach* provides the application use case with security services. The security services in the *with security object approach* consume 17% more processing time for logging account, and generating and verifying a digital signature in the system; the *with standalone application object approach* is faster because it provides no security services.

Similarly, when an application use case is deployed with both security and SFT objects, the average execution time is increased further, as shown in the sixth column of Table 1. The *make order request* application use case (Fig. 4) with security and SFT objects takes 0.31 seconds, which is a 23% increase in run time.

The last column indicates that the time difference between *with security object approach* and *with security object and SFT object approach* is 0.07 seconds for the *make order request* application use case. This result indicates that *with security object and SFT object approach* takes more execution time (3%). However, the *with security object and SFT object approach* makes the system more secure compared to the *with security object approach* alone.

## VII. CONCLUSIONS AND FUTURE WORK

This paper has described an approach to analyzing SFT requirements. SFT requirements were analyzed by means of the analysis model, which was represented using the class diagram and communication diagram. The meta-model for the class diagram and communication diagram was developed to specify the static and dynamic models for an analysis of SFT requirements. New threats were identified in the analysis of SFT requirements, and security and SFT objects against the threats were specified in the analysis model. Our approach can be used by requirements engineers to specify security requirements for applications, as well as SFT requirements against the failures of security requirements.

This paper can be strengthened with further research. The SFT requirements specification and analysis can be extended to the SFT software architecture that describes the components and their interaction for SFT applications. New threats could be identified in the SFT software architecture and, if so, it is necessary for both security and SFT services to be incorporated into the software architecture. Also, we can investigate how both security services and SFT services are encapsulated in secure connectors [8, 9, 10, 13], along with communication patterns.

## REFERENCES

[1] B. Schneier, "Attack trees: Modeling security threats," Dr.Dobbs Journal, pages 21–29, December 1999.

[2] M. E. Shin, S. Dorbala, and D. Jang, "Threat Modeling for Security Failure-Tolerant Requirements", ASE/IEEE International Conference on Privacy, Security, Risk and Trust (PASSAT2013), Washington D.C., USA, 2013.

[3] T. Lodderstedt, D. Basin, J. Doser, "SecureUML: A UML-Based Modeling Language for Model-Driven Security", Fifth International Conference on the Unified Modeling Language, London, UK., 2002.

[4] J. Jürjens, "UMLsec: Extending UML for Secure Systems Development", Fifth International Conference on the Unified Modeling Language, London, UK, 2002.

[5] M. Schumacher, E. B. Fernandez, D. Hybertson, F. Buschmann, and P. Sommerlad, "Security Patterns", Wiley, 2006.

[6] E. B. Fernandez, "Security Patterns in Practice", Wiley, 2013.

[7] H. Gomaa and M. E. Shin, "Modeling Complex Systems by Separating Application and Security Concerns", 9th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2004), Italy, April, 2004.

[8] M. E. Shin and H. Gomaa, "Software Modeling of Evolution to a Secure Application: From Requirements Model to Software Architecture", Science of Computer Programming, Volume 66, Issue 1, pp. 60-70, 2007.

[9] M. E. Shin, B. Malhotra, H. Gomaa, and T. Kang, "Connectors for Secure Software Architectures", 24th International Conference on Software Engineering and Knowledge Engineering (SEKE'2012), San Francisco, July 1-3, 2012.

[10] M. E. Shin, H. Gomaa, D. Pathirage, C. Baker, and B. Malhotra, "Design of Secure Software Architectures with Secure Connectors", International Journal of Software Engineering and Knowledge Engineering, Vol. 26, No. 5, pp 769-805, 2016.

[11] G. Booch, J. Rumbaugh, and I. Jacobson, "The Unified Modeling Language User Guide", Second Edition, Addison Wesley, Reading MA, 2005.

[12] J. Rumbaugh, G. Booch, and I. Jacobson, "The Unified Modeling Language Reference Manual (2nd Edition)", Addison Wesley, Reading MA, 2004.

[13] M. E. Shin, H. Gomaa, and D. Pathirage, Model-based Design of Reusable Secure Connectors," 4th International Workshop on Interplay of Model-Driven and Component Based Software Engineering (ModComp2017), September 17, Austin, Texas, USA, 2017.

[14] M. Shin and D. Pathirage, "Security Requirements for Tolerating Security Failures," 29th International Conference on Software Engineering and Knowledge Engineering, Pittsburgh, USA, July 5-7, 2017.

[15] P. Torr, "Demystifying the Threat-Modeling Process," IEEE Security and Privacy, vol. 03, no. 5, pp. 66-70, Sept/Oct, 2005.

[16] M. Abi-Antoun, D. Wang and P. Torr, "Checking Threat Modeling Data Flow Diagrams for Implementation Conformance and Security", ASE 2007, 21 pages, 2006.

[17] G. Sindre and L. Opdahl, "Eliciting Security Requirements with Misuse Cases," Requirements Engineering, Volume 10 Issue 1, January 2005, pp. 34 - 44.

[18] P. Hope, G. McGraw, and A. I. Anton, "Misuse and Abuse Cases: Getting Past the Positive," IEEE Software, 2003.

[19] I. Alexander, "Misuse Cases: Use Cases with Hostile Intent," IEEE Software, vol.20, no. 1, pp. 58-66, 2003.

[20] J. McDermott and C. Fox, "Using Abuse Case Models for Security Requirements Analysis," In Proceedings of 15th Annual Computer Security Applications Conference (ACSAC'99), pp. 55-64, Phoenix, Arizona, December, 1999.

[21] T. Srivatanakul, "Security Analysis with Deviatonal Techniques," PhD thesis, Department of Computer Science, University of York, UK, 2005.

[22] D. Olawande, G. Sindre, and T. Stalhane, "Pattern-based security requirements specification using ontologies and boilerplates", IEEE Second International Workshop on Requirements Patterns (RePa), 2012.

[23] S. Gantz, "Layered Security Architecture: Establishing Authentication, Authorization, and Accountability", securityarchitecture.com/docs/, 2008.

[24] P. E. Verissimo, N. F. Neves, and M. P. Correia, "Intrusion-Tolerant Architectures: Concepts and Design", Architecting Dependable Systems, Springer-Verlag, Berlin, Heidelberg, 2003.

[25] P. Horn, "Autonomic Computing: IBM's Perspective on the State of Information Technology", [http://people.scs.carleton.ca/~soma/biosec/readings/autonomic\\_computing.pdf](http://people.scs.carleton.ca/~soma/biosec/readings/autonomic_computing.pdf), 2001.

[26] Iman El Mir, D. S. Kim, and A. Haqiq. "Security modeling and analysis of a self-cleansing intrusion tolerance technique." IEEE 11th International Conference on Information Assurance and Security (IAS), 2015.

[27] H. Gomaa, "Software Modeling and Design: UML, Use Cases, Patterns, and Software Architectures", Cambridge University Press, 2011.

# An Approach for System of Systems Requirements Management

Renata M. de Lima, Lisandra M. Fontoura  
Programa de Pós-Graduação em Ciência da Computação, UFSM. Santa Maria, RS, Brazil  
{rlima, lisandra}@inf.ufsm.br

**Abstract**—Systems of Systems (SoS) are becoming more common in our society due to the diversity of stakeholders, heterogeneous technology, ubiquity, or large-scale networks involved in the development of software intensive systems. Consequently, there is an increasing interest in closing the gaps still existing in the System of Systems Engineering field, which is in constant growth due to its complexity. Most of these gaps are related to the SoS requirements management, the lack of well-defined and standardized methods and techniques for Requirements Engineering for SoS makes difficult the development and the evolution of the overall SoS. In this paper we propose an approach to requirements management for SoS through a process capable of organizing the work among stakeholders and managing changes in requirements, thus contributing to a more adaptable evolutionary development of the SoS. The results of a conformity assessment show that our approach is in conformity with the most of international standards related to Systems Engineering and Requirements Engineering.

**Keywords**—System of Systems; SoS; Capability Objectives; Requirements Management; Requirements Engineering.

## I. INTRODUCTION

In different application domains, it is possible to find different systems working together to satisfy a specific goal. This set of Constituent Systems (CS) collaborating with each other establish a System of Systems (SoS). SoS is “a set of systems that results when independent and useful systems are integrated into a larger system that delivers unique capabilities” [4]. The structural differences and the diversity of stakeholders in a SoS development lead to complexities and challenges unexplored by the SE [7].

Several studies indicate that there is a real need for a requirements management that can facilitate the collaborative work among the stakeholders and that can support the evolution of SoS in a flexible and adaptable way [7]. A well-defined management process for SoS requirements would be capable to clarify the management relationships among CS and SoS stakeholders, facilitating the collaboration among them in the life cycle of the SoS development. Thus, more consistent management options would also facilitate the Requirements Engineering for SoS activities, ensuring a more adaptable SoS development. In this paper, we present an approach for SoS requirements management, which aims to organize the work among the stakeholders and the activities of the RESoS.

## II. PROCESS DESCRIPTION

The proposed approach consists of a Requirements Management Process for SoS (RMP-SoS) organized in three activities: Manage Stakeholders, Elicit SoS Requirements and Manage Changes & Validation. The process can be implemented through the production of the documentation using templates and following the workflow (illustrated in Figure 1). The RMP-SoS is more adequate to be applied in Directed SoS, because this type is built with a specific purpose, so even though the CS can operate independently, they are subordinated to a central managed purpose [8].

### A. Activity 1: Manage Stakeholders

The first group of tasks aims to identify who are involved in the RE activities for SoS, as well as their abilities, knowledge and assign roles and responsibilities to the stakeholders, develop a work plan for the next activities, and facilitate the agreements among stakeholders.

1) *Identify Stakeholders*: This task has the main goal to identify the main people who are involved in the requirements activities related to the SoS and to the CS and describe their abilities and knowledge. The responsible for this task is the SoS Requirements Manager and the document generated is the Stakeholders Register.

2) *Assign Roles and Responsibilities*: The purpose of this task is to distribute roles and responsibilities among the registered stakeholders. Roles and responsibilities may vary over the SoS evolutionary development. The responsible for this task is the SoS Requirements Manager, and the artifact created is the Stakeholders Roles and Responsibilities.

3) *Establish Collaboration Agreements*: This task aims to generate documents that express agreements and the level of collaboration among stakeholders from CS and SoS involved in the RE activities. In some ER tasks, it is essential that the SoS requirements engineer get involved with the CS requirements engineer to understand the nature of the CS changes and evaluate impacts on SoS [4]. To meet this need, it is necessary that agreements about sharing data and information be clearly established. The SoS Requirements Manager is responsible for this task and produce the Collaboration Agreements.

4) *Prepare RE Work Plan*: In order to prepare and authorize the beginning of the RE activities, it is necessary to provide a summary of the work that will be done. It is needed to define the SoS type, specify the project (SoS) scope and context, the work purpose for this project phase, the constraints, the requirements traceability policies, as well as the list of stakeholders and their responsibilities [1]. The SoS Requirements Engineer is responsible for the RE Work Plan.



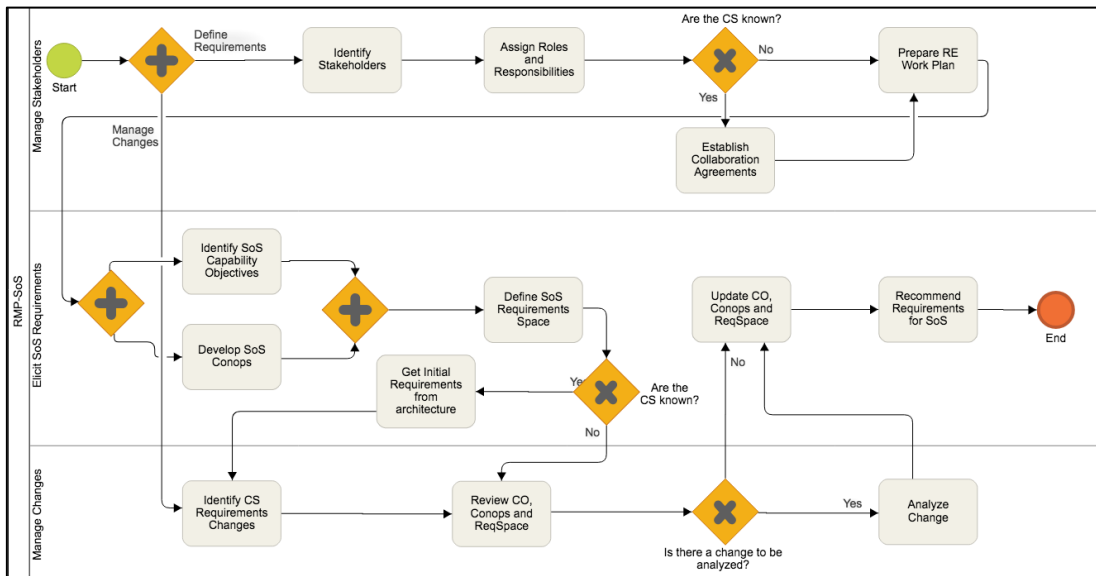


Figure 1 - BPMN Diagram - Workflow of the RMP-SoS

### B. Activity 2: Elicit SoS Requirements:

This second activity has the purpose to organize all the tasks for RESoS in order to generate a set of requirements ready for incorporation to a future functional baseline for SoS.

1) *Identify SoS Capability Objectives (SoS CO)*: It aims to identify the capability objectives for SoS, describing in the users language primary and secondary missions, the operational constraints, and a general vision of the operational environment [1]. The responsible is the SoS Requirements Engineer, and the artifact is the SoS Capability Objectives.

2) *Develop SoS Concept Of Operation (SoS Conops)*: This task has the goal to describe the way the SoS functions works by the user's point of view [3]. SoS Conops document is developed by the SoS Requirements Engineering with actively participation of the SoS Users, who will describe the way they are planning to use the SoS to meet their objectives.

3) *Define SoS Requirements Space (SoS ReqSpace)*: In order to limit the scope of SoS users primary needs and define functionalities to promote the capabilities. And considering the users environment variability, which impacts the way those functionalities will be executed; the SoS Engineering team develop the SoS Requirements Space document [2]. The responsible for this task is the SoS Requirements Engineer.

4) *Get Initial Requirements from Architecture*: This task has the goal to obtain the needed requirements to provide a structure that allows changes in the SoS functionalities over its evolution. SoS architecture is itself a requirements generator, because when SoS engineers develop an architecture, they need to overlap the CS in a structured way to make them work together and share data. However, sometimes this structure can be different from the original design of the CS.

Hence, some changes in the CS may be necessary to support the SoS architecture [4]. In addition, after an architectural construction or simulation, if the designer has applied a good strategy, it is possible to identify and predict emergent behaviors, which will be interesting for the

definition of requirements. SoS Requirements Engineer is the responsible, with intensive collaboration of the CS Requirement Engineers. The artifact is known as SoS Initial Architecture Requirements.

5) *Update SoS CO, SoS Conops and SoS ReqSpace*: This task has the main goal to update the documents based on the corrections recommended by the Review Record, or include/adapt an alteration approved by the Change Record from the task Review SoS CO, SoS Conops and SoS ReqSpace. The responsible is the SoS Requirements Engineer, and the artifacts are SoS Capability Objectives, SoS Conops and SoS Requirements Space documents updated.

6) *Recommend Requirements for SoS*: This task aims to define the requirements that will be addressed in the next SoS increment. The CS Requirements engineers work together with the SoS Requirements Manager and Engineer to prioritize and recommend relevant requirements for each iteration and produce the SoS Recommended Requirements. Then, the selected requirements are moved to the functional baseline to be treated and addressed by the activities associated to the Core "Addressing Requirements & Solution Options".

### C. Activity 3: Manage Changes

The tasks of this activity aim to review the documents for validation and analyze problems or change requests.

#### 1) *Identify CS Requirements Changes*:

To identify changes in the CS requirements, SoS RE team needs to connect to the process that manage the CS requirements, and identify the changes that are capable of affecting its capability objectives. The evolution of SoS is directly related to the evolution of its CS [4]. Therefore, it is important to keep a record of the changes and evolution occurring in the CS and their impact on SoS. The SoS Requirements Engineer and SC Requirements Engineers perform this task. The document generated is CS Information which impacts SoS Requirements.

### 2) *Review SoS CO, SoS Conops and SoS ReqSpace:*

Meetings among key stakeholders are performed to review and validate the SoS Capability Objectives, the SoS Conops, and SoS Requirements Space. All the key stakeholders may participate, but the responsible is the SoS Requirements Engineer. This task can generate a Change Request that is a problem identified or a specific change purpose, which will be analyzed in the sequence. Also, it can generate a Review Record that points out necessary corrections to the documents.

### 3) *Analyze Change Request*

The task begins with a critical analysis of a problem identified by the Review SoS CO, SoS Conops and SoS ReqSpace, or of a specific proposal of change. The Change Control Board performs an analysis of the problem or proposal in order to verify its validity. If it is valid, the effect and impact in terms of cost for the project is evaluated. Once the analysis is done, the decision about to implement the change is taken. The Change Control Board is responsible for generating a Change Record or a new Change Request.

## III. CONFORMITY ASSESSMENT

A conformity assessment is a demonstration that specific requirements related to a product, process, system, person or organizations are attended. This section summarizes the assessment conducted to identify where the RMP-SoS is in conformity with the international standards ISO|IEC|IEEE 15288 [8] and ISO|IEC|IEEE 29148 [5].

ISO|IEC|IEEE 15288 establishes a common framework of process descriptions for describing the life cycle of systems created by humans, defining a set of processes and associated terminology. These processes can be applied at any level of hierarchy of a system's structure. Its Annex G brings an application of these system life cycle processes to a SoS.

The Agreement Processes (G.3.2) says "they are crucial for SoS because they establish the modes of developmental and operational control among the organizations responsible for the SoS and the independent CS". The RMP-SoS has an Establish Collaboration Agreements task, which aims generate collaboration agreements to express how stakeholders will collaborate with each other in a way that the operational management independence of the CS is maintained.

Organizational project-enabling processes (G.3.3), "in a SoS, the owners of a CS usually retain responsibility for engineering their systems as they each have their own processes. The SoS organization implement these processes for the SoS for those considerations that apply to the overall SoS". For example, RMP-SoS is concerned with this topic knowing all the stakeholders in the SoS requirements activities and assigning them with roles and responsibilities adequately through the tasks Identify Stakeholders and Assign Roles and Responsibilities respectively.

Technical Management Process (G.3.4) "is applied to the particular considerations of SoSE - planning, analyzing, organizing and integrating the capabilities of a mix of existing and new systems into a system of systems capability". Also, "the SoS organization must plan an integrated life cycle that recognizes the independent changes in the CS, in addition to

the SoS-initiated changes in a life cycle that threatens the SoS as the system-of-interest". In this context, we consider that RMP-SoS uses some artifacts produced across the life cycle and the evolution process of a SoS. For example, we consider the SoS Architecture as generator of requirements, even though its design is out of the RMP-SoS scope. Also, our proposed process can be integrated in a life cycle that considerate changes in the CS through the task Identify CS Requirements Changes.

Finally, Technical Processes (G.3.5) are concerned with technical actions throughout the life cycle", and some of them can be related to SoS. These technical processes are directly related to requirements and they are addressed in our tasks related to the RESoS in the Elicit SoS Requirements pool.

ISO|IEC|IEEE 29148 (2011) provides a unified treatment of the processes involved in RE throughout the life cycle of systems and software. For Stakeholder requirements definition process (6.2) "the purpose is to define the requirements for a system that can provide the services needed by users and other stakeholders in a defined environment". In this sense, RMP-SoS attends all the activities proposed, because we are able to Elicit Stakeholder Requirements (6.2.3.1) through the task Identify Stakeholders, where we can identify the individual stakeholders and their interest in the SoS throughout its life cycle. Also, through the task Identify SoS Capability Objectives we are able to elicit stakeholders requirements, that is the capability objectives, from the identified stakeholders and establish the traceability of the those requirements.

In order to Define stakeholder requirements (6.2.3.2) our process uses the task Develop SoS Conops, to "define a representative set of activity sequences to identify all required services that correspond to anticipated operational and supported scenarios and environments" and "identify the interaction between the users and the system", by means of describing the operational environment and the operational scenarios in the SoS operational users view.

The issues related to Analyze and maintain stakeholders requirements (6.2.3.3) are addressed in part by our task called Review SoS CO, SoS Conops and SoS ReqSpace, which reviews the documents generated by the tasks mentioned above and make sure that there is no conflicting, missing, incomplete, ambiguous, inconsistent or unverifiable requirements. If there is some problem, the task generates a Review Record or a Change Request in order to fix this.

In Requirements Analysis Processes (6.3) the purpose is "to transform the stakeholder, requirement-driven view of desired services into a technical view of a required product that could deliver those services". RMP-SoS addresses this recommendation by the definition of the SoS Requirements Space. In order to Define system requirements (6.3.3.1), the task Define SoS Requirements Space aims to limit the primary needs of the stakeholders and define the functions to provide those needs achieving the traceability between capabilities and functionalities. Analyze and maintain system requirements (6.3.3.2) is also addressed in part by the task Review SoS CO, SoS Conops and SoS ReqSpace to ensure that the specified system requirements adequately reflect the capability

objectives to address the stakeholders needs and expectations.

The standard also emphasizes that there are RE activities in other technical processes. In Requirements in architectural design (6.4.1), the Definition of the architecture (6.4.1.1) is out of the scope of the RMP-SoS. However, we perform the Analyze and evaluate the architecture (6.4.1.2) in a way that is possible to identify which integration requirements are missing by the tasks Get Requirements from Architecture and in part by the Identify CS Requirements Changes.

About Requirements in verification (6.4.2), the purpose is to confirm that the specified design requirements are fulfilled by the system. For SoS, these issues related to the verification should be done in execution time, and the activities are the responsibility of the processes related to the Core Elements “Assessing Performance to Capability Objectives” and “Addressing Requirements and Solution Options”. The RMP-SoS only provides input for defining and conducting a verification plan based on requirements. The same assumption is valid for Requirements in validation (6.4.3).

Finally, the clause related to Requirements management (6.5) talks about the importance of “establish procedures for defining, controlling, and publishing the baseline requirements for all levels of the system-of-interest”. Our RMP-SoS is totally about that, all the tasks pursue to identify, define, maintain and prioritize consistent requirements during the increments along the SoS life cycle.

About Change management (6.5.2) the standard says “change has to be managed by ensuring that proposed changes go through a defined impact assessment, review, and approval process, and by applying careful requirements tracing and version management”. These issues, in the RMP-SoS are treated by the tasks Analyze Change and Update SoS CO, SoS Conops and SoS ReqSpace. The first one analyses and accesses the validity and the impact of the change in SoS requirements. And the second one implements the change if it was approved, or implement a correction pointed by the Review SoS CO, SoS Conops and SoS ReqSpace.

## V. DISCUSSION

Considering the conformity assessment presented, we assume that our proposed process for SoS requirements address the most of the recommendations suggested by the two International Standards related to its field. It is important to emphasize that there is no standard specific developed for SoSE, and that until now the unique way is the application of existing standards, such as ISO/IEC/IEEE 15288 [3].

In the case of ISO/IEC/IEEE 29148, we had to adapt some concepts in order to make them applicable for SoS. For instance, the stakeholder requirements mentioned can be considered our “capability objectives” in the SoS context. Some processes presented by this standard we understood that they are out of scope of this SoS requirements management. For example, Measurement for requirements (6.5.3) presents activities we understand that is part of the processes related to the Core Element Assessing Performance to Capability Objectives, even though RE could benefit from this process, it is not directly related to the tasks of the RMP-SoS.

In addition, the Information items (7), define three documents as part of the RE processes. However, in the SoS context these documents, even they are similar and have almost the same proposal, they have different names and different ways to be completed. For instance, the Stakeholder requirements specification document (StRS) would be our SoS Capability Objectives. In the same way, the System Requirements Specification document (SyRS) would be the union of our SoS Conops and SoS Requirements Space translated into the SoS Requirements document.

## VI. CONCLUSION AND FUTURE WORK

In this paper we presented a Requirements Management Process for System of Systems (RMP-SoS), which aims to organize the RE activities and the work among stakeholders, by integrating techniques and methods from project management and traditional requirements engineering, providing support for translating capability objectives and the definition of requirements and incorporating changes over the evolutionary development of SoS. As a result of a first conformity assessment, we understand that the process is in conformance with the international standards of systems life cycle processes and requirements engineering.

As future works we intend to improve templates to support the RMP-SoS documentation and develop an online tool to help the application of the process in real projects, which will facilitate the communication among stakeholders and ensure the geographical distribution of the CS is not a problem. After that, the process will be empirically evaluated in a real project.

## VII. ACKNOWLEDGMENT

We gratefully acknowledge financial support from CAPES-Brazil, and the Brazilian Army through the SIS-ASTROS Project (813782/2014), developed in the context of the PrgEE-ASTROS 2020.

## REFERENCES

- [1] ARNAUT, B. M., Ferrari, D. B., & e Souza, M. L. D. O. (2016, October). A requirements engineering and management process in concept phase of complex systems. In *Systems Engineering (ISSE), 2016 IEEE International Symposium on* (pp. 1-6). IEEE.
- [2] DAHMANN, J., Rebovich, G., Lane, J. A., & Lowry, R. (2011). System engineering artifacts for SoS. *IEEE Aerospace and Electronic Systems Magazine*, 26(1), 22-28.
- [3] DAHMANN, Judith; ROEDLER, Garry. Moving towards standardization for system of systems engineering. In: *System of Systems Engineering Conference (SoSE), 2016 11th. IEEE, 2016. p. 1-6.*
- [4] DoD-USA (2008). *Systems Engineering Guide for Systems of Systems*. Washington, DC.
- [5] International Organization for Standardization. 2011. *ISO/IEC/IEEE 29148—Systems and Software Engineering—Life Cycle Processes – Requirements Engineering*.
- [6] International Organization for Standardization. 2015. *ISO/IEC/IEEE 15288. Systems and Software Engineering. System Life Cycle Processes: Annex G Application of System Life Cycle Processes to SoS*.
- [7] LIMA, Renata M.; VARGAS, Daniel. FONTOURA, Lisandra M. System of System Requirements: A Systematic Literature Review using Snowballing. In: *SEKE: Software Engineering and Knowledge Engineering*. Pittsburgh, PA: KSI Research Inc. and Knowledge Systems Institute, 2017. v. 29. p. 97 – 100.
- [8] MAIER, M. W. (1996, July). Architecting principles for systems-of-systems. In *INCOSE International Symposium* (Vol. 6, No. 1).

# A Preliminary Investigation of Self-Admitted Refactorings in Open Source Software

Di Zhang, Bing Li, Zengyang Li\*, Peng Liang  
School of Computer Science, Wuhan University  
Wuhan, China

**Abstract**—In software development, developers commit code changes to the version control system. In a commit message, the committer may explicitly claim that the commit is a refactoring with the intention of code quality improvement. We defined such a commit as a self-admitted refactoring (SAR). Currently, there is little knowledge about the SAR phenomenon, and the impact of SARs on software projects is not clear. In this work, we performed a preliminary investigation on SARs with an emphasis on their impact on code quality using the assessment of code smells. We used two non-trivial open source software projects as cases and employed the PMD tool to detect code smells. The study results shows that: (1) SARs tend to improve code quality, though a small proportion of SARs introduced new code smells; and (2) projects that contain SARs have different results on frequently affected code smells.

**Keywords**—self-admitted refactoring; code smell; case study; code quality

## I. INTRODUCTION

Software systems are evolving over time once they are delivered. Software evolution often comprises up to 75% of the costs of software development [1]. However, the decrease in quality and increase of complexity push developers and practitioners to come up with flexible, maintainable, and extensible techniques for improving software quality and reducing change costs. One of these techniques is refactoring that is “the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure [2].” Software maintainability can be well indicated by code smells [3], while refactoring is a recommended daily practice and considered as an effective way to fix code smells [1].

Interestingly, developers often explicitly claim, in the commit messages of version control systems, that their modifications to the software system are refactorings. It means, by definition, that the maintainability of the software system is expected to have been improved. We call such code modifications, claimed as refactorings by developers, self-admitted refactorings (SARs). The concept of SAR is inspired by [4, 5], in which the term of self-admitted technical debt was introduced. Commits of a project can be divided into two categories: SARs and non-SARs (commits without SARs). Currently, there is little empirical evidence on whether SARs do improve the structure quality of the code.

## II. BACKGROUND AND RELATED WORK

### A. Code Smells and Refactoring

Fowler *et al.* proposed to use code smells to indicate the structure quality issues in code, which are possible refactoring opportunities [2]. They defined 22 common code smells, e.g., duplicated code [2]. Later, some new code smells were proposed. For instance, Kerievsky proposed 5 new code smells, e.g., conditional complexity [6]. Refactoring can help improve software design, software understandability, and development efficiency [2]. An approach based on the quantitative analysis of the dependencies between code smells was proposed by Hamza [7], showing that some code smells require larger effort to remedy and should be concerned by developers in refactorings. Besides, the paper also presents the difference between the code smells proposed by Kerievsky and Fowler. However, in the commit records of a project, there are some special refactorings that developers explicitly admit, and such refactorings have seldom been investigated.

### B. Code Smell Detection Tools

It is a complex and tedious work to detect code smells, and tools for automatic detection of code smells are beneficial to developers. Various methods and tools for automatic detection of code smells were proposed [8]. DÉCOR, a method proposed by Moha *et al.*, has a good performance on specification and detection of code and design smells [8]. Tools like Klockwork, PMD, and FindBugs are the typical instances for detecting potential code errors (e.g., naming flaw) and code smells (e.g., large classes). Some tools (e.g., PMD) have been applied into software development practice.

## III. STUDY DESIGN

We conducted a case study on two non-trivial OSS projects written in Java and hosted on GitHub. We follow the guidelines by Runeson and Höst [9] to describe the case study.

### A. Objective and Research Questions

The goal of this study, described using the Goal-Question-Metric (GQM) approach [10], is to analyze the impact of SARs on source code for the purpose of validating its effectiveness in improving maintainability of the code, from the point of view of software developers in the context of OSS projects. We formulated two main research questions (RQs) as follows:

**RQ1:** Do SARs improve the structure quality of source code?

**Rationale:** The aim of refactoring is to improve the internal quality of software structure [2], thus, we want to explore

---

\*Corresponding author. E-mail: [zengyangli@whu.edu.cn](mailto:zengyangli@whu.edu.cn).  
DOI reference number: 10.18293/SEKE2018-081

whether SARs do improve the structural quality of source code. This RQ can be further divided into two sub-RQs.

**RQ1.1:** Do SARs affect (introduce or reduce) code smells? If yes, which code smells are affected most frequently?

**RQ1.2:** Do SARs tend to introduce less code smells than non-SARs?

**RQ2:** What is the distribution of the severity levels of code smells affected in SARs?

**Rationale:** Code smell detection tools (e.g., PMD) provide code smells' severity information, which suggests the priorities of code smells for developers to fix. In SARs, the distribution of code smell severity levels can be used to assess the status of code quality.

### B. Case and Unit of Analysis

According to Runeson and Höst [9], case studies can be characterized based on the way they define their cases and units of analysis. This study investigates the impact of SARs, thus we use an SAR as the unit of analysis.

For case selection, we applied the following criteria:

- The project is with a history of more than 2 years.
- The project has at least 90% of its code written in Java, since PMD is used to detect code smells and it is dedicated to identifying code smells for Java source code.
- The project has more than 20 SARs.
- The project has more than 10 committers.
- The source code of the project should be well commented (high readability and analyzability) to facilitate data analysis.

### C. Data Collection

#### 1) Data to be collected

To answer the RQs formulated in Section III.A, we collected the data items listed in TABLE I, which also lists the target RQ(s) of each data item.

TABLE I. DATA ITEMS TO BE COLLECTED

#	Data item	Description	Target RQs
D1	NCS – Number of Code Smells	The number of code smells of a software system of a revision (commit)	RQ1.1, RQ2
D2	DNCS – Delta of the Number of Code Smells	The change of the number of code smells in a commit comparing with its immediately previous commit; DNCS>0 if NCS increases, DNCS=0 if NCS does not change, and DNCS<0 if NCS decreases	RQ1.1, RQ1.2, RQ2
D3	SLCS – Severity Level of each newly-introduced code smell	The severity level of each newly-introduced code smell in a commit	RQ2

#### 2) SARs Collection

For each project, we performed the following steps:

- (1) **Download code repository.** Download the code repository of the project from GitHub.
- (2) **Export commit records.** Export the commit records of the project using the TortoiseGit client.
- (3) **Identify candidate SARs.** According to the definition of SAR, one way to identify SARs is to search the keywords in

the commit messages of the selected projects. Fowler *et al.* defined 22 types of refactorings [2], which can be used as basis for SAR identification. We extracted key refactoring verbs as detection roots from the refactorings, and the 22 roots of key words are shown in TABLE II. The output of this step is a set of candidate SARs.

- (4) **Check candidate SARs manually.** Check each candidate SAR manually to exclude unexpected cases, e.g., the developer might write 'not to refactor', but no refactorings actually happened. Besides, we used a refactoring detection tool called Ref-Finder [11], to check whether refactorings had actually happened.

- (5) **Record commits of SARs.** Record the commits corresponding to the identified SARs in a spreadsheet. Each commit contains revision number, committer, etc.

TABLE II. DETECTED KEY WORDS ROOTS

Key Words Roots
Refactor/Extract/Inline/Replace/Introduce/Rename/Move/Hide/Encapsulate/Change/Convert/Separate/Decompose/Consolidate/Add/Parameterize/Preserve/Pull up/Pull down/Collapse/Spilt/Substitute

#### 3) Non-SARs Collection

To answer RQ1.2, for each selected project, it requires to collect a set of normal commits that do not contain SARs. We call such commits as non-SARs. We randomly selected a set of non-SARs, and the size of the non-SAR set equals the number of SARs in the project for eliminating the effect of quantity.

#### 4) Code Smells Collection

We detected code smells through the PMD tool, which is a widely-used code smell detection tool adopting a static detection method. There are 33 rule sets and 237 detecting rules (e.g., Cyclomatic Complexity) in PMD. For our case study, the code repository of the selected projects were downloaded and the corresponding code snapshots to SARs of the code repository were exported. Then, we analyzed the source files of the code snapshots corresponding to each SAR and its immediately previous revision to get the differences of code smells between the two code snapshots. Fig. 2 shows the procedure of code smells collection. For each SAR or non-SAR of each selected project, we performed the following steps:

- (1) **Export source code.** Two revisions of a project need to be exported: the revision corresponding to the SAR (or non-SAR) (V1) and its immediately previous revision (V2).
- (2) **Detect code smells.** Use the PMD plugin for eclipse to detect code smells of revisions V1 and V2. The PMD plugin will generate reports on the detected code smells.
- (3) **Export code smell reports.** Export the code smell reports generated in the previous step.
- (4) **Identify differences in code smell reports.** Compare the code smell reports of V1 and V2 to identify the differences in code smells between the reports. We developed a dedicated tool to accomplish this task.

#### 5) Data Analysis

To answer the RQs formulated in Section III.A, we need to analyze the collected data on SARs and code smells. For RQ1.1, RQ1.2, and RQ2, only descriptive statistics were used.

## IV. STUDY RESULTS

### A. Selected Cases

We selected two OSS projects, i.e., Fastjson<sup>1</sup> and Junit4<sup>2</sup>, as the cases. The two OSS projects are widely used in many software systems. TABLE III shows the demographic information of the three selected projects. Fastjson has 111,247 lines of code, 121 SARs, and 1,662 commits; while Junit4 has 26,579 lines of code, 28 SARs, and 2,090 commits .

TABLE III. DEMOGRAPHIC INFORMATION OF THE SELECTED PROJECTS

Project	Fastjson	Junit4
Sponsor Company	Alibaba	Apache
Access Date	5/21/2016	5/19/2016
Contributors	116	120
Line of Code	111,247	26,579
Percentage of code written in Java	99.90%	99.00%
Number of commits containing SARs	121	28
Number of commits	1,662	2,090
Percentage of SARs against total commits	7.28%	1.34%

### B. Results

#### 1) Impact on code quality (RQ1)

**RQ1.1:** TABLE IV shows the number of SARs in the two cases regarding the changed number of code smells. 70.25% (85/121) and 67.86% (19/28) of SARs did not increase code smells in Fastjson and Junit4, respectively. This suggests that more than half of the SAR revisions of the two projects were intended to decrease code smells compared with their previous versions. That is a positive signal of improving code quality.

TABLE IV. DNCS IN SELF-ADMITTED REFACTORINGS

Project	Fastjson		Junit4	
	#( SAR)	%	#( SAR)	%
DNCS > 0	36	29.75	9	32.14
DNCS = 0	39	32.23	9	32.14
DNCS < 0	46	38.02	10	35.72
DNCS ≤ 0	85	70.25	19	67.86
Total	121	100.00	28	100.00

We listed in TABLE V the top 5 code smells that are affected (introduced or decreased) most. Among all types of code smells, *DataflowAnomalyAnalysis* is the one that is affected most frequently in the two cases. Specifically, *DataflowAnomalyAnalysis* decreased in Fastjson and increased in Junit4 most frequently. This type of code smells happened most frequently as well.

TABLE V. TOP 5 MOST AFFECTED CODE SMELLS

Code smell	Fastjson		Code smell	Junit4	
	DNCS	NCS		DNCS	NCS
DataflowAnomalyAnalysis	-131	1153	DataflowAnomalyAnalysis	18	34
LooseCoupling	-24	102	SignatureDeclareThrowException	8	28
UnusedImports	-20	84	TooManyMethods	6	8
CyclomaticComplexity	-19	127	ConfusingTernary	3	5
SignatureDeclareThrowException	-10	702	PreserveStackTrace	1	11

**RQ1.2:** As shown in TABLE VI, for Fastjson, there were 121 SARs, and thus 121 non-SARs were randomly selected for

comparison. The number of code smells increased (DNCS>0), kept unchanged (DNCS=0), and decreased (DNCS<0) in 76, 29, and 16 non-SARs, respectively. The number of code smells increased, kept unchanged, and decreased, in 36, 39, and 46 SARs, respectively. 37.19% (45/121) of non-SARs did not increase code smells while 70.25% (85/121) of SARs did not increase code smells in Fastjson. It means that, compared with non-SARs, SARs tend not to increase code smells in Fastjson.

However, different from Fastjson, in Junit4, more non-SARs did not increase the number of code smells than SARs. As shown in TABLE VI, 75.00% of non-SARs and 67.86% of SARs did not increase code smells in Junit4.

TABLE VI. DNCS IN SARs AND NON-SARs

Project	Fastjson		Junit4	
	#(Non-SAR)	#(SAR)	#(Non-SAR)	#(SAR)
DNCS > 0	76	36	7	9
DNCS = 0	29	39	11	9
DNCS < 0	16	46	10	10
DNCS ≤ 0	45	85	21	19
Total	121	121	28	28
Proportion (DNCS≤0)	37.19%	70.25%	75.00%	67.86%

#### 2) Severity level distribution of affected code smells (RQ2)

PMD can detect 237 types of code smells. The priority of each code smell is defined in PMD, and it uses numbers 1 – 5 to denote the priority levels: Error High, Error, Warning High, Warning, and Information. A smaller priority number of a code smell means it is more important and more urgent to be fixed.

TABLE VII. PRIORITY DISTRIBUTION OF CODE SMELLS

PMD info	Fastjson					Junit4			
	#(PMD code smell type)	#(Detected code smell type)	Type Percentage	#(Detected code smell)	DNCS	#(Detected code smell type)	Type Percentage	#(Detected code smell)	DNCS
Information (5)	1	1	100.00	1153	131	1	100.00	34	18
Warning (4)	14	1	7.14	84	-20	1	7.14	4	0
Warning High (3)	188	77	40.56	6169	-327	55	29.26	559	137
Error (2)	19	2	10.53	96	8	1	5.26	4	-4
Error High (1)	14	6	42.86	813	-17	3	21.43	39	10

TABLE VII shows the distribution of default severity levels of code smells identified by PMD. #(PMD code smell type) represents the number of priorities of code smells that are defined in PMD. DNCS is defined as the delta of the number of code smells in SARs. #(Detected code smell type) is the number of code smell types that were actually detected in a project. Type Percentage denotes the proportion of detected code smell types against all code smell types that can be detected by PMD. #(Detected code smell) represents the number of priorities of code smells that were actually detected by PMD in the SARs. As shown in TABLE VII, in all the SARs of Fastjson, 131 code smells with priority “Information” and 8 with priority “Error” were introduced; 20 with priority “Warning”, 327 with priority “Warning High”, and 17 with priority “Error High” were removed. As for Junit4, the results are similar in the distribution of code smell severity.

<sup>1</sup> <https://github.com/alibaba/fastjson>

<sup>2</sup> <https://github.com/junit-team/junit4>

## V. DISCUSSION

In this section, we discuss the study results and their implications as well as threats to validity of the results.

### A. Understanding on Study Results

**RQ1.1:** As shown in TABLE IV, more than one half of SARs tend not to increase code smells in the two projects, which indicates that the code quality is likely to be improved through SARs. It is not surprising since developers take the initiative to improve the maintainability of code in SARs. However, the results imply that not all SARs can lead to quality improvement of the source code. Only a small percentage of SARs introduced new code smells. *DataflowAnomalyAnalysis* is the code smell type that happened and affected most frequently. The top 2 largest numbers of code smells may be caused by the small granularity of code smell detection rules in PMD.

**RQ1.2:** From the comparison between the two datasets of SARs and non-SARs, 37.19% of non-SARs and 70.25% of SARs did not increase code smells of project Fastjson, which means less code smells introduced in SARs than in non-SARs. Nevertheless, more non-SARs than SARs did not increase code smells in Junit4, which indicates better performance of non-SARs than SARs of Junit4 in code smells reduction. We reviewed and used Ref-Finder to check the selected non-SARs, and found that refactorings had happened in some non-SARs. Additionally, the scale of Junit4 is relatively small, thus less SAR information of Junit4 than Fastjson may lead to the fluctuation of the results. Not all non-SARs do not contain refactorings and not all SARs contain refactorings. SAR is a signal to find refactorings happened, but it does not mean that refactorings definitely happened in all SARs. This indicates that some developers may not strictly distinguish refactorings from normal code changes, which may result in misunderstandings on developers' modifications on code.

**RQ2:** As shown in TABLE VII, the priority types of actually detected code smells cover all priority types defined in PMD. Take Fastjson for example, 42.86% of the code smells are with priority "Error High". Code smells with priority "Warning High" take the most proportion against all detected code smells, which implies that the code quality of the studied project may be moderate (similar performance in Junit4). This may partially result from the fact that the numbers of predefined code smell types are not balanced and most of code smell types of PMD are with priority "Warning High" (see TABLE VII). The number of code smells with priority "Error" increased, which is a signal of code quality sliding and should be paid special attention to.

### B. Implications

More code smells means quality decline of a project, and in our case study, SARs are generally a positive sign of code quality improvement. However, some SARs may hurt the quality of source code, i.e., a SAR may not be a real refactoring. SARs indicate developers' awareness of code quality improvement, since the fact that developers claim refactorings explicitly, to some extent, represents their willingness to improve the structure quality of the code.

The distribution of severity levels of affected code smells in SARs reflects the maintainability status of a project to certain extent. When more code smells of high severity levels were removed in SARs, the project's maintainability would be more improved.

## VI. CONCLUSIONS

Self-admitted refactorings (SARs) were seldom studied in previous research. In this work, we explored the SAR phenomenon from multiple perspectives. Based on the results on three studied OSS projects, we draw the following conclusions: (1) SARs tended to improve code quality, though a small proportion of SARs introduced new code smells. (2) Different projects do not have the same results on frequent-affected code smells. (3) More than half of SARs did not introduce code smells; however, non-SARs did not suggest less decrease of code smells than SARs. (4) In SARs, most code smells are with a moderate priority of "Warning High" to fix.

## ACKNOWLEDGMENT

This work is supported by the National Key Research and Development Program of China (Nos. 2017YFB1400602 and 2016YFB0800401), the National Natural Science Foundation of China (Nos. 61572371, 61702377, 61472286, and 61773175), the Wuhan Yellow Crane Special Talents Program, the CPSF (No. 2015M582272), the NSF of Hubei Province (No. 2016CFB158), and the Fundamental Research Funds for the Central Universities (No. 2042016kf0033).

## REFERENCES

- [1] M. Abebe and C.-J. Yoo, "Trends, opportunities and challenges of software refactoring: A systematic literature review," *International Journal of Software Engineering and Its Applications*, vol. 8, no. 6, pp. 299-318, 2014.
- [2] M. Fowler, K. Beck, J. Brant, W. Opdyke, and D. Roberts, *Refactoring: improving the design of existing code*. Addison-Wesley Longman Publishing Co., Inc., 1999, pp. 256-256.
- [3] A. Yamashita and S. Counsell, "Code smells as system-level indicators of maintainability: An empirical study," *Journal of Systems and Software*, vol. 86, no. 10, pp. 2639-2653, 2013.
- [4] A. Potdar and E. Shihab, "An Exploratory Study on Self-Admitted Technical Debt," presented at the ICSME '14 Proceedings of the 2014 IEEE International Conference on Software Maintenance and Evolution, Washington, DC, USA, 2014.
- [5] E. Maldonado, E. Shihab, and N. Tsantalis, "Using Natural Language Processing to Automatically Detect Self-Admitted Technical Debt," *IEEE Transactions on Software Engineering*, 2017.
- [6] J. Kerievsky, *Refactoring to Patterns*. Addison-Wesley Professional, 2004, p. 400.
- [7] H. Hamza, S. Counsell, T. Hall, and G. Loizou, "Code smell eradication and associated refactoring," in *European Computing Conference (ECC'08)*, Malta, 2008, pp. 102-107: Springer.
- [8] N. Moha, Y. G. Gueheneuc, L. Duchien, and A. F. L. Meur, "DECOR: A Method for the Specification and Detection of Code and Design Smells," *IEEE Transactions on Software Engineering*, vol. 36, no. 1, pp. 20-36, 2010.
- [9] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, pp. 292-295, 2008.
- [10] V. R. Basili, *Software modeling and measurement: the goal/question/metric paradigm*. University of Maryland at College Park, 1992.
- [11] K. Prete, N. Rachatasumrit, N. Sudan, and M. Kim, "Template-based reconstruction of complex refactorings," in *2010 IEEE International Conference on Software Maintenance*, 2010, pp. 1-10.

# Formalization and Verification of the OpenFlow Bundle Mechanism Using CSP

Huiwen Wang   Huibiao Zhu\*   Yuan Fei   Lili Xiao  
Shanghai Key Laboratory of Trustworthy Computing,  
School of Computer Science and Software Engineering,  
East China Normal University, Shanghai, China

**Abstract**—Software Defined Network (SDN) is an emerging architecture of computer networking. The most important feature of SDN is that it separates the control plane from the data plane. OpenFlow is considered as the first and currently most popular standard southbound interface of SDN. It is a communication protocol which enables the SDN controller to directly interact with the forwarding plane. The widespread use makes the reliability of OpenFlow important. The OpenFlow bundle mechanism is a new mechanism proposed by OpenFlow protocol to guarantee the completeness and consistency of the messages transmitted between SDN switches and controllers during the communication process. Due to the requirement of reliability and security of OpenFlow, we think that it is of great significance to formally analyze and verify the safety-relevant properties of the mechanism. In this paper, we apply Communication Sequential Processes (CSP) and the model checker Process Analysis ToolKit (PAT) to model and verify the OpenFlow bundle mechanism. Our formalization and verification show that the mechanism can satisfy four properties: deadlock freeness, parallelism, atomicity and order property, from which we can conclude that the mechanism offers a better way to guarantee the completeness and consistency.

**Keywords**—OpenFlow; Bundle Mechanism; Formalization; Verification

## I. INTRODUCTION

Software Defined Network (SDN) is an emerging architecture of computer networking which is totally different from the current network infrastructure. SDN separates the control plane and the data plane to realize the flexibility and programmability of the network. OpenFlow protocol was originally proposed for campus network [1]. It is the first and currently the most popular southbound interface of SDN. It first defines the communication protocol which enables the SDN controller to directly interact with the forwarding plane consisting of network devices such as switches [2]. Therefore, formalizing and verifying the mechanism of OpenFlow protocol is necessary.

OpenFlow protocol supports three kinds of messages: controller-to-switch messages, asynchronous messages and symmetric messages [3]. The bundle message is a kind of controller-to-switch messages put forward to guarantee the consistency and completeness of communication. In recent years, several methods have been given to solve problems like packet losses and duplications [4]. The bundle mechanism is one of the solutions proposed by OpenFlow. A bundle is

a sequence of OpenFlow messages sent from the controller to the switch and all the modifications in these messages are applied as a single operation. The bundle mechanism groups related changes and applies them together so that the completeness can be guaranteed [5]. And it also supports the order property. It means that the modifications inside a bundle should be applied in sequence. Our work in this paper is to model and verify the mechanism.

The remainder of this paper is organized as follows. In section 2, we present an overview of the OpenFlow bundle mechanism and a brief introduction to CSP. Section 3 shows the formalization of the OpenFlow bundle mechanism. The implementation and verification of the model are presented in section 4. Finally, we conclude the paper and discuss some possible future work in section 5.

## II. BACKGROUND

In this section, we give an overview of the OpenFlow bundle mechanism and a brief introduction to the CSP is presented as well.

### A. The OpenFlow Bundle Mechanism

OpenFlow specification has defined a configuration that the switch uses bundle messages as a transmission manner. A sequence of OpenFlow messages sent from the controller to the switch is stored in a bundle. Each bundle is specified with a unique bundle ID in a specific controller connection.

The bundle messages are transmitted between the controller and the switch. And there are two kinds of bundle messages: the bundle control message and the bundle add message. The bundle control message is used to perform operations on a bundle, e.g. open operation, close operation and commit operation. The bundle add message is formatted as a regular OpenFlow message which includes modifications in the message to be applied later. The bundle message has two properties: atomicity and order property, which are shown by the parameter flag. Atomicity means that the modifications should be done all or nothing. And order property means that the modifications should be executed in the order they are added in the bundle.

When a bundle is open, messages can be stored in the bundle without being applied. When the switch commits a bundle, the bundle should be closed and the switch should pre-validate the modifications inside the bundle and commit the

\*Corresponding Author. E-mail address: hbzhu@sei.ecnu.edu.cn (H. Zhu).



bundle afterwards. The whole process includes the following five operations: opening a bundle, adding a message to a bundle, closing a bundle, committing a bundle and discarding a bundle. The detailed descriptions are presented as below:

- 1) Opening a bundle: The switch opens a new bundle according to the connection ID and bundle ID pairs. The switch should guarantee the validity of the connection ID, by checking whether the bundle already exists and verifying the correctness of other properties.
- 2) Adding a message to a bundle: The switch adds messages into a bundle and then the switch fetches the bundle by the ID pairs. Once fetched successfully, the switch validates properties of the added OpenFlow messages. For example, the switch should guarantee the consistency of the OpenFlow messages.
- 3) Closing a bundle: The switch closes the fetched bundle. A bundle should be closed before it is committed while a nonexistent bundle can not be closed.
- 4) Committing a bundle: The switch commits the fetched bundle. If a bundle is not closed, the switch closes the bundle and then commits it. The process of committing should satisfies the features of atomicity and order property. The modifications should be applied in all or nothing and in order. Whether the committing is successful or not, the switch discards the bundle afterwards.
- 5) Discarding a bundle: A bundle can be discarded during the whole process from the creation of the bundle. But if the bundle does not exist, the operation fails.

In addition, the switch must support the exchange of echo messages during the transmission of bundle messages to guarantee the liveness of the connection.

## B. CSP

A brief overview of CSP is given in this subsection. Process algebra is a representative of the formal methods, which can illustrate concurrent systems easily using intuitive expressions and strict mathematical theories. CSP is proposed by C.A.R Hoare in 1978, which enriches process algebra [6]. Due to the powerful expression, CSP has been successfully applied in many fields [7], [8]. CSP processes are composed of primitive processes and actions. The syntax of a subset of the CSP is given as below.  $P$  and  $Q$  are two processes.  $a$  and  $b$  are two actions. And  $c$  is a channel.

$$P, Q ::= \text{Skip} | \text{Stop} | a \rightarrow P | c?x \rightarrow P | c!v \rightarrow Q | P \triangleleft b \triangleright Q \\ | P; Q | P \square Q | P || Q | P ||| Q | P [|X] Q$$

- $\text{Skip}$  denotes that a process does nothing but terminates successfully.
- $\text{Stop}$  denotes that the process is in the state of deadlock and does nothing.
- $a \rightarrow P$  represents that the process first engages in action  $a$ , then the subsequent behaviour is like  $P$ .
- $c?x \rightarrow P$  receives a message through the channel  $c$  and assigns it to a variable  $x$ , then behaves like  $P$ .

- $c!v \rightarrow Q$  sends a message  $v$  using the channel  $c$ , then the behaviour is like  $P$ .
- $P \triangleleft b \triangleright Q$  denotes if the condition  $b$  is true, the behaviour is like  $P$ , otherwise, like  $Q$ .
- $P; Q$  performs  $P$  and  $Q$  sequentially.
- $P \square Q$  behaves like either  $P$  or  $Q$  and the choice is made by the environment.
- $P || Q$  denotes that  $P$  runs in parallel with  $Q$ .
- $P ||| Q$  indicates that  $P$  interleaves  $Q$  which means  $P$  and  $Q$  run concurrently and randomly.
- $P [|X] Q$  indicates that  $P$  and  $Q$  perform the concurrent events on the set  $X$  of channels.

## III. FORMALIZATION OF THE OPENFLOW BUNDLE MECHANISM

In this section, we model the OpenFlow bundle mechanism using CSP. The formalization is based on the description of the mechanism presented in section 2.

### A. Sets, Messages and Channels

For convenience, we define seven sets in our model. The set  $ConnectionID$  represents identities of connections and the set  $BundleID$  represents identities of bundles. The set  $Type$  denotes types of bundle messages, e.g.  $echo$ ,  $open$ ,  $close$ ,  $commit$  and  $discard$ . The set  $Flag$  shows the two properties of a bundle, e.g. atomicity and order property. The set  $Operator$  represents operations of switches. The set  $State$  indicates the states of the bundle including  $open$  and  $close$ . And the set  $Content$  represents other message contents. Based on the definitions above, we model the messages used in the mechanism as below:

$$MSG =_{df} MSG_{echo} \cup MSG_{con\_swt} \cup MSG_{swt\_bun},$$

$$MSG_{con\_swt} =_{df} MSG_{con} \cup MSG_{add} \cup MSG_{err},$$

$$MSG_{swt\_bun} =_{df} MSG_{req} \cup MSG_{rep} \cup MSG_{err},$$

$$MSG_{echo} =_{df} \{connection\_id.content | content \in Content, \\ connection\_id \in ConnectionID\}.$$

$$MSG_{con} =_{df} \{connection\_id.bundle\_id.type.flag | \\ connection\_id \in ConnectionID, \\ bundle\_id \in BundleID, \\ type \in Type, flag \in Flag\},$$

$$MSG_{add} =_{df} \{connection\_id.bundle\_id.flag.content | \\ connection\_id \in ConnectionID, \\ bundle\_id \in BundleID, \\ flag \in Flag, content \in Content\},$$

$$MSG_{err} =_{df} \{connection\_id.bundle\_id.type | \\ connection\_id \in ConnectionID, \\ bundle\_id \in BundleID, type \in Type\},$$

$$MSG_{req} =_{df} \{connection\_id.bundle\_id.operator.content | \\ connection\_id \in ConnectionID, \\ bundle\_id \in BundleID, \\ operator \in Operator, content \in Content\},$$

$$MSG_{rep} =_{df} \{ \text{connection\_id.bundle\_id.operator.state} \mid \\ \text{connection\_id} \in \text{ConnectionID}, \\ \text{bundle\_id} \in \text{BundleID}, \\ \text{operator} \in \text{Operator}, \text{state} \in \text{State} \}.$$

We define three kinds of channels to model the communication among components:

- *ComCS* is a channel for controllers and switches to transmit bundle messages.
- *ComSB* is a conceptual channel used to represent the communication between the switch and its bundle process field.
- *ComEcho* is an optional channel for controllers and switches to transmit echo messages.

### B. Components

In this subsection, we use CSP to model OpenFlow controllers and switches which adopt bundle messages in the transmission manner. There are two levels in the model system. The first level happens between the controller and the switch. In the second level, the switch will search the bundle field it stores for the specific connection. To simplify, we abstract a component *Bundle* communicating with the switch to simulate the process that the switch searches the bundle field. Our whole model is composed of three processes: *Controller*, *Switch* and *Bundle*.

**Controller.** The controller sends a sequence of OpenFlow modification requests in a bundle to switches. After that the controller will receive the response messages from switches. Each connection between a controller and a switch has an identity. At the same time, the controller supports exchanging echo messages to check the liveness of the connection. We model the behaviors as followed:

$$\text{Controller}(\text{connection\_id}, \text{bundle\_id}, \text{type}) =_{df} \\ \left( \begin{array}{l} (\text{ComEcho!Msg}_{echo} \rightarrow \text{ComEcho?Msg}_{echo}) \\ \langle \text{type} = \text{echo} \rangle \\ (\text{ComCS!Msg}_{con\_swt} \rightarrow \text{ComCS?Msg}_{con\_swt}) \end{array} \right) \\ \rightarrow \text{Controller}(\text{connection\_id}, \text{bundle\_id}, \text{type});$$

**Switch.** After receiving the message sent from the controller, the switch will search its bundle field to perform validations corresponding to the message received. We abstract the process as the communication between the switch and its bundle field. The switch sends a request message to its bundle field and receives a response after the bundle field checks the request message. Then the switch sends a corresponding response message to the controller. We model the behaviors as below:

$$\text{Switch}(\text{connection\_id}, \text{bundle\_id}, \text{type}) =_{df} \\ \text{ComEcho?Msg}_{con\_swt} \rightarrow \\ \left( \begin{array}{l} (\text{ComEcho!Msg}_{echo}) \\ \langle \text{type} = \text{echo} \rangle \\ (\text{ComSB!Msg}_{swt\_bun} \rightarrow \text{ComSB?Msg}_{swt\_bun}) \\ \rightarrow \text{ComCS?Msg}_{con\_swt} \end{array} \right) \\ \rightarrow \text{Switch}(\text{connection\_id}, \text{bundle\_id}, \text{type});$$

**Bundle.** It is a process abstracted out from the performance of switches after receiving bundle messages. The bundle fields

store bundles created by a switch. And the pre-validations of messages happen in this process. The detailed description can be found in section 2. The behaviors are modelled as below:

$$\text{Bundle}(\text{connection\_id}, \text{bundle\_id}, \text{operator}) =_{df} \\ \text{Com?MSG}_{swt\_bun} \rightarrow \\ \text{Assort}(\text{connection\_id}, \text{bundle\_id}, \text{operator}); \\ \left( \begin{array}{l} \text{Com!MSG}_{swt\_bun} \rightarrow \\ \text{Bundle}(\text{connection\_id}, \text{bundle\_id}, \text{operator}) \end{array} \right);$$

$$\text{Assort}(\text{connection\_id}, \text{bundle\_id}, \text{operator}) =_{df} \\ \text{if}(\text{operator} == \text{opn}) \\ \text{then}(\text{openBundle}(\text{connection\_id}, \text{bundle\_id}, \text{operator})) \\ \text{elseif}(\text{operator} == \text{add}) \\ \text{then}(\text{addBundle}(\text{connection\_id}, \text{bundle\_id}, \text{operator})) \\ \text{elseif}(\text{operator} == \text{cls}) \\ \text{then}(\text{clsBundle}(\text{connection\_id}, \text{bundle\_id}, \text{operator})) \\ \text{elseif}(\text{operator} == \text{cmt}) \\ \text{then}(\text{cmtBundle}(\text{connection\_id}, \text{bundle\_id}, \text{operator})) \\ \text{elseif}(\text{operator} == \text{dis}) \\ \text{then}(\text{delete}());$$

The five operations are described respectively as below:

- 1) *openBundle(connection\_id, bundle\_id, operator)* creates a new bundle in the switch's bundle field. The switch must perform the following validations. The connection should be reliable and the *bundle\_id* should refer to a nonexistent bundle. And if the two conditions are satisfied, the value of *bundle\_id* should be set to *false* and the state of the bundle is set to *open* and then return *true*.

$$\text{openBundle}(\text{connection\_id}, \text{bundle\_id}) =_{df} \\ \left( \begin{array}{l} \text{setContent}(\text{true}, \text{setBundleID}(\text{false}), \\ \text{setState}(\text{open})) \\ \triangleleft \left( \begin{array}{l} (\text{connection\_id} \in \text{ConnectionID}) \\ \wedge (\text{bundle\_id} \notin \text{BundleID}) \end{array} \right) \triangleright \\ \text{setContent}(\text{false}) \end{array} \right) \\ ; \text{openBundle}(\text{connection\_id}, \text{bundle\_id});$$

- 2) *addBundle(connection\_id, bundle\_id, operator)* adds messages to a bundle. When the switch adds messages, it should first fetch the bundle using the *bundle\_id* and *connection\_id* pair. We define the parameter *vars* as the number of messages to be added. If the fetched bundle is open and other properties are legal, the switch can add the message successfully and return *true*.

$$\text{addBundle}(\text{connection\_id}, \text{bundle\_id}) =_{df} \parallel_{i \in \text{vars}} \\ \left( \begin{array}{l} \text{setContent}(\text{true}, \text{setMsgNum}(\text{num} + 1)) \\ \triangleleft \left( \begin{array}{l} (\text{connection\_id} \in \text{ConnectionID}) \\ \wedge (\text{bundle\_id} \in \text{BundleID}) \\ \wedge (\text{state}_{\text{bundle\_id}} = \text{open}) \end{array} \right) \triangleright \\ \text{setContent}(\text{false}) \end{array} \right) \\ ; \text{addBundle}(\text{connection\_id}, \text{bundle\_id});$$

- 3) *clsBundle(connection\_id, bundle\_id, operator)* closes a bundle after finishing adding messages. And the

bundle to be closed should be fetched successfully with  $connection\_id$  and  $bundle\_id$  pairs and its state should be open. Once all these validations are successful, the bundle state is set to  $close$  and return  $true$ .

$$clsBundle(connection\_id, bundle\_id) =_{df} \left( \begin{array}{l} setContent(true, setState(close)) \\ \triangleleft \left( \begin{array}{l} (connection\_id \in ConnectionID) \\ \wedge (bundle\_id \notin BundleID) \\ \wedge (state_{bundle\_id} = open) \end{array} \right) \triangleright \\ setContent(false) \end{array} \right)$$

;  $clsBundle(connection\_id, bundle\_id)$ ;

- 4)  $cmtBundle(connection\_id, bundle\_id, operator)$  commits the bundle. After fetching the bundle with legal  $connection\_id$  and  $bundle\_id$  pairs, the switch performs the following actions. Firstly, if the bundle state is  $open$ , the switch closes the bundle and then continues to commit the bundle. Secondly, it verifies the flag associated with the bundle and performs the corresponding actions.

$$cmtBundle(connection\_id, bundle\_id) =_{df} \begin{array}{l} appMsg(connection\_id, bundle\_id); \\ \left( \begin{array}{l} setContent(true, setCmtNum(MsgNum)); \\ disBundle(connection\_id, bundle\_id) \end{array} \right) \\ \triangleleft \left( \begin{array}{l} (connection\_id \in ConnectionID) \\ \wedge (bundle\_id \in BundleID) \\ \wedge (state_{bundle\_id} = close) \end{array} \right) \triangleright \\ \left( \begin{array}{l} clsBundle(connection\_id, bundle\_id); \\ cmtBundle(connection\_id, bundle\_id) \end{array} \right) \\ \triangleleft \left( \begin{array}{l} (connection\_id \in ConnectionID) \\ \wedge (bundle\_id \in BundleID) \\ \wedge (state_{bundle\_id} = open) \end{array} \right) \triangleright \\ \left( \begin{array}{l} setContent(false, setCmtNum(0)); \\ disBundle(connection\_id, bundle\_id) \end{array} \right) \\ ; cmtBundle(connection\_id, bundle\_id); \end{array}$$

- 5)  $disBundle(connection\_id, bundle\_id, operator)$  discards the bundle fetched with the  $connection\_id$  and  $bundle\_id$  pair. Then it sets the  $bundle\_id$  to be 1 and deletes all the messages in the bundle regardless of the bundle state.

$$disBundle(connection\_id, bundle\_id) =_{df} \left( \begin{array}{l} setContent(true, setBundleID(true), \\ setMsgNum(null)) \\ \triangleleft \left( \begin{array}{l} (connection\_id \in ConnectionID) \\ \wedge (bundle\_id \in BundleID) \end{array} \right) \triangleright \\ setContent(false) \end{array} \right)$$

;  $disBundle(connection\_id, bundle\_id)$ ;

And the mechanism should support the exchange of echo messages and creations of multiple bundles. We add the two processes into our model as below:

**Echo.** Exchanging echo request and echo reply messages is supported during the whole process of the bundle. Echo messages are not included in a bundle and only transmitted between the controller and the switch with the channel

$ComEcho$ . And multiple connections may be included. Multiple bundles can be created in parallel. We use  $N$  to represent the number of interleaving connections.

$$Echo() =_{df} ||_{i \in N} \left( \begin{array}{l} Controller(i, bundle\_id, echo) \\ |[ComEcho]| \\ Switch(i, bundle\_id, echo) \end{array} \right)$$

**BundleSys.** The process of the creation of the bundle can be modeled as the parallel composition of the the controller, switch and bundle process and multiple connections may be included.  $N$  and  $M$  represents the number of connections and the number of bundles respectively.

$$BundleSys() =_{df} ||_{i \in N, j \in M, type \in T_{type}} \left( \begin{array}{l} Controller(i, j, type) |[ComCS]| \\ Switch(i, j, type) |[ComSB]| \\ Bundle() \end{array} \right)$$

### C. System

After modeling the processes  $Echo()$  and  $Bundle()$ , the OpenFlow Bundle architecture can be modeled as the concurrence of the two processes.

$$System() =_{df} Echo() || BundleSys()$$

Fig.1 shows the whole system we have modeled. We mark the  $Echo$  part of the system with color blue and the  $BundleSys$  part with color black.

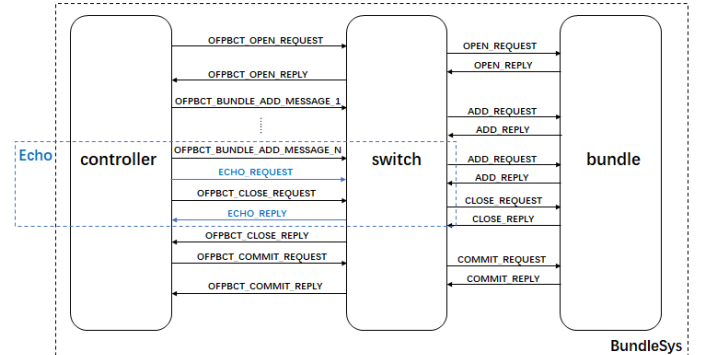


Fig. 1. Modeled System of OpenFlow Bundle Mechanism

## IV. IMPLEMENTATION AND VERIFICATION

In this section, we encode the CSP description above in PAT code and perform a series of validations. The verification is about four key properties of the OpenFlow Bundle Messages transmission. They are *Deadlock Freeness*, *Parallelism*, *Atomicity* and *Order Property*. In the following part, we give the detailed description.

### A. Implementation in PAT

PAT is a model checker tool for automatic system analysis based on CSP [9]. Many systems can be verified in PAT such as concurrent real-time system, probabilistic systems and other

domains [10], [11]. In this subsection, we encode our model in PAT and verify it.

We define some significant channels and variables.  $N$  represents the number of connections between multiple controllers and a switch.  $M$  denotes the number of bundles stored in bundle field.  $P$  indicates the number of messages added into a bundle. We use channel  $ComCS[N]$ , channel  $ComSB[N]$  and channel  $ComEcho[N]$  to describe different transmissions.

In the trial, we set  $N$  to be 3,  $M$  to be 2 and  $M$  to be 4 randomly respectively. And the buffer size is set to be 0 to ensure the communication among components is synchronous.

```
#define N 3;
#define M 2;
#define P 4;
channel ComCS[N] 0;
channel ComSB[N] 0;
channel ComEcho[N] 0;
```

We define some arrays to determine whether the process is successfully executed. We set the variables to be 1 to represent true and 0 to represent false. We also define some other arrays to record the state of the bundle, the messages added inside the bundle and the message committed as the executed results. Some of the declarations are given as below:

```
var EchoReturn = true;
var msgNum[N*M];
var cmtNum[N*M];
```

We define five functions with the same passing parameters and call them with different values. Then we implement the processes from the creation to the discard of a bundle. Because there may be multiple controllers connected to a switch and more than one bundle, we use *subBundleSys* to represent a single connection and use  $i, j$  as *connection\_id* and *bundle\_id* to distinguish it.

```
subBundleSys(i, j) =
(OpenBundle(i, j); AddBundle(i, j);
 CloseBundle(i, j); CommitBundle(i, j)
 ||| Echo(i);
```

The OpenFlow Bundle message transmission system can be implemented by taking advantages of non-determinism and interleaving, with  $i$  identifying the number of each connection. We give the definition of the complete system as follows:

```
BundleSystem() =
Init(); (||| i: {0..N-1} @ subBundleSys(i, 0));
```

### B. Verification

In this subsection, we use model checker PAT to simulate the execution of transmission of OpenFlow bundle messages and verify the properties. PAT searches the state space of the system until it locates a counterexample or exhausts the state space.

#### Property 1: Deadlock Freeness

If a component waits to receive information and no other components feel like sending messages to it, the system gets

stuck in a deadlock state. A security protocol should be free of deadlock. PAT tool provides a primitive assertion to verify the property as below:

```
#assert BundleSystem() deadlockfree;
```

We can conclude from Fig 2 that the system is free of deadlock.

#### Property 2: Parallelism

The system must support exchanging echo messages. Parallelism means echo messages can be transmitted without waiting for the end of the bundle and support multiple controller channels as well. It allows multiple bundles to be created in parallel. Randomly, we define three bundles created by different controllers and we add four messages into each bundle. Our goal is that in the end, the number of messages that each bundle commits successfully is four.

```
#define Parallelism(cmtNum[0] == 4 &&
cmtNum[1] == 4 &&
cmtNum[2] == 4);
#assert BundleSystem() reachesParallelism;
```

Fig 2 shows that the verification result of parallelism is valid which means that the mechanism can support exchange echo messages with multiple bundle messages transmitted in parallel.

#### Property 3: Atomicity

Atomicity means that the switch should commit the messages inside the bundle in an all-or-nothing way. If one or more messages stored in the bundle can not be committed properly, then no messages will be committed. All the messages should be pre-validated. We set the values to true or false to determine whether it can be committed successfully. Then we check whether the number of committed messages of each bundle is zero or all.

```
#define Atm0(cmtNum[0] == 0 xor cmtNum[0] == 4);
#define Atm1(cmtNum[1] == 0 xor cmtNum[1] == 4);
#define Atm2(cmtNum[2] == 0 xor cmtNum[2] == 4);
#define Atomicity(Ato0 && Ato1 && Ato2);
```

If the verification is valid, the OpenFlow bundle mechanism can guarantee atomicity.

```
#assert BundleSystem() |= Atomicity;
```

As shown in Fig 2, the bundle commits either all the messages inside it or none of the messages which satisfies the atomicity property.

#### Property 4: Order Property

If the switch supports the order property, it should strictly commit the messages according to the order they added. We use array *msgApplied* to represent whether the message is committed. There are three valid conditions listed as follows. None of the messages is committed. The former messages are

committed and the latter ones are not. And all of the messages are committed.

```
#define NoApl(msgApplied[0] == 0&&
    msgApplied[1] == 0);
#define ForApl(msgApplied[0] == 1&&
    msgApplied[1] == 0);
#define AllApl(msgApplied[0] == 1&&
    msgApplied[1] == 1);
#define Order(NoApl xor ForApl xor AllApl);
```

The system should satisfy any one of the three conditions.

```
#assert BundleSystem() |= Order;
```

The messages are committed in sequence which satisfies the order property as shown in Fig 2.

Fig.2 shows the verification results of all the properties and they are all valid. We can conclude that the OpenFlow bundle mechanism can guarantee the four properties to keep consistency and completeness of the communication.

### V. CONCLUSION

The OpenFlow bundle mechanism is proposed to provide a method for guaranteeing consistency and completeness of network updates. In this paper, we construct a formal model for the OpenFlow bundle mechanism based on CSP. In addition, we encode the CSP description in the model checker tool PAT and perform the validation of four properties including deadlock freeness, parallelism, atomicity and order property. Corresponding to the verification results, we conclude that the mechanism can guarantee these transmission properties.

In the future, we will continue our work on the formalization and verification of OpenFlow. As the communication process we model in this paper is synchronous, We plan to explore a more general method which can fully guarantee consistency and completeness of the communication process in asynchronous situations based on time.

**Acknowledgement.** This work was partly supported by Shanghai Collaborative Innovation Center of Trustworthy Software for Internet of Things (No. ZF1213).

### REFERENCES

[1] Mckeown, Nick, et al. "OpenFlow:enabling innovation in campus networks." *Acm Sigcomm Computer Communication Review* 38.2(2008):69-74.

[2] Kreutz D, Ramos F M, Verissimo P, et al. Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE*, 2015, 103(1): 14-76.

[3] Lara A, Kolasani A, Ramamurthy B. Network Innovation using OpenFlow: A Survey. *IEEE Communications Surveys & Tutorials*, 2014, 16(1):493-512.

[4] Kohler, Thomas, F. Drr, and K. Rothermel. "Update consistency in software-defined networking based multicast networks." *Network Function Virtualization and Software Defined Network IEEE*, 2016:177-183.

[5] Azodolmolky, Siamak. *Software Defined Networking with OpenFlow*. Packt Publishing, 2013.

[6] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice/Hall International, 1985.

[7] Lowe G, Roscoe B. Using CSP to detect errors in the TMN protocol. *IEEE Transactions on Software Engineering*, 1997, 23(10): 659-669.

[8] Roscoe A W, Huang J. Checking noninterference in Timed CSP. *Formal Aspects of Computing*, 2013, 25(1): 3-35.

The screenshot shows the 'Assertions' window in the PAT model checker. It contains a table of four assertions, each with a checked checkbox and a 'VALID' status. Below the table, there are four detailed verification result blocks, each corresponding to one of the assertions. Each block includes verification settings (Admissible Behavior, Method, LTL formula, System Abstraction) and verification statistics (Visited States, Total Transitions, Time Used, Estimated Memory Used).

Assertion ID	Assertion Text	Verification Result
1	BundleSystem() deadlockfree	VALID
2	BundleSystem() reaches Parallelism	VALID
3	BundleSystem()  = Atomicity	VALID
4	BundleSystem()  = Order	VALID

Fig. 2. Verification Results of Four Properties

[9] PAT, PAT: Process analysis toolkit. [Online]. Available: <http://pat.comp.nus.edu.sg/>

[10] J. Sun, Y. Liu, and J. S. Dong, Model checking CSP revisited: Introducing a process analysis toolkit, in *Leveraging Applications of Formal Methods, Verification and Validation*. Springer, pp. 307-322, 2009.

[11] Yuanjie, S. I., et al. "Model checking with fairness assumptions using PAT." *Frontiers of Computer Science* 8.1(2014):1-16.

# Helpful or Not? An investigation on the feasibility of identifier splitting via CNN-BiLSTM-CRF

Jiechu Li<sup>\*</sup>,<sup>†</sup> Qingfeng Du<sup>\*</sup>,<sup>†</sup> Kun Shi<sup>\*</sup>,<sup>†</sup>,<sup>§</sup> Yu He<sup>\*</sup>,<sup>†</sup> Xin Wang<sup>\*</sup>,<sup>†</sup> and Jincheng Xu<sup>\*</sup>,<sup>†</sup>

<sup>\*</sup>School of Software Engineering, Tongji University

<sup>†</sup>Software Engineering R&D Centre, Jishi Building, Tongji University

<sup>‡</sup>Smart City Labotary, Jishi Building, Tongji University

<sup>§</sup>Shanghai Research and Development Center, Baidu Inc.

Email: {lijiechu, Du\_cloud, skyline, rainlf, wangxin16, xujincheng}@tongji.edu.cn

**Abstract**—We recently introduced a new technique to handle source code identifier splitting. The proposed technique, denoted as CNN-BiLSTM-CRF[a neural network composed of a convolutional neural network(CNN), bidirectional long short-term memory networks(BiLSTM) and conditional random fields(CRFs)] enables us to obtain a model that splits identifiers correctly and effectively. This technique combines the use of a CNN layer with the mature BiLSTM-CRF model. The experimental results indicate that CNN-BiLSTM-CRF delivers outstanding performance on all four of the evaluation oracles. More importantly, we endeavored to provide insight into the practical feasibility of this technique by considering the aspects of generality, data size in demand and construction cost, etc. Finally, we reasoned out that CNN-BiLSTM-CRF should be helpful and improvable for identifier splitting in practical works in terms of the accuracy and feasibility. This was validated by multifaceted experiments.

**Index Terms**—identifier splitting, source code mining, program comprehension, CNN, BiLSTM-CRF, feasibility investigation

## I. INTRODUCTION

The rapid development of natural language processing (NLP) and machine learning has derived many prominent techniques[e.g., information retrieval(IR) model and deep learning] to support software engineering (SE) tasks[1], [2], [3], [4](e.g., feature location and traceability link recovery). These techniques extract domain concepts of corresponding software projects by analyzing textual information mined from software repositories.

Source code identifiers<sup>1</sup>, occupying most of the characters<sup>2</sup> insides programs[6], are one of the critical components that can be mined from software artifacts. To our knowledge, this is owing to the fact that strong consistency is always reflected between the source code and other software artifacts (e.g., documentation). However, the definitions of identifiers are strictly constrained by the syntax rules of programming languages, complicating not only the manual recognition process but also the automatic NLP tokenization process on them.

<sup>1</sup>Source code identifiers are tokens that name language entities including variables, types, functions, and packages in programming languages. These tokens represent different meanings, according to their naming purposes.

<sup>2</sup>An identifier  $id$  is normally in the form of  $(s_0, s_1, s_2, \dots, s_n)$ , where  $s_i$  is a letter, digit, or special character[5].

To be specific, an appointed source code identifier must be composed of a series of terms without explicit blanks (e.g., *XMLParser1*, *dorapntr*, and *treeNode*) in which the terms are normal dictionary words (e.g., *Parser*) and acronyms (e.g., *XML*), abbreviations or, even worse, unmeaningful vocabs. Directly performing NLP procedures (e.g., word embedding) on these irregular strings inevitably degrades the performance of ongoing or upcoming concept-comprehension tasks. Hence, normalizing identifiers into their constituent parts is crucial when leveraging NLP techniques on SE tasks.

Identifier splitting is the first and highly critical step of identifier normalization, with the subsequent step of mapping or expanding the correct split terms into their original dictionary words[7], [8]. Identifier splitting is also considered as an indispensable procedure after tokenization[9], [10]. In fact, existing techniques have already been put forward with the aim to efficiently and correctly split identifiers, including the investigated approach in our study: identifier splitting via a trained **CNN-BiLSTM-CRF** model(CNN-BiLSTM-CRF is used to represent this approach itself in the following literature for short).

The mentioned CNN-BiLSTM-CRF, is derived from our early research achievements. It is proposed under the motivation for better identifier splitting techniques that better contribute to related SE tasks[11]. Our pioneering research and initial experimental results showed that the CNN-BiLSTM-CRF significantly outperforms other state-of-the-art techniques. However, it remains certain thoughtlessness before we resolve to apply this technique to split identifiers in practical SE tasks. These issues certainly exist and are determined by the methodology of CNN-BiLSTM-CRF, which takes samples of manually built oracles as input for training, then tunes and evaluates the outcome model with the rest of the oracle samples. In terms of practical feasibility, there are concerns regarding the **generalization capability** (refers to the ability to correctly split identifiers outside of the oracle with which it was trained) and the **training cost** of the model in addition to the superficial **outstanding performance**.

In this study, our specially designed experiments and qualitative analysis on the obtained results lead to the conclusion that splitting identifiers via CNN-BiLSTM-CRF is helpful and absolutely feasible in terms of the above indicated facets.

In summary, we make the following contributions:

- We prospectively present a new and novel identifier splitting method called CNN-BiLSTM-CRF, which possesses great ability to split source code identifiers accurately.
- We systematically investigate and inspect the practicality of the CNN-BiLSTM-CRF approach regarding its comprehensive aspects.
- We offer evidence that the investigated approach is not only superior to other state-of-the-art identifier splitting techniques but also of enough generality and feasibility based on empirical evaluations performed on a maximum of four benchmark oracles.
- We provide the publicly available package and implemented code for researchers to adopt, replicate, or further explore our work.

The remainder of this paper is structured as follows: Section II presents some significant related work on identifier splitting and concisely describes the mechanism of our proposed CNN-BiLSTM-CRF approach. Section III demonstrates the process of setting up our experiments and discusses the results. Section IV is a deeper discussion of our achieved results and of threats to the validity of this method. Section V concludes this study.

## II. BACKGROUND AND RELATED WORK

### A. Previous Work on Identifier Splitting

Ideally, identifiers in recognizable forms such as the well-known *CamelCase* and *UnderscoreCase* can be effortlessly split by explicit rules (splitting at underscores or changes from lower case to upper case). However, extra strategies still need to be considered to effectively solve the splittings on identifiers in more sophisticated forms (e.g., all characters are in single case or containing digits). In this case, researchers are spurred to promote the performance of identifier splitting, and hence many innovative techniques have been introduced (e.g., [5], [12], [13], [14], [15] and etc.).

Here, we selectively and briefly introduce several of these state-of-the-art identifier splitting techniques. Field et al.[12] first attempted to split identifiers with a simple **greedy** algorithm and used a **simple artificial neural network** approach. **DTW**[14] is based on a modified version of *Dynamic Time Warping*. The fundamental idea behind this approach is to determine the optimal matching of two series of characters  $(x_1, x_2, \dots, x_n)$  and  $(y_1, y_2, \dots, y_m)$ . **GenTest**[7] attempts to score all possible splits for an identifier with a series of *handcrafted metrics*, and the one with the highest score is considered to be the correct split. **LINSEN**[15] uses an efficient approximate string-matching algorithm, *BYP*, in conjunction with nested context-based dictionaries.

A number of man-made oracles for identifier splitting were created from past accomplishments. Thereupon then previous studies could evaluate their techniques on these available oracles. In our study, four frequently used oracles namely *Binkley*[16](containing 2,663 samples), *BTII*[17](21,122), *Jhotdraw*(974), and *Lynx*(3,085)[14], will be applied.

### B. Reformulating Identifier Splitting as a Sequence Labeling Task

The essence of splitting an identifier is to determine at which adjacent positions to insert splits. Thus, we can draw on the experience of the advancing thought to solve Chinese word segmentation[18]—**sequence labeling**. Similarly, identifier splitting can be subtly mapped to a sequence labeling task:

For a given identifier  $id$  with characters sequence  $(c_1, c_2, \dots, c_n)$ , we label each  $c_i$  with a corresponding tag. The set of candidate tags contains only B, M, E, S, and N.

Assuming the unified max-length of the accepted identifier is  $T$ , the interpretation of each tag is as follows: **B**, denoting the beginning of a term; **E**, denoting the end of a term; **M**, denoting the position between the beginning and end of a term; **S** denoting a term with only one character; and **N**, denoting the spare positions of an identifier if its length is less than  $T$ .

After tagging all  $c_i$  of  $id$ , we subsequently insert splits between any adjoining  $c_n$  and  $c_m$  with successive tags **EB**, **ES**, **SB**, or **SS**. For instance, we input identifier *nthreadpool* and obtain the output *SBMMMMMEBMMME*. Based on the rules summarized above, two splits are inserted between  $n$  and  $t$ , and  $d$  and  $p$ , respectively. Consequently, we obtain the splitting result *n-thread-pool*.

### C. CNN-BiLSTM-CRF Model

According to the above discussions, the oracles of identifier splitting can be transferred into the form of sequence labeling accordingly. On this basis, we are able to train a **deep model** for labeling newly input identifiers, and then infer correct splittings of them. The considered deep model is actually our proposed CNN-BiLSTM-CRF model.

We vary the basic **BiLSTM-CRF**[19] network by adopting an additional CNN layer with the aim to extract finer-grained morphology features inside identifiers.

The architecture of CNN-BiLSTM-CRF is illustrated in Fig. 1, showing a slice of the entire network. The left side of Fig. 1 shows the general framework of BiLSTM-CRF. It differs from the traditional framework because the input layer of BiLSTM-CRF takes the replaced *CNN-processed char representation* of the identifier as input.

In particular,  $C_t$  denotes the input character at time  $t$  encoded using one-hot encoding<sup>3</sup>, and  $Conv_t$  denotes the  $t^{th}$  vector after convolution on the original input layer. To ensure one-to-one correspondence, we alternatively add padding around the beginning and end of the original input char representation. After that, we concatenate the output of the convolution layer and the original char representation to generate the final *CNN-processed char representation*. This gives the interpretation of  $Conc_t$ , which denotes the  $t^{th}$  concatenated vector. It is also worth mentioning that we skip the *pooling layer* with the aim of preserving all information of the input identifier in our task.

<sup>3</sup>We perform character-level embedding on each input character with a **one-hot** vector (with shape  $1 \times n$ ). This means there is only one component equal to 1 in this vector. For example,  $C_t$  can be  $[0, 1, 0, \dots, 0]$ .

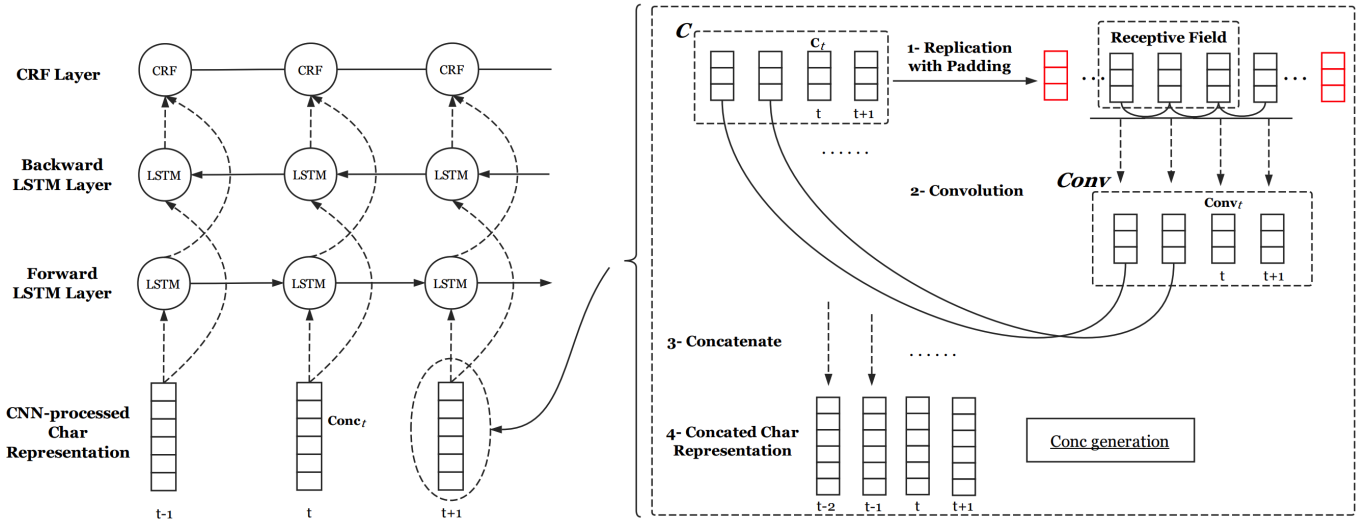


Fig. 1. Architecture of CNN-BiLSTM-CRF. Red rectangles indicate the padding for the CNN layer.

Suppose that  $K$  is the convolution core with shape  $m \times n$  ( $n$  is equal to the length of dimensionality of the one-hot encoded vector). The convolution layer is calculated as follows:

$$Conv_i = f\left(\sum_i C'_{i+m} \circ K_m\right) \quad (1)$$

where symbol  $f$  denotes the activation function of *convolution* layer (normally the **Relu** function), the  $\circ$  operation produces the *Hadamard product* of two vectors, and  $C'_i$  denotes the  $i^{th}$  vector of padded matrix  $C$ , defined as the original input layer in Fig.1.

Consequently, the concatenated representation of an input char is expressed by merging two corresponding vectors of  $Conv_i$  and  $C_i$ , as shown in Equation 2.

$$Cont_{i(2 \times n)} = (Conv_{i(1 \times n)} \quad C_{i(1 \times n)}) \quad (2)$$

With regard to the remaining BiLSTM-CRF, CRF on the top of BiLSTM takes the final softmax layer of the BiLSTM layer as input. The softmax layer produces the probability distribution over labels (i.e., B, M, E, S, and N). The CRF layer calculates the **emission scores** and **transition scores** based on the probability. The emission score of a label can be directly expressed by reusing its output probability, while the transition score is a bit difficult to deduce. This requires us to maintain a  $5 \times 5$  matrix (5 meaning the number of all possible labels), which records the transition score from one label to another label. In nature, the matrix is randomly initialized with small values. To elaborate, for a given output path  $p$  of an identifier  $B \Rightarrow M \Rightarrow E \Rightarrow S$ , we have

$$EmissionScore_p = X_{0,B} + X_{1,M} + X_{2,E} + X_{3,S} \quad (3)$$

$$TransitionScore_p = T_{B,M} + T_{M,E} + T_{E,S} \quad (4)$$

Therefore, for a path  $p$ , the final score of  $p$  is calculated as

$$Score_p = EmissionScore_p + TransitionScore_p \quad (5)$$

The final scores of other possible paths besides  $p$  are calculated in the same way. Among all paths, the path with the highest final score is actually thought to be the real path (i.e., correct path). After determining how to calculate the final score for each possible path, we are now able to define the CRF loss function:

$$LossFunction = -\log\left(\frac{Score_{Realpath}}{\sum_{n=1}^N Score_n}\right) \quad (6)$$

During the training process, the parameter values (including the transition score matrix) of our BiLSTM-CRF model are updated repeatedly to continue increasing the percentage of the score of the real path. Because only bigram interactions between outputs are being modeled, the total scores for all possible paths can be computed using dynamic programming[20]. The log probability of the correct tag sequence is maximized by automatically reducing the loss of Equation 6. When the model is ready, we can infer the labels for the input identifier by using a Viterbi decoding algorithm based on the transition score matrix and the output of the BiLSTM layer.

### III. EXPERIMENTAL VALIDATION

In this section, we describe the series of experiments conducted on the four oracles mentioned previously. We chose 30 to be the unified length of the input layer because we found that identifiers with the length less than 30 occupy more than 96% of the samples on average for all datasets.

In addition, approximately 5,000 manually constructed identifiers<sup>4</sup> were collected as extra training materials, with which we could optionally supplement the training set partitioned from a given oracle. Specifically, these **fake-identifiers** were used in Experiment<sub>III-B</sub> and Experiment<sub>III-C</sub> to further enhance the performance of trained models.

The hyperparameters in use and details of the proposed CNN-BiLSTM-CRF structure are summarized in Table I.

<sup>4</sup>They were generated with the aid of SCOWL – <http://wordlist.aspell.net/>



TABLE I  
THE HYPERPARAMETERS & STRUCTURAL DETAILS TO CONSTRUCT  
OUR PROPOSED CNN-BiLSTM-CRF MODELS

Component	Hyperparameter	Value (Option)
CNN layer	Number of layers	1
	Filter shape	$3 \times n^1$
	Filter stride	1
	Activation function	Relu
BiLSTM layer	Number of layers	2
	Dropout probability	0.0
Fully connected layer	Number of layers <sup>2</sup>	1
	Activation function	None
General settings of training process	Initial learning rate	1.0
	Learning rate decay <sup>3</sup>	0.5
	Max gradient norm	5
	Max training epoch	13
	Batch size	25

<sup>1</sup> As mentioned in Section II.C, the  $n$  is equal to the length of dimensionality of the one-hot encoded vector.

<sup>2</sup> The number of nodes of the hidden layer is set to 200.

<sup>3</sup> The learning rate will decay dynamically from one training epoch to next epoch based on an exponential decay model.

### A. Evaluation Measures

We use the measure **Accuracy** to evaluate the performance of our trained models. To calculate the accuracy, we need to measure whether every predicted split of an identifier is exactly equal to the corresponding correct split in the oracle (i.e., the percentage of correct splitting). Thus, it is simply calculated as follows:

$$Accuracy = \frac{Num(\text{correctly split identifiers})}{Num(\text{all identifiers})} \quad (7)$$

Several previous studies (e.g., [8], [13]) additionally used *Precision*, *Recall*, and *F-measure* to measure the local ability of evaluating techniques. However, we posit that these measures always follow the same trend with *accuracy* and behave extremely congruously, based on observations of our early research and previous experimental results. In addition, most other studies highly related to this topic used no other kinds of measures (e.g., *ROC-AUC* and *Cohen's kappa*<sup>5</sup>). Therefore, we premit considering other metrics in order to make our outcomes clear.

### B. Investigation of the ability to split identifiers

This experiment was set up to validate the effectiveness of our proposed technique intuitively.

<sup>5</sup>There are two reasons for the unsuitability of applying *Cohen's kappa* or *ROC-AUC* to this multiclass classification task: 1) Our task additionally concerns the sequential relatedness of labels, and 2) other baseline techniques used as benchmarks in this study infer identifier splitting on the *word level*, whereas our proposed technique is on a different *character level*.

TABLE II  
ACCURACY OF IDENTIFIER SPLITTING COMPARED WITH OTHER  
STATE-OF-THE-ART APPROACHES

Splitting Approach	Oracle			
	Binkley	BT11	Jhotdraw	Lynx
CNN-BiLSTM-CRF	<b>0.817</b>	<b>0.936</b>	0.912	<b>0.876</b>
GenTest	0.701	0.723	0.795	0.489
INTT	0.774	0.820	0.844	0.625
LIDS	0.713	0.811	0.903	0.542
DTW	-	-	0.931	0.703
LINSEN	-	-	<b>0.949</b>	0.803

To assess the prediction performance of our trained models in a general and proper way, a **10-Time Hold-out Validation**<sup>6</sup> based on **Train/Validation/Test Set Splitting**<sup>7</sup> was used in this experiment. In other words, for every specific oracle, we iteratively trained and evaluated models 10 times, and the average of all results was subsequently used as the final estimation result on our models.

Several techniques (listed in Table II) were selected for benchmarking and were applied to split these oracles at the same time. Slightly different from our proposed technique, these compared techniques performed on all samples of the four oracles because no training processes were used in their cases. Specifically, we implemented LIDS[13], GenTest[7], and INTT[17] because they are provided with publicly available tools. For those that do not have publicly available tools, i.e., DTW and LINSEN, we tried obtaining their experimental results from study[15].

The experimental results are listed in Table II, with the best accuracies among all compared approaches in bold. The results show the following: On the Binkley dataset, CNN-BiLSTM-CRF achieves the highest accuracy (81.7%). On the Jhotdraw dataset, our proposed technique is suboptimal (91.2%) among all techniques. LINSEN seems to perform best on Jhotdraw since it has the highest accuracy (94.9%). However, on BT11 and Lynx, oracles with relatively more samples for training, the performance of our proposed technique is vastly superior to that of all other techniques. In particular, in terms of accuracy, CNN-BiLSTM-CRF outperforms the second best technique by more than 7%–11%.

We conducted a hypothesis test (**Pearson chi-square test** and **odds-ratio**) to investigate whether the difference in performance between CNN-BiLSTM-CRF and other state-of-the-art techniques varies significantly. To simplify our results and

<sup>6</sup>One round of the  $k$ -time hold-out validation (a.k.a. *Monte Carlo Cross Validation*[21]) involves randomly partitioning samples into disjoint subsets, of which at least one is used as training set and at least one is used as test set. Then, the model fit by the training set is evaluated with the test set. This process is independently repeated  $k$  times, where the partitions of samples are accomplished in the same random manner for each run.

<sup>7</sup>In this *three-way split*, the original dataset is randomly partitioned into three parts at each time: 70% for the training set, 15% for the validation set to help avoid overfitting during the training process, and the remaining 15% to evaluate the performance of the model.

TABLE III

SIGNIFICANCE TEST FOR DIFFERENCE IN ACCURACY BETWEEN CNN-BiLSTM-CRF AND OTHER STATE-OF-THE-ART APPROACHES

Oracle	Compared Approach	Chi-square Test	Odds-ratio
Binkley	INTT	$p=0.03750$ (Significant)	1.35(Better)
	LIDS	$p<0.00001$ (Significant)	1.80(Better)
BT11	INTT	$p<0.00001$ (Significant)	3.21(Better)
	LIDS	$p<0.00001$ (Significant)	3.41(Better)
Jhotdraw	DTW	$p=0.4951$	0.80(Worse)
	LINSEN	$p=0.1069$	0.59(Worse)
Lynx	DTW	$p<0.00001$ (Significant)	2.95(Better)
	LINSEN	$p<0.00001$ (Significant)	1.71(Better)

\*<sup>1</sup> The difference between two approaches is significant if the adj. p-value statistic is less than the significance level (0.05).

\*<sup>2</sup> Odds-ratio greater than one means the treatment approach (CNN-BiLSTM-CRF) performs better than the compared control approach.

make things clearer, we chose only the top two well performing techniques except CNN-BiLSTM-CRF for comparison. The statistical results of the hypothesis test are listed in Table III, from which we learn the following:

- CNN-BiLSTM-CRF performs significantly more accurately than other techniques on the Binkley, BT11, and Lynx datasets.
- Even though CNN-BiLSTM-CRF achieves lower accuracy than LINSEN on Jhotdraw, there is no significant difference between them with the p-value of a chi-square test of 0.1069, which does not exceed the critical value of 0.05. This means our technique, for Jhotdraw, performs at least in line with LINSEN (i.e., the technique with the best performed accuracy on Jhotdraw).

### C. Investigation of external generality

This part of experiments was set up to explore the extensive feasibility of applying the CNN-BiLSTM-CRF model to split identifiers outside of the original oracle with which it was trained. Hence, to realize this aim, we trained models on four training sets, namely, *Binkley*, *BT11*, *Jhotdraw*, and *Lynx*. Then, we evaluated the accuracies of trained models by other evaluation sets.

The experimental results are shown in Table IV. The top header of Table IV lists the training sets with which we trained the models. The sidebar lists all evaluation sets. Hence, each cell inside the table refers to the average accuracy we obtained from the models that were trained and estimated with a specific pair of training and evaluation sets. It illustrates that the performance of models on other oracles which they were not trained with, by and large, approaches to the ideal performance reported in Table II (despite declines at different degrees). We even see a slight improvement in Jhotdraw from the model trained with BT11. Not surprisingly, the models trained with *Jhotdraw* perform worst since Jhotdraw contains a limited number of samples (only 974).

TABLE IV

EXTERNAL PERFORMANCE (i.e., ACCURACY OF SPLITTING) OF CNN-BiLSTM-CRF MODELS

$S_{\text{Evaluation}}$ \ $S_{\text{Training}}$	Binkley	BT11	Jhotdraw	Lynx
Binkley	-	0.760	0.655	0.703
BT11	0.833	-	0.796	0.814
Jhotdraw	0.896	0.936	-	0.883
Lynx	0.782	0.844	0.709	-
<b>Total Avg.</b>	<b>0.829</b>	<b>0.825</b>	<b>0.773</b>	<b>0.805</b>

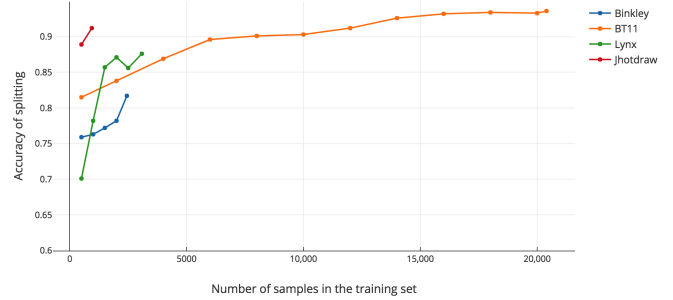


Fig. 2. Effect of training set size on model performance.

### D. Investigation of effect of training set size

By picking and training samples with sizes from small to large in every fixed intervals from all datasets, we built CNN-BiLSTM-CRF models and then evaluated them by recording the related data of the accuracies. Similarly, for each size, we trained 10 models and acquired their average results. The results are shown in Fig. 2, which indicates that models trained on all different datasets follow the same trend. This reveals that as the size of training set increases, the performance of the model improves.

On the largest dataset BT11, however, it is noteworthy that the increment gradually slows down when the size of the training sets exceeds 6,000. It seems that 6,000 could be an appropriate size of training sets if we want to strike a **compromise** between the model performance and training cost under this circumstance.

### E. Investigation of training cost

We determined four categories of training set size, namely 2000, 8000, 15000, and 27000, to survey whether our proposed technique is time-consuming. Specifically, we conducted 10 groups of experiments on four randomly constructed oracles which are corresponding to the above-mentioned four sizes. For each group, we separately trained four models on these oracles, and respectively recorded their training times (graphically provided in Fig. 3). Experiments from group *No. 1* to *No. 10* were carried out independently, but the plotting line of each corresponding size revealed strong consistency and stability on the training time.

To deeply inspect whether the computational time would be influenced by different constructions of samples, we randomly

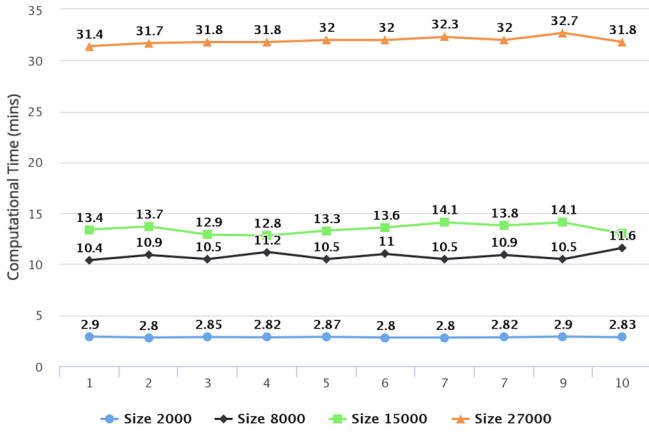


Fig. 3. Computational time of training models on training sets of different sizes. The No. on the x-axis represents that it is the  $i^{th}$  group of experiment. The value on the y-axis represents computational time of training a specific model once.

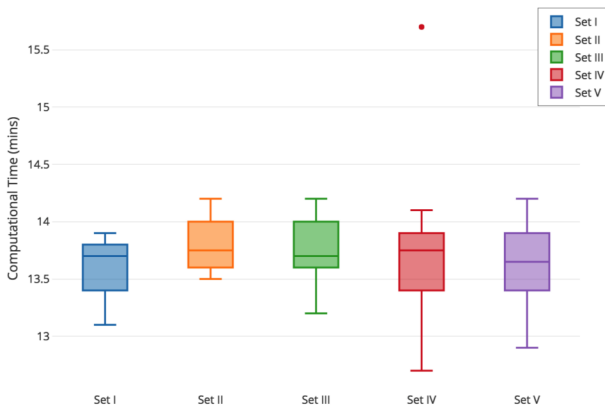


Fig. 4. Training cost of five sets of data with 15,000 samples.

constructed four extra training sets, all with a size of 15,000, to perform similar experiments. We used the *box plot* to measure the dispersion of the computational time of each set. The results are shown in Fig. 4, of which each horizontal line in a box refers to the average training time. We found out that models on all sets of samples finished their training with a nearly identical median of training time (**13.7 min**).

From all the illustrated results, we concluded that

There is not a significant amount of time for training (It takes about, on average, 10 minutes to train a model with 8,000 samples and 31 minutes with 27,000 samples.). In addition, the training time is shown to be sufficiently stable.

These data were produced and collected from a PC with 1\* NVIDIA P100, Octa-core 2.5-GHz CPU and 16 GB of RAM.

#### IV. DISCUSSION

##### A. Deeper insight into CNN-BiLSTM-CRF approach

We determined that the CNN-BiLSTM-CRF model is suitable for handling identifier splitting after pondering the relationship between this adopted model and domain-specific data (i.e., identifiers).

Through utilizing the characteristics of a CNN to perform character-grained feature extraction, the model is capable of learning the word morphology transformations. This provides an expansion ability for our technique to effectively handle multiple forms of a term. This will become more significant if models are trained on large datasets. Furthermore, the BiLSTM network enables the model to automatically learn contextual information, i.e., whether to insert a split in the current position according to characters in front and behind. Third, the CRF layer helps to eliminate those candidates splitting with few probabilities (e.g., tag M should never be followed by B or S). By integrating all of the needed abilities, our proposed technique certainly performs well.

We noticed from the above results that when our proposed technique was applied to oracles with a sufficient number of samples, the performance was significantly better than oracles with fewer samples. This is because neural networks always perform more poorly on small datasets. A larger training set provides richer information for a neural network to learn and to avoid overfitting. (Table II and Fig. 2 both strongly confirm this point.)

##### B. Possibility for improvement

After observing 200 incorrectly split identifiers inferred by our trained models, we figured out that a large proportion of them (93/200) were caused by a lack of training materials. This caused our models to fail to recognize terms inside these identifiers (e.g., *NXScanf*, where *Scanf* is the name of a C library function).

Given the extensibility of our technique, we are able to address this problem easily by mining more complete identifiers or constructing more customized *fake-identifiers*. These samples are then appended to the oracle before training a model. In addition, we believe that some unused deep learning or tuning skills provide the possibility of improving the performance from the model itself.

##### C. Is CNN-BiLSTM-CRF really feasible?

We deduce the answer of this question to be yes, based on all of the aforementioned outcomes. First, this proposed approach is superior if the training samples are customized and sufficient enough (according to the results in part B of Section III). Second, the model is of great extensibility. The results in Table IV indicate that most trained models achieve a total average accuracy of 80% when splitting those identifiers which they never met. This convinced us that it is entirely possible to construct a model with sufficient generality. Although we failed to obtain good performance on Jhotdraw, to some extent this is insignificant because we will certainly ensure that our models are trained on oracles with enough samples in practice. Third, training the model is feasible with regard to time. Only a little time is needed to construct an available CNN-BiLSTM-CRF model, and this is within the acceptable range. (Samples of all oracles are within 21,000 in our study, but all of them led to good performance). The cost of inferring a newly received identifier could even be omitted.

#### D. Threats to validity

The validity of our study has three major threats.

The threat to **content validity** is that using only four oracles as in our study seems to be insufficient for the hypothesis tests to be reliable. Although they contain identifiers that originated from several kinds of programming languages (i.e., C, C++, and Java), they still cannot represent all kinds of languages. Moreover, some of these oracles may not contain enough multiplex samples because of their small size.

Threats to **internal validity** include the hyperparameters of constructing the CNN-BiLSTM-CRF network and the randomness of the training process of a neural network. We selected the hyperparameters (e.g., the activation function and batch size) based on practical experience from the academic area.

The threat to **instrumental validity** is that we merely observed accuracy data on the Jhotdraw and Lynx datasets of LINSSEN and DTW from a previous study. However, we still made a strong effort to compare our proposed technique with other state-of-the-art techniques on two other datasets.

We realized our approach with **Tensorflow 1.4**, a mature deep learning framework maintained by Google. The ready-made package to split identifiers, implemented code, oracles, and indications to reproduce our experimental results can be found in our Github repository<sup>8</sup>.

#### V. CONCLUSION

We described a completely new approach, CNN-BiLSTM-CRF, to perform identifier splitting. In the premise of properly setting up the model, this approach is superior to other state-of-the-art techniques. We further investigated the feasibility of the proposed technique. The experimental results jointly demonstrated that the technique is practical and efficient with regard to training cost and the demand on the size of the training sets. The results also showed that this approach exhibits good generalization performance on various datasets, including splitting identifiers outside from trained-with oracles. Ultimately, splitting identifiers via CNN-BiLSTM-CRF is proven to be helpful in practice, in combination with all the qualitative and quantitative analyses in this study.

Detailed heuristics and processes of constructing CNN-BiLSTM-CRF models will be introduced in the next phase of our research. In the future, we will also improve the feasibility and generality of this approach by integrating *transfer learning* techniques (e.g., investigation of cross-programming-language identifier splitting).

#### ACKNOWLEDGMENT

Sincere appreciation to Li Sun, Juan Qiu, Robbie Xie, and Kanglin Yin, who provided us with valuable comments and tremendous encouragement. We are also grateful toward S. Butler, D. Binkley, and Madani et al. because it would not have been possible to complete this work without their previous substantial efforts on the topic of identifier splitting and their publicly available oracles.

<sup>8</sup><https://github.com/jaki2012/IdentifierSplitting-SEKE2018>

#### REFERENCES

- [1] M. White, C. Vendome, M. Linares-Vásquez, and D. Poshyvanyk, "Toward deep learning software repositories," in *Mining Software Repositories (MSR), 2015 IEEE/ACM 12th Working Conference on*, pp. 334–345, IEEE, 2015.
- [2] M. White, M. Tufano, C. Vendome, and D. Poshyvanyk, "Deep learning code fragments for code clone detection," in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, pp. 87–98, ACM, 2016.
- [3] P. W. McBurney and C. McMillan, "Automatic source code summarization of context for java methods," *IEEE Transactions on Software Engineering*, vol. 42, no. 2, pp. 103–119, 2016.
- [4] P. W. McBurney and C. McMillan, "Automatic documentation generation via source code summarization of method context," in *Proceedings of the 22nd International Conference on Program Comprehension*, pp. 279–290, ACM, 2014.
- [5] E. Enslin, E. Hill, L. Pollock, and K. Vijay-Shanker, "Mining source code to automatically split identifiers for software analysis," in *Mining Software Repositories, 2009. MSR'09. 6th IEEE International Working Conference on*, pp. 71–80, IEEE, 2009.
- [6] F. Deissenboeck and M. Pizka, "Concise and consistent naming," *Software Quality Journal*, vol. 14, no. 3, pp. 261–282, 2006.
- [7] D. Lawrie, D. Binkley, and C. Morrell, "Normalizing source code vocabulary," in *Reverse Engineering (WCRE), 2010 17th Working Conference on*, pp. 3–12, IEEE, 2010.
- [8] E. Hill, D. Binkley, D. Lawrie, L. Pollock, and K. Vijay-Shanker, "An empirical study of identifier splitting techniques," *Empirical Software Engineering*, vol. 19, no. 6, pp. 1754–1780, 2014.
- [9] S. W. Thomas, A. E. Hassan, and D. Blostein, "Mining unstructured software repositories," in *Evolving Software Systems*, pp. 139–162, Springer, 2014.
- [10] S. Wang, D. Lo, B. Vasilescu, and A. Serebrenik, "Entagrec: An enhanced tag recommendation system for software information sites," in *Software Maintenance and Evolution (ICSME), 2014 IEEE International Conference on*, pp. 291–300, IEEE, 2014.
- [11] B. Dit, L. Guerrouj, D. Poshyvanyk, and G. Antoniol, "Can better identifier splitting techniques help feature location?," in *Program Comprehension (ICPC), 2011 IEEE 19th International Conference on*, pp. 11–20, IEEE, 2011.
- [12] H. Feild, D. Binkley, and D. Lawrie, "An empirical comparison of techniques for extracting concept abbreviations from identifiers," in *Proceedings of IASTED International Conference on Software Engineering and Applications (SEA'06)*, 2006.
- [13] N. R. Carvalho, J. J. Almeida, P. R. Henriques, and M. J. Varanda, "From source code identifiers to natural language terms," *Journal of Systems and Software*, vol. 100, pp. 117–128, 2015.
- [14] N. Madani, L. Guerrouj, M. Di Penta, Y.-G. Gueheneuc, and G. Antoniol, "Recognizing words from source code identifiers using speech recognition techniques," in *Software Maintenance and Reengineering (CSMR), 2010 14th European Conference on*, pp. 68–77, IEEE, 2010.
- [15] A. Corazza, S. Di Martino, and V. Maggio, "Linsen: An efficient approach to split identifiers and expand abbreviations," in *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*, pp. 233–242, IEEE, 2012.
- [16] D. Binkley, D. Lawrie, L. Pollock, E. Hill, and K. Vijay-Shanker, "A dataset for evaluating identifier splitters," in *Proceedings of the 10th Working Conference on Mining Software Repositories*, pp. 401–404, IEEE Press, 2013.
- [17] S. Butler, M. Wermelinger, Y. Yu, and H. Sharp, "Improving the tokenization of identifier names," *ECOOP 2011—Object-Oriented Programming*, pp. 130–154, 2011.
- [18] N. Xue, "Chinese word segmentation as character tagging," *International Journal of Computational Linguistics & Chinese Language Processing, Volume 8, Number 1, February 2003: Special Issue on Word Formation and Chinese Language Processing*, vol. 8, no. 1, pp. 29–48, 2003.
- [19] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging," *arXiv preprint arXiv:1508.01991*, 2015.
- [20] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," *arXiv preprint arXiv:1603.01360*, 2016.
- [21] Q.-S. Xu and Y.-Z. Liang, "Monte carlo cross validation," *Chemometrics and Intelligent Laboratory Systems*, vol. 56, no. 1, pp. 1–11, 2001.

# *How to Incorporate a Usability Technique in the Open Source Software Development Process*

Lucrecia Llerena, Nancy  
Rodriguez

Dpto. de Ingeniería Informática  
Universidad Autónoma de Madrid  
Madrid, Spain  
lucrecia.llerena@estudiante.uam.es,  
nancy.rodriguez@estudiante.uam.es

John W. Castro

Dpto. de Ingeniería Informática y  
Ciencias de la Computación  
Universidad de Atacama  
Copiapó, Chile  
john.castro@uda.cl

Silvia T. Acuña

Dpto. de Ingeniería Informática  
Universidad Autónoma de Madrid  
Madrid, Spain  
silvia.acunna@uam.es

**Abstract**— The growth in the number of non-developer open source software (OSS) application users and the escalating use of these applications have both created a need for, and interest in, developing usable OSS. OSS communities are unclear about which techniques to use in each activity of the development process. The aim of our research is to adopt the visual brainstorming usability technique in the HistoryCal OSS project and determine the feasibility of adapting the technique for application. To do this, we participated as volunteers in the HistoryCal project. We used the case study research method to investigate technique application and community participation. As a result, we identified adverse conditions that were an obstacle to technique application and modified the technique to make it applicable. We can conclude from our experience that these changes were helpful for applying the technique, although it was not easy to recruit OSS users to participate in usability technique application.

**Keywords**- *open source software; usability techniques; requirements engineering; visual brainstorming.*

## I. INTRODUCTION

The usability technique definition and integration into OSS projects is a complicated process, about which there are few papers [1]–[4]. These papers suggest that usability techniques should be reconceptualized, but they do not explain how the OSS community should go about adaptation. Nichols and Twidale [5] are the only authors to put forward some general ideas for improving usability. However, the issues to be taken into account to adopt such techniques in OSS developments are unclear. In particular, few studies have reported the application of the visual brainstorming (VB) technique in OSS projects [4], [6]. It appears to be less straightforward to integrate usability into the OSS development process than into commercial development due to some of the characteristics of the OSS community. Consequently, usability technique adoption is a demanding task because most HCI techniques are not designed for the type of environment in which OSS is developed [7].

On one hand, the human-computer interaction (HCI) field offers usability techniques whose key aim is to build usable software. However, they are applied as part of HCI methods and not within the OSS development process. On the other hand, the OSS development process focuses on source code and thus on feature development. The OSS development process has a number of characteristics (i.e., a developer’s culture that may be somewhat distant from the interaction). This prevents many of the HCI usability techniques from being adopted directly [7].

Requirements engineering activities play a very important role in the success or failure of an OSS project. However, they are sometimes extremely hard to perform because there is no definition of OSS user segments before the software is developed. Also, it is far from straightforward to address all the requirements analysis activities due to the particular characteristics of OSS development groups. Additionally, OSS projects have not adopted many usability techniques related to the requirements engineering and product concept development activities [7]. The next step after selecting the activity is to pick one related usability techniques for adoption in the OSS development process. VB is a technique involving idea-sketches used to explore alternative designs [8]. This technique helps to focus product concept design on its hypothetical features [9], that is, developers can use this technique to discover a user’s mental model of the product. We selected the VB from among other techniques because it can benefit the user interface (UI) design process: it generates creative ideas in the process of solving specific problems, with one hour of session they obtain positive results, supports Conceptual design by generating metaphors for UI architectures and providing new ways to implement old functionalities [10]. On this ground, we selected the VB technique for adoption in an OSS project.

Our research spans two areas: OSS developments and HCI. We use usability techniques as a bridge to communicate these two areas, where our aim is to deploy HCI knowledge in the OSS development communities. If adapted, usability techniques can be adopted in the OSS development process [7]. Therefore, the aim of our research is to adapt the VB usability technique [11] for adoption in the OSS development process

and determine the feasibility of adopting this usability technique in a real OSS project. To do this, we first identified and analysed which obstacles had to be overcome in order to apply VB in OSS projects. This paper makes a significant contribution to the field of SE and particularly OSS development projects because there are few papers reporting the use of the VB technique and detailing how it has been applied in OSS development projects [4], [6]. The contributions of this paper are as follows: (i) We identify the adverse conditions that are an obstacle to the application of the VB technique in OSS developments, (ii) we propose adaptations for each of the steps of this technique to enable its adoption in the OSS development process, as there are no specified procedures for applying HCI techniques in this type of development projects, (iii) we give some recommendations on how to improve the UI of an OSS application by applying a collaborative usability technique (VB) to tailor the original interface design to real user needs.

This paper is organized as follows. After this introduction, we describe the related works. We then illustrate the research method. Then, we report the proposed solution. We then report the results and discussion. Finally, we outline the conclusions.

## II. RELATED WORKS

In recent years, the worldwide OSS community has adopted just over 50% of the HCI techniques related to evaluation. However, only about 20% of the usability techniques related to requirements engineering and design activities have been adopted [7]. Therefore, more research is required to support the adoption of techniques related to requirements engineering in OSS developments. In view of the importance of HCI and SE, it is only logical to study the user-centred software development activities in OSS projects. This is especially true of the requirements engineering stage, because the discovery of user requirements during the early development activities is useful for putting right any defects in software detected later on [4]. In this paper, we adapt the VB technique used in the product concept development activity. According to Preece et al. [11], product concept development relies on the creation of a mental model based on psychological theories related to HCI. Ferré [8] explains that this activity covers issues regarding how users envisage the system. Therefore, this activity aims to provide a picture of the product before defining the features that the system should offer.

There are papers in the literature reporting the usability evaluation of some OSS applications [12], [13]. Assa et al. [13] study the usability issues facing software developers using code analysers by evaluating one of the popular open-source static-code analysis tools. Ternauciuc and Vasiliu [12] tried to inventory the existing methods for testing and improving usability, with a particular focus on e-learning platforms.

In particular, very few studies have reported the application of the VB technique in OSS projects [4], [6]. In the Carrot2 OSS project, the original application was redesigned according to its target end users (data mining researchers). This project adopted the VB technique in order to generate ideas for designing the new interface [6]. According to Osiński and Weis [6], the technique was applied as prescribed by HCI. However, the VB technique reported by Terry et al. [4] was adapted for

adoption to develop a bit map graphics application. In the adapted technique, ideas were gathered using a wiki instead of at face-to-face meetings as established by HCI. Thanks to the wiki, anybody involved in the project could put forward his or her interface design ideas.

On the other hand, Castro [7] proposes a framework for integrating usability techniques into OSS developments. This framework is composed of a number of general adaptations in response to the adverse conditions for adopting usability techniques in OSS development projects. Castro [7] has identified the unfavourable conditions that give rise to adaptations. Unfavourable conditions are classified into three groups (families of adaptations). First, some usability techniques require an expert in usability (most OSS projects do not involve experts). Second, certain techniques require the participation of users or that several of them are physically gathered (OSS users are geographically distributed throughout the world). Finally, some techniques require several steps for their execution, a previous preparation or need some initial information (the work in the OSS community is completely voluntary and performed in the free time of its members) [7].

Although research examining usability in OSS has been published, there is no standardized procedure for determining how to adopt usability in OSS development. The first step in our research is to study how the OSS community uses usability techniques in their development projects. Castro's work proposes an integration framework that can incorporate most usability techniques in OSS developments [7]. It is important to clarify that this framework only proposes the general adaptations that must be made to the techniques. These adaptations depend on the requirements of the technique that cannot be satisfied by the way the OSS community works. Castro's research [7] was validated on only two OSS projects and for three usability techniques (user profiles, direct observation and post-test observation). Therefore Castro's proposal [7] requires further validation by adapting new usability techniques and participating in more OSS projects.

## III. RESEARCH METHOD

In our research, the collected data nature is qualitative (texts, images, documents) [14]. We used a case study as the qualitative research method to validate our research [15]. From a case study, we learn about the experiences of applying usability techniques adapted to OSS projects. This research method is used when the phenomenon under investigation (in this case, the adoption of an adapted usability technique) is studied within its real setting (in this case, an OSS project). OSS projects are the perfect setting for the case study reported here because OSS communities are, according to several authors [16], [17], unfamiliar with usability techniques. Small project teams in particular have little information about what techniques are at their disposal for improving usability [1], [18].

This case study aims to determine whether the VB usability technique can be adapted for use in requirements engineering activities in an OSS project. There are several OSS project repositories. One of the most popular is SourceForge.net. This repository classifies OSS projects by categories. Since this technique is related to requirements engineering for product

concept development, we looked at projects with a low level of coding (that is, projects where key features were still being added) that were not overly ambitious and were at the very early development stages (alpha version) in order to select a suitable OSS project in which to adopt the selected usability technique. Considering the above, we selected the HistoryCal OSS project. Thanks to the characteristics of this project, we can adopt the VB usability technique in a requirements activity (product concept development). Therefore, the benefits of applying the technique will have a bigger impact on the development process and software system usability.

In this research, we first identified the obstacles to applying the VB technique in the HistoryCal OSS project. It is important to mention that the usability techniques cannot be applied directly in the OSS developments so it is necessary to make adaptations so that they can be incorporated in OSS. We then decided how to deal with the obstacles. Finally, we proposed the adaptations necessary to adopt the VB technique in the case at hand. We created web artefacts to improve communication with OSS community members and efficiently synchronize the necessary activities to apply the VB usability technique. The most important part of the data analysis process was to build a web artefact, since HCI does not recommend any specified document or particular tool for gathering information during the application of usability technique. An accurate description of the explanations given by users participating in the OSS project is essential for interpreting and laying the general foundations of our research. The web artefact used to test the feasibility of the proposed technique was a blog, because this is a technique commonly used by the OSS community [19].

We used blogs in the VB technique to gather information and collect sketches related to the UI of the application under study. Thanks to this web artefact, we were able to set up a virtual meeting point with OSS users who are geographically distributed all over the world. Using such web artefacts, we aimed to record user opinions about the selected OSS project UI. We selected a blog because it is a web artefact providing better control of the opinions expressed by users during our research (for example, the researcher controls the information or sketches submitted by users for transmission to the developers). Researchers should control the user-blog interaction in order to ensure that they focus exclusively on graphical interface design only and do not get sidetracked by topics unrelated to the research.

#### IV. PROPOSED SOLUTION

In this section, we describe the VB usability technique applied in an OSS project. Firstly, we describe the case study design. Secondly, we specify the characteristics of the selected OSS project. Thirdly, we describe the selected usability technique as prescribed by HCI. We then introduce the adaptations made to the VB technique for application in an OSS project. Finally, we report the results of applying this usability technique.

##### A. Case Study Design

Case study is a non-experimental design type, since we do not randomly assign the subjects, nor do we control the groups. In addition, subjects are observed in their real context.

Depending on the paradigm in which the researchers are located, they have decided to use a case study with a positivist approach. A positivistic case study within qualitative methods is particularly suitable for research in information systems. The research based on the positivist case study is characterized by: not manipulating the experimental units, the results are basically obtained from the capacity of integration that has the researcher in the case, the study should focus on current situations, the phenomenon is studied in its real environment, only one or a few entities (individuals, group, community) are examined, the phenomenon of interest is not isolated from its context and there is no controlled observation that involves manipulation of the experimental unit [20]. These characteristics are present in our research.

We describe the case study following the guidelines set out by Runeson and Host [15]. According to these guidelines, we divide our research into two parts: an exploratory part and a descriptive part. We start by looking at what happens in a real-world scenario and then we describe what happens when we apply the adapted techniques to improve application usability [15]. The aim is, to determine whether, thanks to the proposed adaptations, the VB technique can be adopted in the OSS HistoryCal project. Our case study is based on the following research questions: How to incorporate the VB technique in a real OSS project?

##### B. Context and technique execution

We selected HistoryCal as the OSS project in which to adopt VB. HistoryCal is a calculator designed to work with different world calendar schemes, calculating date ranges and ages based on these calendars. This application is written in C++, and the reported number of downloads from its website is one per week. As shown in “Fig. 1”, HistoryCal’s graphical UI has a lot of room for improvement. At the same time, it is a project of special interest since popular office suites do not usually include a date converter. HistoryCal has only one member, who plays the role of both developer and administrator. We selected a small OSS project, like HistoryCal, in order to control all aspects with a view to conducting a larger-scale study.

The VB technique is a tool for generating new ideas about a particular topic or problem [10]. A group of three to four people is the ideal number for applying this technique [10]. In this case, we got five OSS users. This is a large enough number of users to be able to apply the VB technique. However, we expected a higher participation rate because, according to the related literature, OSS application users are very cooperative [21]. We discuss this issue in the discussion of results.

As prescribed by HCI, the aim of the VB technique is to generate ideas for interface design [8]. This technique cannot be applied directly, that is, as is prescribed by HCI, in OSS development projects because the OSS community has, as discussed above, characteristics that are uncommon in the HCI world. In addition, OSS users are characterized mainly because they collaborate voluntarily, that is to say, without remuneration. As a result, recruiting and retaining new members is a critical success factor for an OSS project [22].

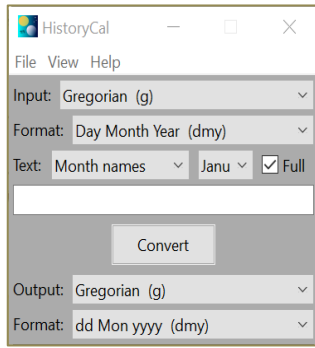


Figure 1. Original HistoryCal Interface

Even though usability techniques demand conditions that, as a rule, OSS projects cannot meet, the techniques can be adapted to bring them into line with the idiosyncrasy of the OSS world [1]–[4]. In the analysed literature [8]–[10], there are different procedures for applying the VB technique. Although they are all very similar, the procedure proposed by Wilson [9] is a good option for user-centred development processes because it gives a simple description of the steps for applying this technique. According to Wilson [9], this technique is composed of four steps. In the following, we describe these steps, the identified adverse conditions and adaptations proposed to tackle these adverse conditions. The first step is to arrange a meeting with the users to execute the VB technique. Alternative UI designs proposed by participants are explored and validated at small-scale meetings. This requires user participation at face-to-face meetings [9]. This is a condition that OSS projects cannot meet because of their particular features. To tackle this adverse condition, we suggest setting up a blog for users to post their comments on the UI. Due to certain developer and user language-related circumstances surrounding the selected OSS project, we created two blogs for this case study.

The second step is to gather the designs proposed by users (as mock-ups) based on which the UI is improved [9]. This step is problematic because the OSS community is distributed all over the world, and face-to-face meetings to build mock-ups are out of the question. Instead, we suggest using a blog where users can make comments and upload their designs. The third step is for users to evaluate the designs. To do this, they are required to attend face-to-face meetings [9]. Due again to the geographical location of users, OSS projects cannot meet this condition. On this ground, we suggest that the designs be posted on the blog at regular intervals for users to evaluate at their convenience. The designs were published an average of 2 per week during the month that the blog was active for this purpose. Finally, the fourth step is for a usability expert to design the selected user interface [9]. To overcome this obstacle (OSS project teams do not usually include a usability expert), we suggest that a HCI student under the supervision of mentor should stand in for the usability expert. It is important to mention that the mentor not only monitors the application of the technique, but also participates in the application of it. In addition, the HCI student's participation as a usability expert is competent thanks to the knowledge acquired during his studies.

Table I. summarizes the steps, identified adverse conditions and proposed adaptations for the VB technique [9]. There are mainly two adaptations. First, users participate online through web artefacts (e.g. blog). Second, we suggest that a HCI student or group of students under the supervision of a mentor replace the usability expert. In this case, the expert was replaced by a HCI student supervised by a mentor.

TABLE I. STEPS, ADVERSE CONDITIONS AND PROPOSED ADAPTATIONS

<i>Steps</i>	<i>Adverse conditions</i>	<i>Proposed adaptations</i>
1. Users meet to apply the technique	User participation at face-to-face meetings is required	Users participate posting their comments regarding the interface design in web artefacts (e.g., blog).
2. Users submit their design tips (in the form of mock-ups)	Users are located at different geographical sites.	Users provide feedback and attach their designs in their blog comments.
3. Users evaluate the designs	Users are not at the same geographical location.	The designs are published at regular intervals on the blog for evaluation by users.
4. An expert designs the final interface	A usability expert specializing in the technique is required	The expert is replaced by a HCI student or group of students supervised by a mentor.

## V. RESULTS AND DISCUSSION

In this section, we describe the results of the case study that were taken into account to adopt the adapted usability technique (VB).

### A. Results Analysis

As discussed earlier, we applied the adapted VB technique to the HistoryCal OSS project. We had difficulties recruiting real users to participate in technique application because the developer did not have a list of HistoryCal user emails. As we did not have access to this list of real users, we posted messages in the project forum and webpage inviting users to participate in the application of the VB technique. Finally, none of these real HistoryCal users replied to our invitation, and we decided to look at other user recruitment options (like social networks, email, LinkedIn and classmates). We then publicized the project using a mailing list with 150 LinkedIn contacts supplied by one of the researchers and 13 students of the HCI course taught as part of the Master in Information and Communication Technologies Research and Innovation at the School of Engineering, Autonomous University of Madrid. Finally, only five responses were received from all the users contacted (by email, LinkedIn and HCI students).

The VB technique was applied by creating two blogs on the WordPress platform: one in English<sup>1</sup> and the other in Spanish<sup>2</sup>. We built two versions of the blog because not all users are fluent in English and the developer does not understand Spanish. When the users (all native Spanish speakers) submitted their design tips or comments by email to one of the authors of this paper, they were translated to English and published on both blogs. Then, the developer examined and responded to/commented on the tips or comments. All this feedback was published on the blog. Additionally, the

<sup>1</sup> <https://historycalhci.wordpress.com>

<sup>2</sup> <https://historycalhcies.wordpress.com>



developer responses/comments were translated and published on the Spanish blog. There is no risk of getting a low quality translation of these comments because they were validated by a bilingual member of the research team.

In the following, we summarize some responses given by five users and posted on the blog: two users highlighted that they had trouble understanding how the application worked and that an example should be added, such as the input date in the selected format and another user suggested adding a calendar as an input data picker control. The interface developed as result of applying this technique is shown in “Fig. 2”. This interface was created based on the inputs of users and the feedback received by the application developer.

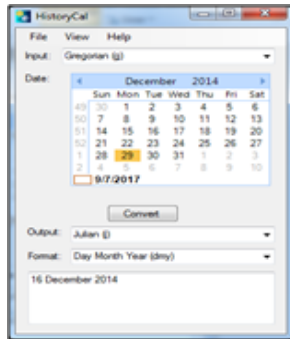


Figure 2. Interface design after applying the VB technique

The administrator was not very familiar with usability issues in the software development process. While he was acquainted with the concept of usability, his knowledge of usability techniques was limited. The project did not have a usability manager. Additionally, the administrator had not considered usability criteria in the design of the HistoryCal UI.

The results of applying the VB technique in the OSS HistoryCal project were sent by electronic mail to the project developer, who found our findings interesting. The project developer needed time to consider and build the findings into new versions of the HistoryCal project. On the other hand, several significant improvements were made to HistoryCal features following some of our recommendations. These improvements mainly consist of the inclusion of new calendars (i.e. Scottish, Julian, Gregorian). Another major improvement was the extension of the date format to comply with ISO standard: 8601.

### B. Discussion Of Results

In this section, we discuss and answer the research question raised in this research: How to incorporate the VB technique in a real OSS project?

Usability techniques were built for other types of software development, that is, they were not designed bearing in mind the characteristics of the OSS development process. On this ground, the techniques need to be adapted. These adaptations are based on the adverse conditions for technique application. Some of the adverse conditions can be overcome using web artefacts (i.e., wikis) with which the OSS community is familiar. Thus, community members will be acquainted with the many of these adaptations, encouraging to some extent the application of the usability techniques. Within the OSS

community the blog is the most frequently used web artefact to facilitate open discussion and maintain communication through a distributed community of members [19]. Indeed, the proposed solution to use a blog has been effective in adapting the VB technique in the HistoryCal project: Note that we tested the feasibility of adopting usability techniques in OSS projects adapted using web artefacts (i.e., a blog) rather than the effectiveness of blog use as such. With regard to our proposal of substituting a HCI student or group of students under the supervision of a mentor for the usability expert, the expert was replaced in this case by a HCI student supervised by a mentor.

By applying the VB technique to the HistoryCal project, we were able to confirm that it is very hard to get a representative set of real end users, as discussed by Raza et al. in their empirical study [21]. However, our experience of participating in large projects (e.g., OpenOffice Writer) has revealed that it is very difficult to recruit real end users to participate in the application of usability techniques in OSS projects generally. We had banked on the OSS project leader keeping a list of e-mails of representative real end users, but this was not the case. The fact that OSS developers are unacquainted with their user profile is a problem that has already been detected in other papers [17], [23], [24]. One contribution of our research is to provide empirical evidence that this really is the case. The OSS user participation rate during technique application was low. We believe that there are two main reasons for this. Firstly, none of the users were had basic knowledge in graphic design. The technique requires users with notions of graphic design because they will be required to give tips for improving the UI, for example, by creating sketches (albeit with simple graphics programs like Paint) of UI improvements. Secondly, an OSS usability project is unlikely to be able to provide any sort of incentive to encourage user participation. Despite all these problems, however, the adaptation of the VB technique was reliable for adoption in the HistoryCal project, according to the theory when using the VB technique it does not take many users to get a reliable result (three to four people is the ideal number) [10].

The application of this technique would be better if users were allowed to submit, in either textual or graphic format via a web artefact (like a blog or wiki where user inputs are collected), their comments and designs at their leisure, that is, without passing through the project administrator filter. Note that the user proposals and comments were reviewed by Nick Matthews, the developer. Again, some sort of incentive, like, for example, recognition for having collaborated in usability technique application, should be included to engage more users and increase participation in usability projects in the field of OSS. It is important to mention that these improvements were not implemented during the time of the study case and these suggestions were established at the end of the research. Therefore, we consider as future work to apply the VB technique in another OSS project with similar characteristics and implementing the aforementioned improvements. The main limitation of our research is the number of case studies (only one). Another threat is that the participants recruited to apply VB are not representative of end users.

## VI. CONCLUSIONS

The goal of this research was to evaluate the feasibility of adopting HCI usability techniques in OSS projects. Thanks to the technique adaptations, adoption was possible and we were able to account for some OSS development characteristics that pose an obstacle to the application of the techniques as per HCI recommendations. In particular, we adapted the VB usability technique for application in the HistoryCal OSS project.

The developer was receptive and interested in participating and receiving the results of the VB technique application to improve his software. After analysing and applying the usability technique, we found that there are adverse conditions that are an obstacle to technique application such as problems locating OSS users interested in applying the technique, geographical distribution and time differences and OSS community motivation.

This research sets out the procedure for applying the VB usability technique in OSS projects despite obstacles that are not exclusive to usability techniques: (i) lack of motivation causes a low user participation rate, (ii) OSS users are not usability experts, (iii) OSS users and developers are geographically distributed, (iv) there are language barriers between users and developers, and (v) OSS projects do not have HCI experts to help with the application of usability techniques. The results of this research are applicable to small projects similar to the one for which we volunteered (i.e. HistoryCal). This is because large OSS projects have different characteristics: (i) they are very active and popular projects with a large user base, (ii) bugs are reported in multiple sources, (iii) common problems are accessible in online infrastructures (e.g. email lists, forums, etc.), and (iv) their work practices and tests are documented. Being a qualitative study with these specific characteristics, an exact replication for other OSS project types would not be possible.

We believe that, in order to improve the integration of usability techniques in OSS projects, the OSS community has to start attaching importance to and raising awareness about the repercussions that the issues addressed by the HCI field have on software development. Additionally, as HCI techniques need to be adapted to overcome the adverse conditions for adoption in OSS development projects, the OSS community also has to broaden its view of software development in order to consider usability and not focus exclusively on feature development. In the future, we aim to conduct further case studies to adapt and apply other usability techniques in OSS projects. We will analyse other web artefacts that can be adapted to improve communication in OSS communities (for example, social networks) and gradually raise the awareness of OSS developers about the benefits of applying HCI usability techniques.

## ACKNOWLEDGMENT

This research was funded by the SENESCYT-Ecuador, and Quevedo State Technical University. Also this research was funded by the Spanish Ministry of Education, Culture and Sports FLEXOR (TIN2014-52129-R) and TIN2014-60490-P

projects and the eMadrid-CM project (S2013/ICE-2715). Finally, this research received funding from the "DIUDA 22316 Project" of the University of Atacama.

## REFERENCES

- [1] G. Çetin and M. Gokturk, "A Measurement Based Framework for Assessment of Usability-Centricness of OSS Projects," in *SITIS'08*, 2008, pp. 585–592.
- [2] G. Çetin, D. Verzulli, and S. Frings, "An Analysis of Involvement of HCI Experts in Distributed Software Development: Practical Issues," in *OCSC'07*, vol. 4564, Springer., 2007, pp. 32–40.
- [3] H. Hedberg, N. Iivari, M. Rajanen, and L. Harjumaa, "Assuring Quality and Usability in Open Source Software Development," in *FLOSS'07*, 2007, pp. 1–5.
- [4] M. Terry, M. Kay, and B. Lafreniere, "Perceptions and Practices of Usability in the FOSS Community," in *International Conference on Human Factors in Computing Systems CHI 2010*, 2010, pp. 999–1008.
- [5] D. M. Nichols and M. B. Twidale, "The Usability of Open Source Software," *First Monday*, vol. 8, no. 1, p. 21, Jan. 2003.
- [6] S. Osiński and D. Weiss, "Introducing usability practices to OSS: The insiders' experience," *IFIP Int. Fed. Inf. Process.*, vol. 234, no. d, pp. 313–318, 2007.
- [7] J. W. Castro, "Incorporating Usability in the Open Source Software Development Process," Doctoral thesis. Departamento de Ingeniería Informática. Universidad Autónoma de Madrid, 2014.
- [8] X. Ferré, "Marco de Integración de la Usabilidad en el Proceso de Desarrollo Software.," Doctoral thesis. Facultad de Informática. Universidad Politécnica de Madrid, 2005.
- [9] C. Wilson, *Brainstorming and Beyond: A User-Centered Design Method*. Morgan Kaufmann., 2013.
- [10] A. Osborn, *Applied Imagination: Principles and Procedures of Creative Problem-Solving*, 3rd Revise. Charles Scribner's Son, 1963.
- [11] J. Preece, Y. Rogers, H. Sharp, D. Benyon, S. Holland, and T. Carey, *Human-Computer Interaction*, 1st Ed. Addison-Wesley Pub. Co, 1994.
- [12] A. Ternauciuc and R. Vasiliu, "Testing usability in Moodle: When and How to do it," in *SISY 2015*, pp. 263–268.
- [13] H. Assa, S. Chiasson, and R. Biddle, "Cesar: Visual Representation of Source Code Vulnerabilities," *2016 IEEE Symp. Vis. Cyber Secur.*, pp. 1–8, 2016.
- [14] A. A. Khan and J. Keung, "Systematic review of success factors and barriers for software process improvement in global software development," *IET Softw.*, no. April, pp. 1–11, 2016.
- [15] P. Runeson, M. Host, A. Rainer, and B. Regnell, *Case Study Research in SE: Guidelines and Examples*. John Wiley & Sons., 2012.
- [16] J. Blitzer, W. Schrettl, and P. J. H. Schröder, "Intrinsic Motivation versus Signaling in Open Source Software Development," *J. Comp. Econ.*, vol. 35, no. 1, pp. 160–169, 2007.
- [17] D. M. Nichols and M. B. Twidale, "Usability Processes in Open Source Projects," *Softw. Process Improv. Pract.*, vol. 11, no. 2, pp. 149–162, 2006.
- [18] L. Nielsen and M. Bødker, "To Do or Not to Do: Usability in Open Source Development.," *Interfaces*, vol. 71, pp. 10–11, 2007.
- [19] D. Pagano and W. Maalej, "How do open source communities blog?," *Empir. Softw. Eng.*, vol. 18, no. 6, pp. 1090–1124, 2013.
- [20] Line Dubé and Guy Paré, "Rigor in Information Systems Positivist Case Research: Current Practices, Trends, and Recommendations," *MIS Q.*, vol. 27, no. 4, pp. 597–636, 2003.
- [21] A. Raza, L. F. Capretz, and F. Ahmed, "An Open Source Usability Maturity Model (OS-UMM).," *J. Comput. Hum. Behav.*, vol. 28, no. 4, pp. 1109–1121, 2012.
- [22] K. Crowston, K. Wei, J. Howison, and A. Wiggins, "Free/Libre open-source software development: What We Know and What We Do Not Know," *ACM Comput. Surv.*, vol. 44, no. 2, pp. 1–35, 2012.
- [23] C. Benson, M. Müller-Prove, and J. Mzourek, "Professional Usability in Open Source Projects: GNOME, OpenOffice.org, NetBeans," in *CHI Extended Abstract on Human Factors in Computing System - CHI EA'04*, 2004, pp. 1083–1084.
- [24] M. Müller-Prove, "User Experience for OpenOffice.org.," *Interfaces (Providence)*, vol. 71, no. Summer, pp. 8–9, 2007.

# XPA: An Open Source IDE for XACML Policies

Roshan Shrestha

Department of Computer Science  
Boise State University  
Boise, ID 83725, USA  
roshanshrestha@boisestate.edu

Shuai Peng

Department of Computer Science  
Boise State University  
Boise, ID 83725, USA  
shuaipeng@boisestate.edu

Turner Lehmbecker

Department of Computer Science  
Eastern Washington University  
Cheney, WA, USA  
edmfrosty@gmail.com

Dianxiang Xu

Department of Computer Science  
Boise State University  
Boise, ID 83725, USA  
dianxiangxu@boisestate.edu

**Abstract**—This paper presents XPA (XACML Policy Analyzer), an open source IDE (Integrated Development Environment) for testing, debugging, and mutating XACML 3.0 policies. XACML is an OASIS standard for specifying attribute-based access control policies. XPA provides a variety of new techniques for generating test cases from policies, localizing bugs in faulty policies, and repairing faulty policy elements. XPA has been applied to numerous XACML policies from the literature and real-world applications. These policies have been used to quantitatively evaluate the effectiveness of various testing and debugging methods. For system developers and administrators, XPA is a practical IDE for developing dependable XACML policies. For access control researchers, XPA offers a versatile toolkit for studying and evaluating new testing, debugging, and verification techniques.

**Keywords**—access control, XACML, testing, fault localization, debugging

## I. INTRODUCTION

Attribute-Based Access Control (ABAC) is a new generation of access control techniques. It makes authorization decisions based on attributes of users, resources, actions, and environments [1]. Due to its fine granularity and high flexibility, ABAC is playing an increasing role in business and federal security domains. XACML (eXtensible Access Control Markup Language) is an OASIS standard for specifying ABAC policies in the XML format [2]. It has been integrated in major identity management products, such as Oracle Identity Manager and WSO2 Identity Server. Although these products allow user to edit and query XACML policies, there is a lack of tool support for policy testing, debugging, and evaluation.

The inherent complexity of real-world ABAC policies and the expressiveness of the XACML language indicate the likely existence of access control defects and the difficulty in finding them. The access control defects may result from omission or misunderstanding of access control requirements, unexpected interactions between security policy and business logic, and coding errors. These defects need to be uncovered and fixed before the system is deployed; otherwise they may lead to unauthorized access or denial of service. For quality assurance purposes, we argue that, similar to system and software development, policy development should follow a rigorous engineering process, including requirements analysis, design (e.g., decomposition and modularization), coding (e.g., in the XACML language), validation (e.g., testing and debugging), deployment and maintenance. Thus, an integrated development environment (IDE) is needed to provide computer-aided support for various activities in this engineering process.

This paper presents XPA (XACML Policy Analyzer), an evolving IDE for the development and implementation of dependable XACML policies. It consists of a variety of tools for editing, compiling, testing, debugging, and mutating XACML policies. The main features are: (1) coverage-based test generation using a constraint solver for XACML policies, (2) mutation-based test generation using a constraint solver for XACML policies, (3) coverage-based fault localization of XACML policies, and (4) mutation-based repair of XACML policies. The underlying technical approach of each feature implies substantial research effort and its elaboration requires a separate paper. The fault localization and repair methods for debugging XACML policies appeared in our previous work [3] [4], but their implementations have been improved for efficiency and user-friendliness. The policy mutator in XPA is currently the only one that supports XACML 3.0 and second-order mutation (i.e., application of two mutation operators). Other mutation tools for XACML [5][6] can only apply one mutation operator to XACML 1.0 and 2.0 policies.

The remainder of this paper is organized as follows: Section II gives a brief introduction to XACML policies. Section III presents the architecture of XPA, Section IV describes the mutation tool for XACML 3.0 policies. Sections V and VI present coverage-based and mutation-based test generators, respectively. Section VII introduces fault localizer and policy repairer. Section VIII summarizes the evaluations of XPA. Section IX reviews and compares related work. Section X concludes this paper.

## II. XACML POLICIES

The first class entities in XACML are policy and policy set. A policy set consists of a policy set target, a policy-combining algorithm identifier, a list of policies or policy sets, an obligation expression, and an advice expression. Policy set target, obligation expression, and advice expression are optional. An obligation expression describes the string attached to the access privilege, whereas an advice expression describes an optional suggestion on the access. A policy comprises a policy target, a rule-combining algorithm identifier, a list of rules, an obligation expression, and an advice expression. A rule consists of a target, a condition, an effect (permit or deny), an obligation expression, and an advice expression. The rule target specifies the set of requests to which the rule is intended to apply. The rule condition refines the applicability of the rule established by the rule target. The target of a rule, policy, or policy set is a conjunctive sequence of AnyOf clauses. Each AnyOf clause is a disjunctive sequence of AllOf clauses, and each AllOf clause is a conjunctive sequence of match

---

This work was supported in part by US National Science Foundation (NSF) under grants CNS 1618229 and CNS 1461133.

predicates. A match predicate compares attribute values in an access request with the embedded attributes. Logical expressions for match predicates and rule conditions can apply a great variety of predefined functions and data types (such as string, Boolean, integer, double, time, and dates) to attributes. XACML provides four pre-defined categories of attributes: subject, resource, action, and environment. It also allows user to introduce additional attribute categories.

When an access request is fed to an XACML engine that is running a policy set or policy, the engine will return an access decision (permit, deny, not applicable, or indeterminate) per the policy set or policy. The decision may be attached with obligation or advice, depending on the policy or policy set. An access request consists of a list of attribute names, types, and values. In this paper, it is also called test input, specified in a text file. A complete test case is composed of both test input and expected access decision (i.e., oracle value). The oracle value for a test input is usually determined by the access control requirements of the system under development. When a policy set or policy is known to be correct (e.g., for experiment purposes), the actual response of the policy set or policy can be recorded and then used as the oracle value of the corresponding test input. In an evolving policy development process, the actual access decisions of test inputs from earlier policy versions can be recorded and then used as the oracle values of corresponding test inputs for testing the current or future versions if their correctness has been confirmed before. Given a test case for a policy set or policy, the actual response returned by the XACML engine depends on the evaluation of all policy elements. Consider a typical policy set with a list of policies, where each policy is composed of a list of rules. The final access decision per the policy set depends on the evaluation results of the policy set target, access decisions of individual policies within the policy set, and the policy combining algorithm. The access decision of each individual policy depends on the evaluation results of the policy target, access decisions of individual rules in the policy, and the rule combining algorithm. XACML3.0 provides 11 rule combining algorithms and 12 policy combining algorithms. The most commonly used combining algorithms are Deny-overrides, Permit-overrides, First-applicable, Deny-unless-permit, and Permit-unless-deny.

### III. THE ARCHITECTURE OF XPA

Figure 1 shows the architecture of XPA. The main components are: editor, test runner, fault localizer, policy repairer, policy mutator, mutation-based test generators, and coverage-based test generators. It is implemented in Java and AspectJ (an aspect-oriented extension to Java). The editor is adapted from the open source project UMU-XACML-Editor [7], which was originally developed for XACML 1.0 and 2.0. The XACML engine is Balana [8], the only open source implementation for XACML 3.0 when we started this project.

The test runner feeds a test suite to the XACML engine running a policy set or policy and reports the pass/fail result of each test. For a test case without an oracle value (expected response), the actual response is recorded. For a test case with an oracle value, the test runner also compares the oracle value with the actual response and makes a verdict of pass or fail.

The failure of a test case indicates that the policy or policy set under test has one or more faults if the test input and the oracle value are both correct.

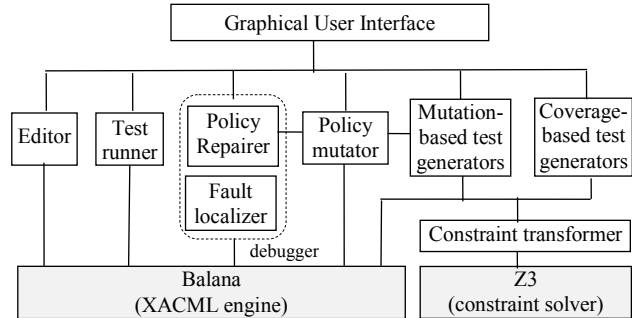


Fig. 1. Architecture of XPA

When there is a test failure, the fault localizer can be used to pinpoint the possible locations of faults (e.g., policy elements in the policy or policy set under test). It ranks all policy elements in the descending order of their suspicion scores calculated from the execution of the entire test suite. The user can then examine the top-ranked elements to determine whether they are faulty and how to fix them. Because Balana does not keep track of test execution information, we use an aspect-oriented instrumentation technique in AspectJ to monitor the evaluation result of each policy element when each test case is executed. This technique does not need to modify the source code of Balana.

The policy repairer takes a step further, aiming to repair a faulty policy automatically. It attempts to make a series of changes to the faulty policy, i.e., mutate the faulty policy, so as to make all test cases pass. The repair attempt may or may not be successful, depending on the faults. Note that automatic repair is a hard problem. To the best of our knowledge, our work is the first effort toward automatic repair of XACML policies and policy sets. The current repairer can fix a fault policy with no more than two simple faults. A simple fault is one that can be corrected by one mutation operation.

The policy mutator is a program that generates mutants of a given policy or policy set. Each mutant is a variation of the original policy or policy set. A first-order mutant is obtained by applying one mutation operator to make one change, whereas a second-order mutant is created by applying two mutation operators to make two changes. As the mutants of a correct policy contain different types of faults, they are commonly used to evaluate the effectiveness of a testing method, i.e., how many faults can be detected.

XPA also exploits policy mutation for test generation purposes. Given an original policy and its mutant, a mutation-based test generator produces an access request such that the two policies yield different responses. To do so, it first collects the constraint on attributes by comparing the two policy versions and then feeds the constraint to Z3 to find attribute values to satisfy the constraint, and converts the result into an access request. For a set of policy mutants, the mutation-based test generators can produce an optimal test suite that reveal all faulty mutants. This test suite can be used to test a policy or

policy set without knowing whether the policy or policy set is faulty. It can also be used to measure other testing methods. Z3 is an SMT (Satisfiability Modulo Theories) Solver from Microsoft Research [9]. It is worth pointing out that, although policy mutation is commonly used for evaluating testing methods in the literature, mutation-based test generation and mutation-based policy repair in our work are new.

Coverage-based test generators are a set of programs that generate a test suite from a given policy or policy set according to a chosen coverage criterion. The main coverage criteria are rule coverage, rule pair coverage, permit/deny rule pair coverage, decision coverage, MC/DC (modified-condition and decision coverage), non-error decision coverage, and non-error MC/DC. Each coverage-based test generator first collects the constraints on attributes according to the chosen coverage criterion, feeds each constraint to Z3, and converts the result into an access request. The coverage-based test generators focus on the extent to which the policy elements are exercised by tests, whereas the mutation-based test generators aim to produce tests that can reveal hypothesized faults. Both are useful for quality assurance of XACML policies.

#### IV. POLICY MUTATOR

The policy mutator creates mutants of a policy set or policy by applying mutation operators to the policy set or policy. Mutation operators are defined with respect to a fault model, which represents a comprehensive set of fault types in XACML. Table I shows the fault model (i.e., column 1) and mutation operators for each fault type. Application of one mutation operator may result in a number of mutants. For example, the rule combining algorithm of a policy can be changed to any of the other rule combining algorithms.

TABLE I. FAULT MODEL AND MUTATION OPERATORS

Fault type	Mutation operator	
	Name	Mutation
Incorrect policy/ policy set target	PTT	set Policy/set Target True
	PTF	set Policy/set Target False
Incorrect rule/policy combining algorithm	CRC	Change Rule/Policy Combining algorithm
Incorrect rule effect	CRE	Change Rule Effect
Incorrect rule target	RTT	set Rule Target True
	RTF	set Rule Target False
Incorrect rule condition	RCT	set Rule Condition True
	RCF	set Rule Condition False
	ANF	Add Not Function in condition
	RNF	Remove Not Function in condition
Incorrect rule ordering	FPR	First Permit Rules
	FDR	First Deny Rules
Missing rule	RER	REmove a Rule
Missing target element	RPTE	Remove Parallel Target Element

Mutation operators in Table I are named with respect to correct policy sets and policies. Mutants of a correct policy set or policy may or may not contain faults. It is possible that a mutant is functionally equivalent to its original version, i.e., they always yield the same access decision for any access

request. Mutants of a correct policy set or policy are commonly used for evaluating the fault detection capability of a testing method in term of mutation score. A mutant is said to be killed if a failure is reported by any test case produced by the testing method. A mutant that is not killed may be equivalent to the original policy. Given a test suite produced by a testing method, its mutation score or mutant-killing ratio is as follows:

$$\frac{\text{number of killed mutants}}{\text{total number of mutants} - \text{number of equivalent mutants}}$$

Note that mutation operators can be applied to a policy set or policy no matter whether the policy set or policy is known to be correct or faulty. In particular, XPA applies mutation to test generation (Section VI) and policy repair (Section VII). In these cases, the fault types in Table I do not represent the meanings of mutation operators.

#### V. COVERAGE-BASED TEST GENERATORS

The coverage-based test generators produce access requests from a given policy set or policy to satisfy a chosen coverage criterion. As policies are special cases of policy sets, we describe the coverage criteria with respect to policy sets.

**Rule coverage:** A test suite for a policy set is said to satisfy rule coverage of the policy set if, for each rule in each policy of the policy set, there is at least one test in the test suite that evaluates the rule to its specified effect (permit or deny).

**Decision coverage:** A test suite for a policy set is said to satisfy decision coverage of the policy set if the test suite covers all three decisions (true, false, error) of each decision expression, including the policy set target, the target of each policy, the target and condition of each rule in each policy.

**Non-error decision coverage:** A test suite for a policy set is said to satisfy non-error decision coverage of the policy set if the test suite covers all non-error decisions (true and false) of each decision expression, including the policy set target, the policy target of each policy, the rule target and condition of each rule in each policy.

**MC/DC:** A test suite for a policy set is said to satisfy MC/DC of the policy set if the test suite satisfies MC/DC and covers the error condition of each decision expression, including the policy set target, the policy target of each policy, the rule target and condition of each rule in each policy.

**Non-error MC/DC:** A test suite for a policy set is said to satisfy MC/DC of the policy set if the test suite satisfies MC/DC of each decision expression, including the policy set target, the policy target of each policy, the rule target and condition of each rule in each policy.

**Rule pair coverage:** A test suite for a policy set is said to satisfy rule pair coverage of the policy set if, for each pair of rules within each policy, the test suite has a test to make both rules evaluate to their specified effects.

The above coverage criteria can also be used to measure the coverage adequacy of a given test suite. Such a test suite may be produced by other testing methods when a policy is developed or represent actual access requests in operation.

Generally speaking, test suites of different coverage criteria have different levels of fault detection capabilities. The measurement of coverage adequacy provides important guidelines for the development of access control tests.

## VI. MUTATION-BASED TEST GENERATORS

Given a policy set or policy whose correctness is unknown, mutation-based test generators create access requests by comparing the policy set or policy with each of its mutants (i.e., a hypothesized fault). The mutants are obtained by applying the mutation operators in Table I. A mutation-based test generator with respect to a mutation operator tries to generate one access request for each mutant obtained by the mutation operator. For such an access request, the original version and the mutant will respond with different access decisions. Assuming that one version is correct and the other version is faulty, the idea of mutation-based test generation relies on the following fault detection conditions: (1) **Reachability condition**: the access request must reach the mutated policy element, such as rule target, rule condition, rule effect, policy target, policy set target, and rule/policy combining algorithm. (2) **Necessity condition**: the access request must make the mutated element and the corresponding element in the original version evaluate to different intermediate results; (3) **Propagation condition**: the access request must make the mutant and the original produce different responses. Propagation condition largely depends on the rule and policy combining algorithms.

By comparing the two policy versions, the mutation-based test generator derives a constraint that is composed of all three conditions. Then it feeds the constraint to Z3. If the constraint is solved, the solution is converted into an access request; otherwise the two policy versions are considered to be equivalent, assuming Z3 is sound and complete.

The key challenge of mutation-based test generation is the formalization of reachability condition, necessity condition, and propagation condition for each kind of mutants. The idea originated from fault-based testing or constraint-based testing in the software testing community. However, practical mutation-based test generators for software remain to be seen unless for toy examples – it is difficult, if not impossible, to formulate the fault detection conditions because of the inherent complexity of software. Due to the special structure of XACML policy sets and policies, we have been able to automatically derive complete fault detection conditions of all mutants. The details will be described in a separate paper.

## VII. AUTOMATED DEBUGGER

The automated debugger consists of the fault localizer and the mutation-based policy repairer. Fault localizer aims to identify which element of a policy set or policy is likely faulty if there is a failure when it is tested with a test suite. The basic idea is to build a correlation between the evaluation result (firing or not) of each policy element and the test verdict (pass or fail) for each test case. The correlation data is then used to rank all policy elements with a certain scoring method. A policy element with a high suspicion score has a high probability of having fault(s). XPA has implemented 14

scoring methods selected from the best-performing spectrum-based methods for software fault localization [10].

The policy repairer takes a step further to modify the policy set or policy so that no test in the test suite will fail. According to the suspicion rankings from the fault localizer, the repairer starts with the most suspicious policy element, mutates it to create a new policy set or policy, and runs the new policy set or policy to check if all tests pass. If there is no failure, the repair is successful; otherwise the repairer will try another mutation or another suspicious element. Because a policy set or policy may have a number of faults, the repairer exploits the notion of plausible fix. A plausible fix does not make all tests pass. Instead, it makes the debugging progressive, indicated by a decreased number of failed tests. The repairer allows user to set up the depth of mutation for repair, a scoring method for sorting suspicious elements, and select some or all of the mutation operators. If the repair attempt is successful, XPA presents the relevant policy elements of both original and repaired versions.

## VIII. EVALUATION AND APPLICATION

We have applied a number of XACML policies to XPA, as listed in Table II. All of them are available at the project website. Three policies, *continue*, *fedora*, and *itrust*, were obtained from the literature. They were originally coded in 1.0 or 2.0. We upgraded them to 3.0 without changing their semantics. We also created three variations of *itrust* (*itrust5*, *itrust10*, and *itrust20*) for studying scalability of testing and debugging methods. *itrustX* has *X* times as many rules as *itrust*. The new rules are created by replicating original rules with new attribute values. *HL7* is a real world policy set provided by an XACML developer. The system that uses the *HL7* policy set is not available, though. GPMS (Grant Proposal Management System) is an open source Java project that we have developed as an exemplar application of XACML. The motivation behind GPMS was that there is no real-world XACML3.0 application whose policy files and application source code are publicly available. GPMS is a web-based application for an academic institution to manage the internal workflow for grant submissions. It uses XACML to implement a fine-grained access control of the workflow.

TABLE II. SAMPLE XACML3.0 POLICIES

Policy	# of rules	# of lines of the XACML file
continue	15	229
fedora <sup>1</sup>	12	227
itrust <sup>2</sup>	64	1,283
itrust5	320	6,403
itrust10	640	12,803
itrust20	1,280	25,603
HL7	19	809
GPMS policy	97	7,678

**Evaluation of the coverage-based test generators**: Six policies (*continue*, *fedora*, *itrust*, *itrust5*, *itrust10*, and *itrust20*)

<sup>1</sup> <http://www.fedora.info>

<sup>2</sup> <http://agile.csc.ncsu.edu/iTrust/wiki/doku.php?id=start>

have been used to evaluate all coverage-based test generators. As they are considered to be the correct version, the oracle value of each test input is the actual response from the original policy. The fault detection capability of each test generator is assessed through mutation testing, where both first-order and second-order mutants were generated by the policy mutator. The results show that the MC/DC test suite has the highest mutation score, whereas the rule coverage test suite was only able to kill about 50% of the mutants. All test generators have acceptable time performance for all policies.

**Evaluation of the mutation-based test generators:** All policies in Table II have been applied to the mutation-based test generators. They are able to generate a test input for every non-equivalent first-order mutants.

**Evaluation of the fault localizer:** All policies except *itrust20* and the GPMS policy in Table II have been applied to the fault localizer. The first-order and second-order mutants of each policy are used as inputs to the fault localizer. The experiments show that the 14 scoring methods have varying accuracy. *Naish2* and *CBI-Inc* can accurately localize the faults regardless of the policy size. The actual faulty policy element is usually among a few top candidates that are suggested by the *Naish2* and *CBI-Inc* methods.

**Evaluation of the policy repairer:** All policies except *itrust20* and the GPMS policy in Table II have been applied to the policy repair. Both first-order and second-order mutants of each policy are used as inputs. All scoring methods were able to repair them. This indicates that the mutation operators for policy mutation and the mutation operators for policy repair are reversible. The *Naish2* and *CBI-Inc* methods have the best time performance to locate the faulty elements.

**Application to GPMS:** XPA was used to test the GPMS policy in the development process of GPMS. The mutants of the GPMS policy is currently being used to evaluate the fault detection capability of a model-based test method for GPMS.

## IX. RELATED WORK

Several methods have been proposed to generate test inputs for XACML policies: Cirg [11] generates access requests from counterexamples produced by the change-impact analysis of two synthesized versions. The difference of the two versions of a policy targets a test coverage goal (e.g., rule, or condition). Because access requests are encoded in XML, they must conform to the XML Context Schema. Bertolino et al., have developed different test generation algorithms by considering the structures of the Context Schema, such as Preliminary XPT and Incremental XPT [12]. Li et al. [8] used symbolic execution technique to generate access requests by converting the XACML policy under test into semantically equivalent C Code Representation (CCR) and symbolically executing CCR to create test inputs and translating the test inputs to access requests. The coverage-based test generators in XPA are different from the above work except for the rule coverage. In addition to the new coverage criteria, XPA generates access requests for exercising error conditions. Policy mutation has

been used to evaluate the above testing methods, but limited to 1.0 and 2.0 [5][6]. XPA also uses policy mutation for test generation and policy repair.

## X. CONCLUSIONS

We have presented a comprehensive toolkit, XPA, for editing, testing, debugging, and mutating XACML policies. It also provides an infrastructure for experimentation with new testing and debugging methods. For example, when mutation is used to evaluate a new testing method against a policy, XPA can apply the test suite to all mutants of all or selected mutation operators and produce a summary of test execution results.

Our future work will focus on tool support for access control requirements analysis and policy maintenance in the policy engineering process. We plan to develop a computer-aided approach for transforming access control requirements specification in a natural language (e.g., English) into XACML policies. We will also implement various refactoring methods to facilitate changes of XACML policies.

## REFERENCES

- [1] V. Hu, D. Ferraiolo, R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller, K. Scarfone, "Guide to attribute based access control (ABAC) definition and considerations," NIST Special Publication 800-162, October 2013.
- [2] OASIS, "eXtensible Access Control Markup Language (XACML) Version 3.0," <http://www.oasisopen.org/committees/xacml/>, Jan. 2013.
- [3] D. Xu, Z. Wang, S. Peng, N. Shen, "Automated fault localization of XACML policies," Proc. of the 21st ACM Symp. on Access Control Models and Technologies (SACMAT'16), pp. 137-147, June 2016.
- [4] D. Xu and S. Peng, "Towards automatic repair of access control policies," Proc. of the 14th IEEE Conference on Privacy, Security and Trust (PST'16), pp. 485-492, Auckland, New Zealand, December 2016.
- [5] E. Martin, and T. Xie, "A fault model and mutation testing of access control policies," Proc. of WWW'07, pp. 667-676, May 2007.
- [6] A. Bertolino, S. Daoudagh, F. Lonetti, E. Marchetti, "XACMUT: XACML 2.0 mutants generator," Proc. of 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops, pp. 28-33, 2013.
- [7] P. G. Morcillo, A. J. Lázaro, G. Tormo UMU-XACML-Editor. <http://umu-xacmleditor.sourceforge.net/>
- [8] WSO2. "Balana: An open source XACML 3.0 implementation." <http://xacmlinfo.org/2012/08/16/balana-the-open-source-xacml-3-0-implementation/>
- [9] L. de Moura and N. Björner, "Z3: An efficient SMT solver," Proc. of the 14th International Conference Tools and Algorithms for the Construction and Analysis of Systems (TACAS'08), LNCS volume 4963. 2008, Springer.
- [10] X. Xie, T. Y. Chen, F. Kuo, and B. Xu. "A theoretical analysis of the risk evaluation formulas for spectrum-based fault localization," ACM Trans. on Software Engineering and Methodology (TOSEM), 22(4):31.
- [11] E. Martin, and T.Xie. "Automated test generation for access control policies via change-impact analysis." Proceedings of the Third International Workshop on Software Engineering for Secure Systems. IEEE Computer Society, 2007, pp.5-11.
- [12] A. Bertolino, S. Daoudagh, F. Lonetti, and E.marchetti. "The X-CREATE Framework-A Comparison of XACML Policy Testing Strategies." Proc. of the 8th International Conference on Web Information Systems and Technologies (WEBIST). pp.155-160.

# Automatic Detection of Public Development Projects in Large Open Source Ecosystems: An Exploratory Study on GitHub

Can Cheng, Bing Li, Zengyang Li\*, Peng Liang  
School of Computer Science  
Wuhan University  
Wuhan, China

**Abstract**— Hosting over 10 million of software projects, GitHub is one of the most important data sources to study behavior of developers and software projects. However, with the increase of the size of open source datasets, the potential threats to mining these datasets have also grown. As the dataset grows, it becomes gradually unrealistic for human to confirm quality of all samples. Some studies have investigated this problem and provided solutions to avoid threats in sample selection, but some of these solutions (e.g., finding development projects) require human intervention. When the amount of data to be processed increases, these semi-automatic solutions become less useful since the effort in need for human intervention is far beyond affordable. To solve this problem, we investigated the GHTorrent dataset and proposed a method to detect public development projects. The results show that our method can effectively improve the sample selection process in two ways: (1) We provide a simple model to automatically select samples (with 0.827 precision and 0.947 recall); (2) We also offer a complex model to help researchers carefully screen samples (with 63.2% less effort than manually confirming all samples, and can achieve 0.926 precision and 0.959 recall).

**Keywords**- *open source ecosystem; project sample selection; automated method; public development project;*

## I. INTRODUCTION

In recent years, the GitHub ecosystem has witnessed an increasing popularity, and it has attracted more than one hundred studies focused on it [1]. With more than 10 million of software projects hosted on this ecosystem, it is hard to select appropriate samples (i.e., projects) when conducting large scale case studies. Early studies often use manual selection methods to select samples. But as large datasets appear [2], researchers often face a dilemma that they want to use a large dataset to verify the generality of their results and at the same time they cannot confirm whether these projects meet their research goals. Thus, it is important to find a way to automate the sample selection process.

Most research on GitHub needs to satisfy an implicit hypothesis: their sampled projects under investigation are public development projects, which means that these projects

should be open to public and the content of these projects should be about development. For example, when studying communities and teams of projects in GitHub, their samples must be public development projects. Because many projects hosted on GitHub are not for software development (e.g., blogs, translation and student homework) and are private [3]. When the sample size is small, it is doable to read the descriptions and readme files manually to select appropriate samples. However, as the dataset becomes larger, this manual method becomes inefficient. Hence, in this study, our goal is to automatically detect public development projects.

We developed a model in this work to automatically detect public development projects based on the J48 decision tree algorithm [4]. We verified our model on a dataset of 6,715 GitHub projects labeled by master and PhD students on software engineering. Our model performs well in classifying public development projects. The main contributions of this work are:

- We identified a set of words and phrases (e.g., mirror, personal) that can reflect projects' properties.
- We fitted a simple decision tree that can classify public development projects with a precision of 0.839 and a recall of 0.950, and this model can help researchers effectively select appropriate samples.
- For those studies that have strict requirements on the dataset, we provide a complex decision tree, with 63.2% less effort than manually confirming all samples, and it can obtain a classification with a precision of 0.926 and a recall of 0.959.

In the rest of this paper, related work is discussed in Section II. Design of this study is described in Section III. The results are elaborated in Section IV. Threats to validity of the results are presented in Section V, and this work is concluded in Section VI.

## II. RELATED WORK

### A. Problems in Studying Open Source Ecosystem

Nowadays, more and more research studied software ecosystems [5] (most objects of software ecosystem research are open source software (OSS) ecosystems), the potential reasons for this phenomenon is that open source ecosystems

---

\*Corresponding author. E-mail: [zengyangli@whu.edu.cn](mailto:zengyangli@whu.edu.cn).  
DOI reference number: 10.18293/SEKE2018-085



provide publicly available historical datasets which researchers can benefit from.

However, there are some problems when studying the historical data of open source ecosystems. Howison and Crowston found that projects hosted on SourceForge were often abandoned and their information was often missing since some project data are hosted outside SourceForge [6]. Weiss argued that it is not necessary to consider all SourceForge data because of the fickleness of some projects [7]. Rainer and Gale conducted in-depth analysis on the quality of SourceForge data [8]. They noted that only 1% of SourceForge projects were actually active, and suggested that researchers should be careful when using project samples. Kalliamvakou *et al.* investigated on the defects of GitHub datasets [3, 9]. They detected 13 perils and gave strategies to avoid these perils.

Although researchers have already been aware of the problem, some solutions to the problem still remain in the stage of manual verification. When facing a rapidly-increasing amount of data on OSS projects, these methods are not effective any more.

### B. Studies on GitHub Datasets

As our dataset was retrieved from the GitHub ecosystem [10], we first investigate the datasets which can be potentially used to study GitHub. Cosentino *et al.* conducted a systematic mapping study on GitHub [1] and concluded that currently there are six ways to get GitHub data: (1) GHTorrent [2], (2) GitHub Archive, (3) GitHub API, (4) others (e.g., BOA [11]), (5) manual approach, and (6) a mixture of them. It is pointed out by Kalliamvakou *et al.* [3] that GitHub Archive started data collection in 2011, and is an incomplete mirror to GitHub. GHTorrent, in comparison, has retrieved the complete history of GitHub. Moreover, GHTorrent can be extended by GitHub API which is most frequently used, hence we collected studies based on GHTorrent.

We selected high quality articles (published in top journals/conferences or highly cited) that used GHTorrent to see how researchers chose samples to avoid potential threats. The results are shown in TABLE I.

TABLE I. SAMPLE SELECTION METHODS IN LITERATURE

Method	Literature
Remove projects that have poor performance in some dimensions (e.g., pull request)	[12-18]
Select projects that are top in some dimensions (e.g., star number)	[19-21]
Only consider projects in several programming languages (e.g., ruby)	[13, 14, 18]
Remove projects that are forks	[17, 18]
Use a task-related strategy (e.g., select .rb files in ruby projects)	[12, 13, 17, 18, 21]
Have no clear project selection strategy	[22-25]

As shown in TABLE I, sample selection methods are not unified in literature. Most of the literature aims to select samples according to specific criteria (e.g., projects that are top in number of stars), which reduces the diversity of samples. In addition, these sample selection methods also do not distinguish between development projects and the projects for other purposes, like storage. Intuitively, development projects and projects for other purposes are developed in different ways,

and this will negatively affect the validity of research results. In this work, we tried to solve this problem.

## III. STUDY DESIGN

As our goal is to automatically detect public development projects, and standard datasets are required to validate our proposed model. In this section, we discuss how to create a standard dataset and how to automatically detect public development projects.

### A. Key Concepts

In this study, the key concept is public development project, thus we first give its definition in two aspects:

**Public project:** Projects that are not built for private use. Anyone has the opportunity to participate in these projects.  
**Development project:** Projects that are set up for software development. This type of tasks include repositories of libraries, plugins, gems, frameworks, add-ons, and so on [3].

### B. Research Questions

We formulate the following research questions (RQs) to investigate the feasibility of automatic detection of public development projects.

**RQ1:** Can public development projects be detected automatically without collecting additional data?

**Rationale:** There are many datasets (e.g., GHTorrent) and tools (e.g., GitHub API and web crawlers) that can be used to collect OSS data. However, although some methods (e.g., web crawlers) can get rich information, these methods are inefficient due to access restrictions of GitHub to fixed IP addresses. In order to increase the usability of our approach, we decided to do this work with only readily available data. In this study, we only use data from GHTorrent.

**RQ2:** How well can public development projects be detected?

**Rationale:** Existing sample selection methods have either a good recall rate (e.g., removing poor projects) or a good precision rate (e.g., selecting top projects). These methods have their own flaws: (1) inaccurate samples lead to inaccurate conclusions; (2) missing samples result in poor generalizability of the conclusions. In this study, we aim to automatically select public development projects with both high precision and recall rates.

### C. Standard Dataset

Creating a standard dataset means that we need to know which projects are public development projects. Since we cannot obtain a publicly available dataset that contains such information, we decided to manually create such a dataset (i.e., extending the GHTorrent dataset).

In order to make the dataset more convincing, the size of our dataset should not be too small. To achieve this goal we asked several master and PhD students on software engineering to manually examine which projects are public development projects. Due to the limited resource, we could not mobilize many people to confirm all projects hosted on GitHub, which contains more than 10 million projects. Therefore, with limited

samples that we can manually check, we should create a strategy to ensure that our samples contain as many types of projects as possible.

As the number of projects that we can manually check is limited (around 10,000 projects according to our available resource), we should select 10,000 samples from over 10 million projects. Then there are two possible strategies: (1) Select all projects created over a period of time; (2) Randomly select a specific number of projects. Considering that a random selection is unstable (i.e., different selection results may contain different types of projects), we decided to select all projects created over a period of time in this study.

After selecting samples, we need participants to decide whether a sample is a public development project. This process may introduce personal bias. In order to alleviate this problem, we first defined how to identify public software development projects (see TABLE II). Then, we randomly selected 100 samples and strictly labeled these samples according to identification criteria in TABLE II. We showed these labeled samples to participants to give them a first impression to reduce their personal bias.

TABLE II. IDENTIFICATION CRITERIA OF PUBLIC PROJECTS AND SOFTWARE DEVELOPMENT PROJECTS

Category	Identification
Public project	If the project’s description and readme file do not state that the project is a <b>private project</b> (i.e., this project is established for the project owner’s own use), we classify this project as a public project.
Software development project	We classify projects as <b>software development projects</b> if their contents are files used to build tools of any sort. This type of use includes repositories of libraries, plugins, gems, frameworks, add-ons, and so on [3].

Lastly, we assign tasks to participants according to the following procedure (see Fig. 1):

- 1) Divide the set of samples into several subsets.
- 2) For each sample subset, assign a participant to decide whether each sample meets the criteria in TABLE II. If participants are not sure which category a sample belongs to, they can put such a sample in a collection “undecided” first.
- 3) Collect all undecided samples, and convene all participants to discuss how to categorize these samples. For each sample, participants need to give their classification and reasons. If a consensus cannot be achieved, we vote to decide which category the project should be assigned to .
- 4) Form the final dataset by combining clearly categorized samples based on the discussion.

In conclusion, we first collected samples created over a period of time in GitHub. Then we assigned these samples to participants for manual classification. Finally, by summarizing these results, we got the final standard dataset.

#### D. Automatic Detection of Public Development Projects

After getting the standard dataset, the next step is to automatically classify these samples. In the GHTorrent dataset, the project-related data include issue data, commit data, committer data, and basic data of the project (owner,

description, star number, watcher number, language, and readme file). Hence, we collected 1,000 samples (in the sample set discussed above) and their project-related data to conduct a pilot study that investigates the difference between public development project samples and the remaining samples.

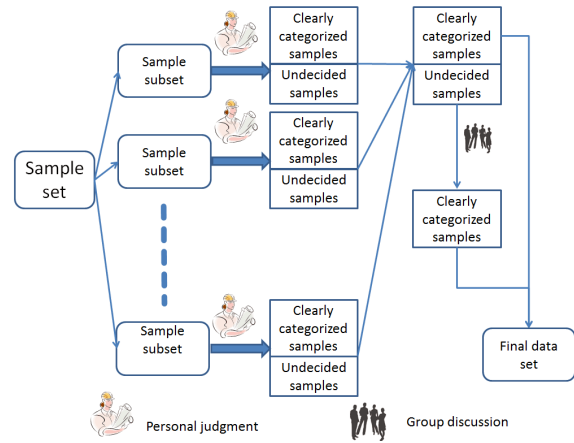


Figure 1. Procedure of classifying projects.

The main findings of this pilot study are twofold (1) most projects (80% in our samples) clearly stated their purposes in the project description. (2) We cannot simply exclude a project that contains some special strings. For example, a project with the word “test” in its description does not necessarily mean that this project is a private project, and this project may be a ‘testing tool’.

The inspiration from this pilot study is that the decision tree model is suitable to address our RQs, because a combination of keywords can avoid the problem in finding (2). For example, if a description contains the keyword ‘test’, the project may be a private project that aims to test how to use GitHub. But if the description also contains the keyword ‘tool’, then this project may be a testing tool, and should be classified into public development projects.

In order to improve the keywords that are used in the decision tree, we used the following procedure to gradually improve the quality of matched strings (see Fig. 2):

- (1) Generate an initial matching string from the survey samples (1,000 samples discussed above) in the pilot study. For example, we can match projects from other ecosystem by matching string “%mirror%”.
- (2) Test the results through the J48 decision tree algorithm on the standard dataset.
- (3) Count the number of misclassified samples; if the number exceeds 15% of the standard dataset, investigate these samples and update our strings, and return to (2), else, output the final strings.

As shown in TABLE I, there are several methods to select research samples. Removing projects that are forks is necessary for all researchers who want to study base projects. This method should be used by all studies that aim to investigate base projects, and consequently it is not necessary to compare our model with these methods. Some task-related strategies were used for special research purposes and these methods are not in the scope of our comparison. For example, selecting only

Java projects means that the purpose is to study the pattern that exists merely in Java projects. As our objective is to study a common model to select samples, we should not compare our model with those task-related methods. Hence, our model should be compared with the two methods: (1) removing poor projects and (2) selecting top projects.

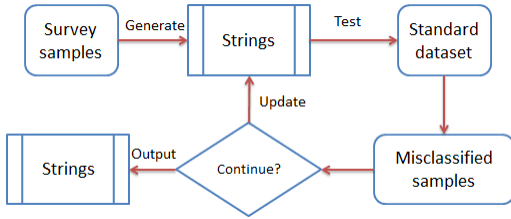


Figure 2 The process of updating string.

There are different dimensions to measure success. We select four dimensions that are widely accepted and easy to obtain [26]: committer numbers, community member numbers, star numbers, and watcher numbers. Then we used different thresholds to segment the dataset to select samples. Methods are shown in Fig. 3.

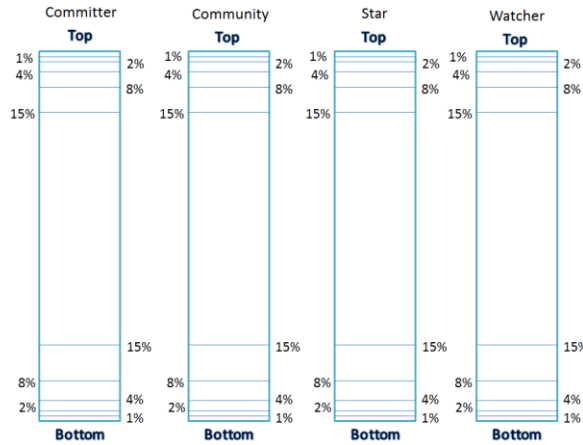


Figure 3 Baseline method schematic diagram.

For each dimension, we have ten choices to select samples (i.e., selecting top 1%,2%,4%,8%,15% projects or deleting bottom 1%,2%,4%,8%,15% projects). Therefore, we should compare 40 different strategies on selecting samples.

#### IV. STUDY RESULTS

##### A. Standard Dataset

Following the procedure in Section III, we first collected all projects established in GitHub between 2012-1-1 and 2012-1-15. We chose this period of time because GHTorrent started data collection in 2012, and we need to collect about 10,000 projects (considering about the work load). Then, we deleted projects that are forks (recommended by [17, 18]) and obtained 8,638 samples. In addition, we deleted projects that are not described in English or have been removed. Some projects in the GHTorrent dataset have been removed from GitHub by their owners, therefore, we deleted such projects due to the lack of sufficient development information to classify such projects.

Finally, we got 6,715 projects acting as the standard dataset in this study.

Then, we labeled these samples by human judgment. These samples were sent to four participants and we asked them to determine whether these projects are public development projects using our definitions in TABLE II. Doubtful projects (625 projects) were discussed and classified through a meeting. Finally, we obtained 6,715 labeled projects.

##### B. Sample Features

After several rounds of iterative process described in Fig. 2. For each sample, we collected keywords existed in the description and URL as features of the sample. Besides, we also collected basic information of projects to help classify samples. All features are shown in TABLE III.

TABLE III. SAMPLE FEATURES

Feature Source	Feature Content	Remarks
Description	mirror, fork, moved, longer, test, personal, website, framework, tool, module, component, app, system, dotfiles, collection, blog, plugin, library, server, config, guide, set, repository, deprecated, file, demo, my, github, dot, simple, extension, helper, template, http, https, source, setting, list of, collection of, example, vim, sample, university, school, practice, backup, intro, first, tutorial, course, copy, null, localization, storage, theme, resume, clone, translation, documentation	If the description of a project contains a feature string, this feature is set to 1. If not, this feature is set to 0. For example, project <i>i</i> 's description contains string "mirror", then we set feature "mirror" of project <i>i</i> to 1.
Basic information	star number, watcher number, community member number, committer number. have language	This information is available through the GitHub API.
URL	Dot, config, doc	If URL of a project contain this string, this feature is set to 1. If not, this feature is set to 0.

##### C. Models and Effects

We used the J48 algorithm in Weka<sup>1</sup> tool to fit the model, and debug parameter *confidencefactor*<sup>2</sup> to control model size and results. Then, we got two typical models: a simple model to automatically classify projects and a complex model (in need of human judgment) that can meet strict requirements of studies.

###### 1) Simple model

When *confidencefactor* is set to 0.05, we can get a simple model. The decision tree model is shown in Fig. 4. This model can achieve precision of 0.827 and recall of 0.947 in classifying public development projects. Furthermore, this model is stable, in the sense that precision and recall do not change drastically (precision of 0.820 and recall of 0.941 relatively) when using the 10-fold cross-validation.

###### 2) Complex model

<sup>1</sup> <https://www.cs.waikato.ac.nz/ml/weka/>

<sup>2</sup> A parameter that affects the pruning process of a decision tree. The smaller the parameter value, the smaller the model.

```

simple = 0
Tutorial = 0
dot = 0
  have_language = 0: FALSE
  have_language = 1
  mirror = 0
  my = 0
  collectionof = 0
  fork = 0
  personal = 0
  url_dot = 0
  demo = 0
  example = 0
  test = 0
  url_config = 0
  config = 0
  blog = 0: TRUE
  blog = 1
  community <= 26
  star <= 4
  committer <= 1: TRUE
  committer > 1: FALSE
  star > 4: FALSE
  community > 26: TRUE
  config = 1
  watcher <= 5: FALSE
  watcher > 5: TRUE
  url_config = 1
  vim = 0
  committer <= 2: FALSE
  committer > 2: TRUE
  vim = 1: TRUE
  test = 1
  watcher <= 1
  framework = 0: FALSE
  framework = 1: TRUE
  watcher > 1: TRUE
  example = 1
  watcher <= 13
  framework = 0: FALSE
  framework = 1: TRUE
  watcher > 13: TRUE
  demo = 1
  example = 0: FALSE
  example = 1: TRUE
  url_dot = 1
  is_null = 0
  set = 0
  config = 0: TRUE
  config = 1: FALSE
  set = 1: FALSE
  is_null = 1: FALSE
  personal = 1: FALSE
  fork = 1: FALSE
  collectionof = 1: FALSE
  my = 1: FALSE (
  mirror = 1: FALSE
  dot = 1: FALSE
  Tutorial = 1: FALSE
simple = 1: FALSE

```

Figure 4. Simple decision tree model. Nodes with TRUE or FALSE is leaf nodes and TRUE or FALSE is the corresponding judgment on these nodes.

When *confidencefactor* is set to 0.5, we can get a complex model. The decision tree model cannot be shown in this paper due to the space limit, and this decision tree is provided online<sup>3</sup>. The complex model can achieve a precision of 0.837 and a recall of 0.956. The complex model has a distinct characteristic: a large proportion of misclassified samples are located on two leaf nodes. We showed these paths which are the classification conditions to obtain projects in these leaf nodes in TABLE IV.

TABLE IV. THREE DECISION PATHS THAT CONTAIN THE VAST MAJORITY OF MISCLASSIFIED SAMPLES

Decision Path (common part)	Decision Path (sub-part)	Correct/Incorrect
Simple = 0; tutorial = 0; dot = 1; have_language = 1; mirror = 0; my = 0; collectionof = 0; fork = 0; personal = 0; url_dot = 0; demo = 0; example = 0; test = 0; url_config = 0; config = 0; blog = 0; plugin = 0; library = 0; framework = 0; star <= 2; sample = 0; source = 0; set = 0; committer <= 2; app = 0	Committer <= 1; Star > 0; Classify_Result= TRUE	1,467/355
	Committer > 1; community <= 2; Classify_Result= TRUE	538/113

If we can manually confirm the samples (2,473 out of 6,715, 36.8%) on these nodes, this model achieves a precision of 0.926 and a recall of 0.959. The result can satisfy most of strict requirements of studies.

#### D. Comparison with Baseline Method

We compared our model with two base-line methods: selecting top projects and deleting bottom projects. The results are shown in TABLE V and TABLE VI, respectively.

<sup>3</sup> <https://github.com/sekematerial/S-E-K-E-supplementary-material>

TABLE V. RESULTS OF SELECTING TOP PROJECTS

	top	precision	recall		top	precision	recall
Committer	1%	0.761	0.011	Star	1%	0.865	0.013
	2%	0.768	0.233		2%	0.880	0.026
	4%	0.791	0.048		4%	0.865	0.052
	8%	0.783	0.095		8%	0.834	0.101
	15%	0.771	0.176		15%	0.824	0.188
Community	1%	0.805	0.012	Watcher	1%	0.895	0.013
	2%	0.835	0.025		2%	0.895	0.027
	4%	0.835	0.050		4%	0.869	0.052
	8%	0.811	0.098		8%	0.843	0.102
	15%	0.772	0.176		15%	0.810	0.184

TABLE VI. RESULTS OF DELETING BOTTOM PROJECTS

	bottom	precision	recall		bottom	precision	recall
Committer	1%	0.656	0.988	Star	1%	0.655	0.986
	2%	0.655	0.976		2%	0.652	0.973
	4%	0.652	0.952		4%	0.648	0.947
	8%	0.646	0.904		8%	0.642	0.898
	15%	0.637	0.824		15%	0.627	0.811
Community	1%	0.655	0.987	Watcher	1%	0.654	0.986
	2%	0.653	0.974		2%	0.652	0.972
	4%	0.649	0.949		4%	0.648	0.947
	8%	0.644	0.901		8%	0.640	0.897
	15%	0.635	0.821		15%	0.629	0.814

We can see from the results that compared with our model (0.827 precision and 0.943 recall), these two baseline methods have obvious weaknesses. Selecting top method has a low recall rate, and the precision of deleting bottom method is not very high. Besides, the two methods do not have room for debugging, which means that there are no direct methods to improve the results of the two methods, while by using 63.2% less human effort than manually confirming all samples, our complex model can get a classification result with precision of 0.926 and recall of 0.959. This is an advantage of our approach.

#### E. Answers to Research Questions

We formulated two research questions and perform experiments on the standard dataset with 6,715 labeled samples. Our results can answer these research questions:

**Answer to RQ1:** Yes, we can automatically detect public development projects with only descriptions and basic properties of projects (e.g., committer number), and achieve acceptable accuracy (0.827 precision and 0.943 recall).

**Answer to RQ2:** (1) Our model performs better than existing methods. The precision and recall rates are acceptable to be applied to the experiments on GitHub. (2) For studies that have strict requirements on the dataset, our work can reduce 63.2% human resources in selecting samples.

## V. THREATS TO VALIDITY

In this section, we identified several threats to the validity of the study results.

#### A. Construct Validity

In this study, we only measured whether a project is a public development project. Thus, the construct validity is whether projects classified as “TRUE” are real public development projects. As the concept **development project** and **public project** are clearly defined and we asked participants if they had any doubt to classify a sample. Doubtful projects were put into the “undecided” set to be

further discussed by a group. Then, in the discussion session, all undecided projects are discussed and decided. Hence, the construct validity is limited.

### B. External Validity

External validity in this study depends on whether the obtained results can be generalized to the GitHub ecosystem, or further other OSS ecosystems. The information used in this study to classify a public development project only contains description, URL, and basic properties of the project, and such information also exists in other OSS ecosystems. We believe that our model can be applied to other OSS ecosystems.

### C. Reliability

The GHTorrent dataset used in this study is provided by [10], and it is a public and popular dataset for studying OSS development behaviors. Hence, this study can be replicated using the dataset. At the same time, to mitigate personal bias, we asked participants to avoid classifying samples that may cause disagreements, and then classified these samples through discussion in the data collection procedure. Hence, we believe that our work is relatively reliable.

## VI. CONCLUSIONS

The study aims to develop a model to automatically detect public development projects. The main points of this study are summarized as follows. First, we can automatically detect public development projects with a precision of 0.827 and a recall of 0.943, which is better than existing sample selection methods. Second, by using 63.2% less human effort than manually confirming all samples, we can get better results with a precision of 0.926 and a recall of 0.959, which can meet strict sample requirements.

### ACKNOWLEDGMENT

This work is supported by the National Key Research and Development Program of China (Nos. 2017YFB1400602 and 2016YFB0800401), the National Natural Science Foundation of China (Nos. 61572371, 61702377, and 61773175), the Wuhan Yellow Crane Special Talents Program, the CPSF (No. 2015M582272), the Natural Science Foundation of Hubei Province (No. 2016CFB158), and the Fundamental Research Funds for the Central Universities (No. 2042016kf0033).

### REFERENCES

- [1] V. Cosentino, J. L. C. Izquierdo, and J. Cabot, "A Systematic Mapping Study of Software Development With GitHub," *IEEE Access*, vol. 5, pp. 7173-7192, 2017.
- [2] G. Gousios and D. Spinellis, "GHTorrent: Github's data from a firehose," in *International Working Conference on Mining Software Repositories*, 2012, pp. 12-21.
- [3] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "An in-depth study of the promises and perils of mining GitHub," *Empirical Software Engineering*, vol. 21, pp. 2035-2071, 2016.
- [4] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *Acm Sigkdd Explorations Newsletter*, vol. 11, pp. 10-18, 2009.
- [5] K. Manikas and K. M. Hansen, "Software ecosystems—a systematic literature review," *Journal of Systems and Software*, vol. 86, pp. 1294-1306, 2013.

- [6] J. Howison and K. Crowston, "The perils and pitfalls of mining SourceForge," in *International Working Conference on Mining Software Repositories*, 2004, pp. 7-11.
- [7] D. Weiss, "Quantitative analysis of open source projects on sourceforge," in *Conference on Open Source Software*, 2005, pp. 100-104.
- [8] A. Rainer and S. Gale, "Evaluating the Quality and Quantity of Data on Open Source Software Projects," in *Conference on Open Source Software*, 2005, pp. 11-15.
- [9] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "The promises and perils of mining github," in *International Working Conference on Mining Software Repositories*, 2014, pp. 92-101.
- [10] G. Gousios, "The GHTorrent dataset and tool suite," in *International Working Conference on Mining Software Repositories*, 2013, pp. 233-236.
- [11] T. N. Nguyen, "Boa: A language and infrastructure for analyzing ultra-large-scale software repositories," in *International Conference on Software Engineering* 2013.
- [12] G. Gousios, M. Pinzger, and A. V. Deursen, "An exploratory study of the pull-based software development model," in *International Conference on Software Engineering*, 2014, pp. 345-355.
- [13] B. Vasilescu, Y. Yu, H. Wang, P. Devanbu, and V. Filkov, "Quality and productivity outcomes relating to continuous integration in GitHub," in *Joint Meeting on FSE/ESEC*, 2015, pp. 805-816.
- [14] Y. Yu, H. Wang, V. Filkov, P. Devanbu, and B. Vasilescu, "Wait for It: Determinants of Pull Request Evaluation Latency on GitHub," in *International Working Conference on Mining Software Repositories*, 2015, pp. 367-371.
- [15] Y. Yu, H. Wang, G. Yin, and T. Wang, "Reviewer recommendation for pull-requests in GitHub: What can we learn from code review and bug assignment?," *Information and Software Technology*, vol. 74, pp. 204-218, 2016.
- [16] G. C. Murphy, R. Terra, J. Figueiredo, and D. Serey, "Do developers discuss design?," in *International Working Conference on Mining Software Repositories*, 2014, pp. 340-343.
- [17] R. Kikas, M. Dumas, and D. Pfahl, "Using dynamic and contextual features to predict issue lifetime in GitHub projects," in *International Working Conference on Mining Software Repositories*, 2016.
- [18] E. Constantinou and T. Mens, "Socio-technical evolution of the Ruby ecosystem in GitHub," in *IEEE International Conference on Software Analysis, Evolution and Reengineering*, 2017, pp. 34-44.
- [19] R. Padhye, S. Mani, and V. S. Sinha, "A study of external community contribution to open-source projects on GitHub," in *Working Conference on Mining Software Repositories*, 2014, pp. 332-335.
- [20] M. Hilton, T. Tunnell, K. Huang, D. Marinov, and D. Dig, "Usage, costs, and benefits of continuous integration in open-source projects," in *IEEE/ACM International Conference on Automated Software Engineering*, 2016, pp. 426-437.
- [21] J. Xavier, A. Macedo, and M. D. A. Maia, "Understanding the popularity of reporters and assignees in the Github," in *International Conference on Software Engineering & Knowledge Engineering*, 2014.
- [22] K. Aggarwal, A. Hindle, and E. Stroulia, "Co-evolution of project documentation and popularity within github," in *Working Conference on Mining Software Repositories*, 2014, pp. 360-363.
- [23] M. M. Rahman and C. K. Roy, "An Insight into the Pull Request of GitHub," in *Working Conference on Mining Software Repositories*, 2014, pp. 364-367.
- [24] K. Yamashita, S. Mcintosh, Y. Kamei, and N. Ubayashi, "Magnet or sticky? an OSS project-by-project typology," 2014, pp. 344-347.
- [25] J. Cabot, J. L. C. Izquierdo, V. Cosentino, and B. Rolandi, "Exploring the use of labels to categorize issues in Open-Source Software projects," in *IEEE International Conference on Software Analysis, Evolution and Reengineering*, 2015, pp. 550-554.
- [26] K. Crowston, J. Howison, and H. Annabi, "Information systems success in free and open source software development: Theory and measures," *Software Process: Improvement and Practice*, vol. 11, pp. 123-148, 2006.

# Recovering Three-Level Architectures from the Code of Open-Source Java Spring Projects

Alexandre Le Borgne<sup>1</sup>, David Delahaye<sup>2</sup>, Marianne Huchard<sup>2</sup>,  
Christelle Urtado<sup>1</sup>, and Sylvain Vauttier<sup>1</sup>

<sup>1</sup>LGI2P, IMT Mines Ales & Montpellier University, Ales, France  
{Alexandre.Le-Borgne, Christelle.Urtado, Sylvain.Vauttier}@mines-ales.fr

<sup>2</sup>Montpellier University, CNRS, LIRMM, Montpellier, France  
{David.Delahaye, Marianne.Huchard}@lirmm.fr

## Abstract

*Despite the well-admitted benefits of keeping design decisions as a documentation all along the lifecycle of software, many software projects have lost this information. In order to use design information to guide software maintenance and evolution, this paper proposes to retro-engineer architecture descriptions from source code. The originality of this work is to target a three-leveled architecture description language which represents software specification, configuration and deployment. Retro-engineering these three levels will provide a more precise source of guidance for the maintenance of software. Targeted projects are open-source Java projects that use Spring to describe the implemented "architecture".*

**Keywords:** Component-Based Software Engineering, Model-driven engineering, Architecture retro-engineering from code, Architecture evolution, Component reuse, Architecture reuse.

## I. Introduction

As software systems constantly become more complex, retrieving design decisions has become an increasingly important problematic when conceptual documentation is missing. However, despite numerous researches in the field of software architecture reconstruction, few work was dedicated to extract raw ("as implemented") component-based description. It is important, in the first place, to understand design decisions to recover architectures as

they are implemented and to not perform any improvement (re-engineering tasks) altogether. Moreover, it is important to represent the software at three abstraction levels in order to trace design decisions through the whole development process. To do so, we use the Dedal [12], [8] architecture description language (ADL) developed by our team. This paper proposes to reconstruct component-based architectures from Java Spring [6] projects.

The remainder of this paper is organized as follows. Section II presents the background of our approach. The core of the paper is developed in Section III where component-based architecture reconstruction from Java Spring projects is explained. Section IV details the existing work in software architecture reconstruction and Section V concludes on perspectives.

## II. Background

### A. Dedal, a Three-Level Architecture Description Language

Dedal [12], [8] is a three-level architecture description language (ADL) designed to give a representation of the entire life cycle of architectures and a support to manage their evolution. Design is represented by the *Specification* level which is composed of abstract component types. Those types are called *roles* which means that they define the functionalities present in the components of the future software. Implementation choices are captured by the *Configuration* level. This architecture level is composed of concrete component classes which are realizations of

the roles. Deployment is described in the *Assembly* level. This level is composed of a set of component instances that define how to tailor software for specific execution contexts.

## B. Spring

Java Spring framework [6] is widely used in industry to manage the deployment of software architectures. It provides standardized architecture deployment capabilities thanks to a container that is able to handle explicit architecture descriptors. Architecture descriptors are defined as XML files or directly embedded in the code as annotations. They are based on the concept of beans, which define the objects that the container must instantiate and connect in order to set the initial architecture up. For instance, the deployment descriptor of Figure 1a is composed of four beans: *lampDesk*, *lampSitting*, *clock1* and *orchestrator1*.

Connections between beans are handled by the container using dependency injection to preserve decoupling. Beans only declare reference attributes corresponding to their dependencies with other beans. These dependencies are then resolved at runtime thanks to the connections defined in the deployment descriptor. For instance, the deployment descriptor in Figure 1a defines three connections between the orchestrator, the lamps and the clock beans. Those connections are defined by the *property* tag which corresponds to the injection of dependencies. For instance, `<property name = "clock" ref = "clock1"/ >` sets the *clock1* bean as the *clock* property of the *orchestrator1* bean.

As compared to raw code, Spring projects provide some

```
<bean class="AdjustableLamp" id="lampDesk" />
<bean class="AdjustableLamp" id="lampSitting" />
<bean class="Clock" id="clock1" />
<bean class="Orchestrator" id="orchestrator1" >
  <property name="lamps">
    <set><ref bean="lampDesk" />
    <ref bean="lampSitting" /></set></property>
  <property name="clock" ref="clock1" />
</bean>
```

(a) HAS Spring description

```
public class Clock implements Time {
  public void run() {...}
}
public class AdjustableLamp extends Light {
  ...
  public void switchState(State s) {...}
  public void adjustIntensity(int value) {...}
}
public class Orchestrator extends HomeOrchestratorImpl {
  protected void run() {...}
}
```

(b) HAS classes instantiated in Figure 1a

explicit architectural descriptions. However, these descriptions do not capture design decisions and thus cannot be considered as abstractions of the software architecture.

## III. Extracting Component-Based Software Architectures

In order to ease the extraction of information from the Spring descriptor, using a model-driven approach, a small domain specific language named SpringDSL has been developed. It consists of an implementation of the Spring XML descriptor grammar in EMF<sup>1</sup>. XText<sup>2</sup> has been used to automatically generate the corresponding EMF metamodel since we could not find an already developed metamodel. This metamodel enables thus to parse Spring XML descriptors and get all their content as concept instances.

This section discusses how each concept of Dedal is extracted from the source code and a Spring deployment description.

### A. Extracting Components

Considering component extraction as a model transformation between SpringDSL and Dedal, only a simple mapping is required to extract the assembly and also a small part of the configuration models. To do so, the eclipse QVT<sup>3</sup> language is used since it defines model to

<sup>1</sup><https://www.eclipse.org/modeling/emf/> [Last seen 03-14-2018].

<sup>2</sup><https://www.eclipse.org/Xtext/> [Last seen 03-14-2018].

<sup>3</sup><https://projects.eclipse.org/projects/modeling.mmt.qvt-oml> [Last seen 03-14-2018].

```
public interface Time {
  public void run();
}
public abstract class Light {
  ...
  public abstract void switchState(State s);
}
public abstract class HomeOrchestrator {
  Light[] lights;
  Time clock;
  public abstract void run();
}
public class HomeOrchestratorImpl
  extends HomeOrchestrator {
  protected void run(){...}
  public HomeOrchestratorImpl(Light[] lights,
    Time clock) {...}
  public Light[] getLights() {...}
  public void setLights(Light[] lights) {...}
  public Time getClock() {...}
  public void setClock(Time clock) {...}
}
```

(c) HAS most abstract classes

Figure 1: HAS implementation

```

component_role HomeOrchestrator_role
id "_C-nKzSH3EeicKLj-dMtreg" required_interfaces
interface ILight_HomeOrchestrator_role
interface_direction required
implementation ILight_Type;
interface ITime_HomeOrchestrator_role
interface_direction required
implementation Time;

```

Figure 2: Extracted component class: *Orchestrator*

model transformations through the concept of mapping. As a first step, beans are mapped to the component instances of the architecture **Assembly**, using the *id* of the beans as the name of the components and the *class* attribute as the instantiated component class. Thus the bean tag describing the *Orchestrator* instance *orchestrator1* in Figure 1a is mapped as the **component\_instance** of Figure 3 named *orchestrator1* that is an **instance\_of** the **primitive\_component\_class** *Orchestrator*. If the component class does not exist in the **Configuration** yet, then it is created.

Figures 1b and 1c present an extract from the Java code of the Home Automation Software (HAS) example. Code introspection enables to extract complementary information required to build higher level architecture models and more detailed component definitions.

For generating the component roles of the architecture **Specification**, the type hierarchy of the beans classes is analyzed, in order to extract the most generic, thus reusable, architecture model as possible. The main idea is to retrieve the abstract superclasses that are realized by the bean class corresponding to a component class. To extract the component role, the type hierarchy is traversed and the role which is picked is the most generic component role which still holds all the required interfaces that are present in the corresponding component class and which preserves the connections which exist in the **Configuration**. Figure 2 is the **component\_role** *HomeOrchestrator\_role* that is realized by the **primitive\_component\_class** *Orchestrator*.

## B. Extracting Interfaces

Two types of interfaces are distinguished: (i) **provided** and (ii) **required** interfaces. All the methods that are provided by the beans classes must be provided into respective component interfaces. However, in order to not provide only one large interface per component, the interfaces are cut according the type hierarchy of classes. In other words, each implemented interface is mapped as a component interface and if a class does not implement an interface, a "conceptual" interface is extracted, which is composed of the public methods of the beans class, except for getters/setters that are used whether to initialize properties or

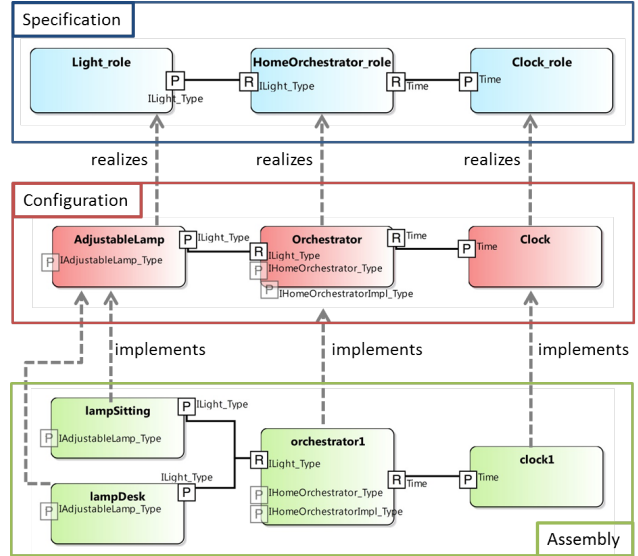


Figure 3: HAS generated Dedal three-level architecture

```

class_connection "_C9ZCsCH3EeicKLj-dMtreg"
client Orchestrator.ILight_Orchestrator
server AdjustableLamp.ILight_AdjustableLamp;

```

Figure 4: Extracted class connection

to manage connections. For extracting required interfaces, the reference attributes declared in the Spring descriptor are used to manage the binding of the beans. The type and the name of the attribute are used to generate the type and the name of the corresponding interface.

The *HomeOrchestrator\_role* extracted interfaces are described in Figure 2 with their names, direction and implemented types.

## C. Extracting Connections

The XML description (Figure 1a) makes it possible to start the extraction of connections between components. Indeed, thanks to the dependency injection, clients and servers of connections are identifiable. For instance, in the current example, the *orchestrator1* bean contains a property which refers to *clock1*, so it is possible to map a new connection between *orchestrator1* and *clock1*. This connection is propagated to the **Configuration** level by creating a connection between the two instantiated component classes. Following the same principle, the connection between the realized component roles is created.

Then the interfaces that are implied in connections must be set. To do so, we search among two connected components which are their matching interfaces. For matching two interfaces, their types must be equal and their direction complementary (provided with required). Thus, for instance in our case, *Clock* provides the *ITime* interface



of type *Time* and, *Orchestrator* requires an interface of the same type (Figure 3). Then those two interfaces match and the connection presented in Figure 4 can be set. Then it is propagated following the instantiate relation between component class interfaces and component instance interface (Figure 3). Finally, connection between roles are set in the same way as the connections between component classes.

Figure 3 is the visual representation of the three-level Dedal architecture which is composed of four component instances into the **Assembly** (that correspond to the beans of the Spring description), the component classes of the **Configuration** that are instantiated by the component instances and also the component roles into the **Specification** which are realized by component classes. The connections between components are also represented.

#### IV. Related Work

This section narrows the studied approaches to the ones which extract component-based architecture descriptions and, if possible, from object-oriented code. Moreover, retro-engineering approaches which consist in simply abstracting the software artifacts for retrieving raw design decisions are differentiated from re-engineering ones which intend to re-organize the extracted information and/or the software artifacts.

In their work, Ducasse *et al.* [3] defined a taxonomy for categorizing software architecture reconstruction approaches. Following this taxonomy, the **goals** of the discussed approach are twofold. The first goal is to improve component reuse, by extracting component-based architecture descriptions, such as MAP [10], PuLSE/SAVE [7] and ROMANTIC [1], [9] approaches, but targeting the Dedal [12], [8] ADL. The second goal is to provide the foundations for managing conformance checking (Bauhaus [4], [2], DiscoTect [11], PuLSE/SAVE [7]), evolution, co-evolution (PuLSE/SAVE [7], Huang *et al.* [5]) and maintenance using the formal rules that have previously [8] been defined in Dedal.

However none of the studied methods intends to extract raw information of how the software is implemented. Moreover, all the discussed approaches only deal with two levels of abstraction (*i.e.*, implementation and architecture) that may not correspond to the same paradigms (code vs component-based architecture description). Indeed three component-based architecture descriptions are essential for maintaining, evolving, tracking software life-cycle since it gives a more global and direct understanding to the architect which can get an overview of the code structure by managing components. Moreover, even the approaches which seem to fit with the discussed one, either recover architecture in a semi automatic manner from execution

trace of software (*i.e.*, DiscoTect [11]) or do not reconstruct raw architecture such as ROMANTIC [1], [9] approach which performs re-engineering of the deployed architecture by clustering classes into bigger semantic components that encapsulate classes.

This is why a retro-engineering approach is proposed that builds three-level component-based architecture description from structural artifacts.

#### V. Conclusion and Future Work

This paper introduces an approach for software architecture reconstruction, using three levels of architecture models (**Assembly**, **Configuration**, **Specification**). An aspect of future work will be to improve and refine the extraction of the **Specification** for making it more abstract.

Real Spring projects have already been identified in open-source repositories in order to perform large scale experimentations on evolution and reuse. Getting projects from open-source repositories will also allow the implementation of versioning mechanisms.

#### References

- [1] Z. Alshara, A. D. Seriai, C. Tibermacine, H. L. Bouziane, C. Dony, and A. Shatnawi. Materializing architecture recovered from object-oriented source code in component-based languages. In *10th ECSA Proc.*, volume 9839 of *LNCS*, pages 309–325, Copenhagen, Denmark, Nov. / Dec. 2016. Springer.
- [2] A. Christl, R. Koschke, and M. A. Storey. Equipping the reflexion method with automated clustering. In *12th WCRE Proc.*, pages 10–98, Pittsburgh, USA, Nov. 2005.
- [3] S. Ducasse and D. Pollet. Software architecture reconstruction: A process-oriented taxonomy. *IEEE TSE*, 35(4):573–591, 2009.
- [4] T. Eisenbarth, R. Koschke, and D. Simon. Locating features in source code. *IEEE TSE*, 29(3):210–224, 2003.
- [5] G. Huang, H. Mei, and F. Q. Yang. Runtime recovery and manipulation of software architecture of component-based systems. *Automated Software Engineering*, 13(2):257–281, April 2006.
- [6] R. Johnson, J. Hoeller, K. Donald, C. Sampaleanu, R. Harrop, et al. The spring framework – reference documentation. *Interface*, 21:27, 2004.
- [7] J. Knodel, M. Lindvall, D. Muthig, and M. Naab. Static evaluation of software architectures. In *10th CSMR Proc.*, pages 279–294, Bari, Italy, March 2006. IEEE.
- [8] A. Mokni, C. Urtado, S. Vauttier, M. Huchard, and H. Y. Zhang. A formal approach for managing component-based architecture evolution. *SCP*, 127:24–49, 2016.
- [9] A. Shatnawi, A. D. Seriai, H. Sahraoui, and Z. Alshara. Reverse engineering reusable software components from object-oriented APIs. *JSS*, 131:442–460, 2017.
- [10] C. Stoermer and L. O’Brien. Map–mining architectures for product line evaluations. In *IEEE/IFIP WICSA Proc.*, pages 35–44, Amsterdam, The Netherlands, Aug. 2001.
- [11] H. Yan, D. Garlan, B. Schmerl, J. Aldrich, and R. Kazman. DiscoTect: a system for discovering architectures from running systems. In *26th ICSE Proc.*, pages 470–479, Edinburgh, UK, May 2004.
- [12] H. Y. Zhang, C. Urtado, and S. Vauttier. Architecture-centric component-based development needs a three-level ADL. In *4th ECSA Proc.*, volume 6285 of *LNCS*, pages 295–310, Copenhagen, Denmark, Aug. 2010. Springer.

# An Agent-based Software Framework for Machine Learning Tuning

Jefry Sastre<sup>1</sup>, Marx Viana<sup>1</sup>, Carlos Lucena<sup>1</sup>

<sup>1</sup>Laboratory of Software Engineering (LES) - Pontifical Catholic University - PUC-Rio  
Rio de Janeiro, RJ - Brazil  
{jperez, mleles, lucena}@inf.puc-rio.br

**Abstract**— Nowadays, the challenge of knowledge discovery is to mine massive amounts of data available online. The most widely used approaches to tackle that challenge are based on machine learning techniques. In spite of being very powerful, those techniques require their parameters to be calibrated in order to generate models with better quality. Such calibration processes are time-consuming and rely on the skills of machine learning experts. Within this context, this research presents a framework based on software agents for automating the calibration of machine learning models. This approach integrates concepts from Agent Oriented Software Engineering (AOSE) and Machine Learning (ML). As a proof of concept, we first train a model for the IRIS dataset and then we show how our approach improves the quality of new models generated by our framework.

**Keywords.** *Agent oriented Software Engineering (AOSE); Machine Learning; Time-consuming*

## I. INTRODUCTION

The big data era is coming! According to [1], every minute on the internet over 4 million queries are made on Google, more than 200 million emails are sent and users share almost 2.5 million pieces of content on Facebook, among other actions. The amount of data generated is growing exponentially [2] and we need to be prepared to face all the challenges upfront. Peter Norving at Google's Zeitgeist Conference (2011) refers to this matter, stating: "We don't have better algorithms. We just have more data".

Indeed, the huge volume of data available is a massive challenge to be accepted, but at the same time a vast opportunity to learn from the data to generate more expert artificial intelligence software and to enhance the knowledge discovery processes (KDD). One of the most popular and widely used approaches to generate knowledge is through the machine learning techniques. In order to be better rewarded from machine learning algorithms, we need to adjust their parameters. This calibration process makes the resulting models more accurate and, certainly, more profitable; but the drawback at stake is time. The tuning process is generally done by hand, is highly time consuming and strongly relies on the skills of machine learning experts, turning it into an extenuated, endless process. We foresee a chance to incorporate the Agent Oriented Software Engineering (AOSE) [23] area to automate the process of tuning the models prone to the generation of more accurate models and, at the same time, reduce efforts dedicated to produce more profitable models.

DOI reference number: 10.18293/SEKE2018-074

Agents are software components with autonomy, reactivity, proactiveness and social capabilities [3]. Agent autonomy comes in handy when a system needs to make its own decisions. The agents can also use their proactiveness to guide the tuning process. As a result, it is possible to see how multiagent systems can contribute to the automation of machine learning. To solve this problem, we propose a framework based on software agents to handle the tuning of the machine learning models.

The contributions are: (i) the framework will facilitate building new models that might display good performance based on the previously trained models. This includes a new set of possibilities in the selection of the ways and strategies that will guide the optimizations; (ii) the framework allows the creation of an ensemble of models to predict and negotiate a consensus among all the predictors in order to deliver a solution. In addition, the results of the system do not depend on a single trained model, but rather on a set of models that might be specialized at detecting specific characteristics; (iii) the framework reduces the time spent by the user to train a successful model with a multiagent system to support the training process. The idea is to configure some of the training and allow the framework to handle the training results, the timing and the long wait for the end of the training and the start of a new one without human interference, and (iv) to validate a case scenario, IRIS. This test case is an exploratory study taken as a proof of concept but instantiating the framework and exploiting the agents to generate new models.

This paper is organized as follows. Section 2 gives an overview of the main concepts. Section 3 shows the related work. Section 4 presents the framework. Section 5 describes an exploratory study. Finally, Section 6 offers the conclusion and future work.

## II. BACKGROUND

First, we will discuss the relation between multiagent systems and machine learning. After, the KDD process.

### A. Multiagent Systems and Machine Learning

A multiagent system can be defined as an environment shared by autonomous entities that live, interact, receive information and can act in the environment [5]. These agents are abstractions with the following properties [6]: (i) autonomy — it is the capability of taking their own actions within their environment; (ii) reactivity — it is the capability of response to the changes in the environment, which involves a notion of perception of the environment; (iii) social ability — it is the capability of interaction with other agents and possibly humans, and (iv) proactive ability — it is the capability to take actions towards the agent's goals.

The exploratory study in Section 5 shows how agents' properties are useful in the simulation of the training process to optimize the parameters of a model based on the previously trained models. It also evidences how the software agents are able to propose new models that might be more accurate. The idea of joining together these two areas seems very natural. In artificial intelligence, we consider that software agents are autonomous entities and are capable of making decisions without human interference. On the other hand, learning is a crucial part of the autonomy: the more skilled the agent, the better decisions it will take [7]. Indeed, in most dynamic domains it is extremely hard to predefine the agents' actions, which mostly emerge with new behaviors in order to adapt themselves to the current situation.

There are several aspects to take into account when dealing with machine learning in multiagent systems. First, the coordination of agents — there must be some coordination mechanism for agents to engage and interact in some way. Second, dealing with cooperation can be a problem when agents need to team up to achieve some goals. Third, the noisy environment — specifically, how to deal with supervised learning when the result can be biased by the noise. Finally, together with the noisy environment comes the partial knowledge; to deal with it, agents use strategies and metaheuristics to guide the search, as in [8]. Some approaches use a machine learning model in the agents' activities cycle to take actions [5]. Other approaches use a multiagent system — known as multiagent learning (MAL) — to learn [9] [10]. In the latter approaches the integration of the agents' capabilities and the learning algorithms are combined to solve a problem from another domain. Nevertheless, our approach is a multiagent system applied to a machine learning domain.

### B. KDD Methodologies

The KDD process [11] [12] contains five stages: (i) Selection: This stage is to precisely define a target dataset. It can be done by directly selecting a dataset or a subset of features; (ii) Pre-processing: This stage focuses on cleaning the data. It means that the data most of the times is generated crowded with null values and inconsistencies and needs to be cleaned in order to become profitable; (iii) Transformation: This stage aims at applying some transformation algorithms to generate the final dataset to explore. It is common to use dimensionality reduction algorithms, normalize the data, etc; (iv) Data Mining: This stage focuses on the search of the required patterns in the data according to the mining objectives, and (v) Interpretation/Evaluation: This stage consists of the interpretation and evaluation of the extracted patterns.

## III. RELATED WORK

Many authors [13, 14, 15, 16, 17] broach the idea of creating systems to support the data mining process. A common discussion among all authors is about the target user and the environment — some systems are designed to be used by domain experts and others by data mining experts. Systems dedicated to non-experts, normally focus on the analysis of the domain's specific features while other systems propose educational environments for novices to learn and interact. On the other hand, systems designed to be used by data mining experts focus on performance, optimizations and coding capabilities. Within the context of non-experts, [18] presents a simulation tool that aims at creating an initial intuition on neural networks with a very

user-friendly interface and it has proven to be a great choice for educational purposes. However, the datasets available for analysis are fixed and focused only on gaining some understanding of the learning process. There are some commercial solutions, such as Google Prediction [19] and Azure Machine Learning [20]. Both are on line services and provide support to the data mining process by means of an intuitive interface and a huge collection of ready to use algorithms. However, Azure is not free and Google Prediction's dataset size is limited to 250 megabytes. WEKA [17], [21] is a system that offers a collection of algorithms to explore real world datasets. It has three well defined categories of algorithms: (i) dataset processing, (ii) machine learning schemes, and (iii) output processing. By combining all these tools together, WEKA has proved essential to the analysis process and as an introductory tool for educational environments. However, all these algorithms are presented as black boxes and do not focus on distributed ways to improve the data mining processes.

There are some solutions that target data mining experts and focus on tools to improve the techniques. MLI [15] presents an API to easily code machine learning algorithms, using their proposed operations for data loading and linear algebra to boost the performance; but it relies on the expertise of the programmers rather than the use of previously tested and well-established implementations of the algorithms. ML Base [14] is another solution that provides a Domain Specific Language (DSL) with high level abstractions to simplify the process. It creates very elaborated plans — logical and physical — that come with several optimizations to gain performance and accuracy. The solution aims at solving a problem with a single model. However, the composition of models that create ensembles has been proven to outperform single models, and according to [16] many algorithms and large datasets can be slow and limited. The work [13] presents LARA, a DSL to reduce problems created when the pre-processing and the algebra are done by using different programming paradigms. It includes optimizations that are normally loose in the mismatch of the paradigms. In addition, LARA compiles to an intermediate representation to enable optimizations and finally compiles with different languages. On the other hand, it is embedded into Scala and it is focused on coding. Predict-ML [16] is a software that uses big clinical data to build predictive models automatically. It presents techniques to automatically select algorithms, hyper parameters and temporal aggregations of the clinical data, but the innovations are focuses on the clinical area and the system is still in the design phase.

All these solutions focus on reducing usage complexity, tuning hyper parameters and gaining some understanding of the data, but none of the previous approaches aims at creating a shared environment to enhance the interaction between the users and the system. By using the agent's capabilities, users and agents can both solve the data mining process, complementing each other's weaknesses.

### A. Auto ML

Auto ML is a new area in computer science pursuing the progressive automation of the machine learning process [22]. This area addresses all the aspects which are related to machine learning automation, such as search and selection of model, hyper parameters optimization, feature engineering, meta learning and transfer learning, among others. Within this

context, a challenge to boost new solutions towards the Auto ML goals was created. This challenge includes a novel design element: code submission. The code runs in an open-source platform ensuring there is no human intervention during testing phases and that all proposed solutions run on hardware equality. The challenge contains six phases in which the dataset difficulty is progressively increased. After each phase, the competitors have a Tweakathon time to improve their method with access to the previously tested datasets. This challenge aims at advancing the theoretical state of the art about model selection, implementing useful automation solutions, a chance to compare results of the automatic software and the Tweakathon phase and to disseminate the top solutions and papers.

#### IV. PROPOSED SOLUTION

This section describes the main elements required to understand the solution proposed in this paper. In addition, we will provide an overview of the architecture and discuss the different components, including the data model and the software agents.

##### A. The Architecture

The application is implemented using the software agents, as illustrated in Fig. 1. It contains a module for: (i) data storage (DB); (ii) data access (ORM); (iii) agents; (iv) optimizations (OPT), and (v) API layer — which will bring the functionalities to the final user.

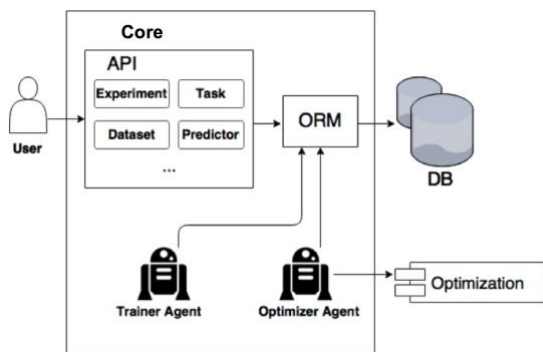


Figure 1. The proposed architecture.

The API is directly connected to the ORM. The ORM is in charge of all the operations that require data access. It allows the system to be independent from the physical data storage and it is also the only way to interact with the data. The data refers to the relevant concepts that appear in the domain and their relationships. All of them are physically saved in the DB module. Considering the user's experience, the main flow of the application only involves the API, the ORM and DB modules. ORM provides stability and independence for the following layers to use, allowing: (i) the change of the data provider without changing the core of the project, and (ii) the design of the logic without specific read, write operations that might bind the solution to a particular data access. The software agents interact in this flow via ORM module and expertly use the main application flow the same way as normal users do. They retrieve, run and propose new experiments in a collaborative environment. By working together, the users (as domain experts) and the agents (as machine learning experts) increase the number of experiments, searching for a better model to identify the desired patterns. The agents in charge of the optimizations trust most of

the algorithmic analyses in the fifth and last module dedicated to the Optimizations. The *Trainer Agent* and the *Optimizer Agent* are both hot spots [23]. Therefore, it is possible to add new models into the system by creating subclasses and implementing the particular details of the new model.

By using the API, the users can evaluate the results, that is, they can check if the results meet the initial objective. This phase is crucial, because the models selected to be deployed will finally be in contact with non-controlled environments and real-life mining examples. Nevertheless, if the users determine that the models are not ready to be used, they can define a new experiment or allow the agents to search for better models. At all times, the users can monitor the results obtained and then, analyze, retrieve and compare several of the model's parameters.

##### B. Data Model

Fig. 2 presents the data model of the concepts involved in the problem. We used the entity-relationship model (ERM) [24]. The entities are: (i) Task: Aims at capturing the training process of a successful model for a machine learning problem, i.e., it is a collection of experiments; (ii) Experiment: Defines an experiment, but this concept just contains the common aspects, such as *running\_time*, *train\_accuracy*, etc; (iii) Decision Tree: Defines a specific kind of experiment. In fact, it defines an experiment to train a decision tree and contains aspects such as *max\_depth*; (iv) Support Vector Machine: Defines a support vector machine type of experiment and contains attributes such as *kernel*; (v) Neural Network: Defines a neural network type of experiment and contains attributes such as the model that specifies the structure of the network; (vi) Host: Defines a computer in the network, and basically selects the computer in which the model is going to be trained; (vii) Dataset: Represents a generic data collection, used as the examples to train a model; (viii) RData: Represents a particular type of dataset generated from a script executed in R [25] and contains the environment variables at the save point, and (ix) CSV: Represents a standard data exchange format. Most of the time it is a collection of comma separated fields.

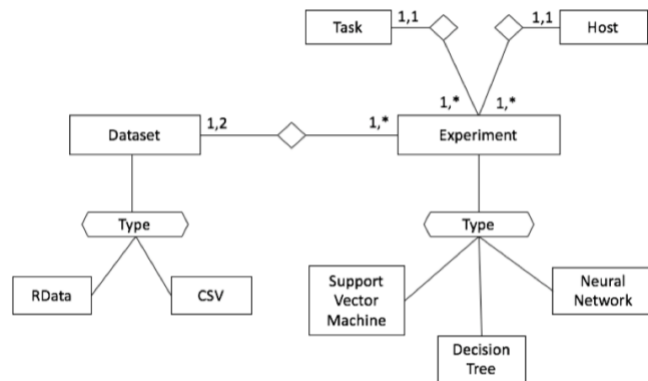


Figure 2. The architecture proposed.

##### C. Agents Model

Figure 3 shows and details the agent-based model proposed. In all the cases, the agent's cyclic behavior was the best option for these software agents – for instance, in the exploratory study presented in Section 5 the agents have a cyclic behavior with 10 seconds between iterations. The *Trainer Agent* is responsible for training an experiment. In

order to do so, it has to accomplish several subtasks. First of all, it needs to understand the type of experiment that the agent is going to execute. For each type of experiment, there are different parameters used to set up the training process. Based on these parameters the agent determines the type of dataset that is going to be used and it loads the data. At this point, the strategy pattern [26] was used to define which algorithm should be chosen to train and validate the results. After the validation, the agent has to collect all the variables being measured and write the experiment back. Fig. 4 describes this process.

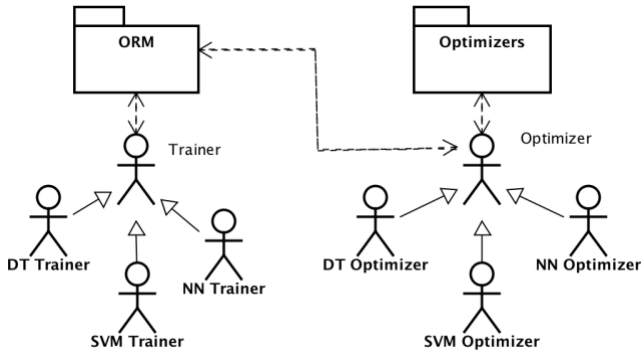


Figure 3. Agents Model.

A specific trainer was created to override the specificities of each model and to set up some initialization variables, such as the type of experiment. To run an experiment, both the experiment and the datasets to be used in the training and testing must be previously defined. This process only runs the experiments and collects the results. On the other hand, due to the characteristics of the agent's cyclic behavior, if there are no experiments programmed to run, the agent waits a few seconds and asks again. Therefore, once a new experiment is added to the database, it will be automatically detected and executed at the right time. Another important detail is that the experiments are executed as if they were on a queue — one at a time in each host. But it is possible to program a set of experiments that the agents will automatically run until all the experiments have been executed.

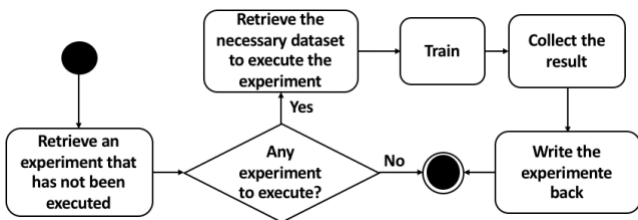


Figure 4. Trainer Agent Activity Diagram.

The *Optimizer Agent* is responsible for generating new models that might have good performance and accuracy based on the previously executed experiments of the same type.

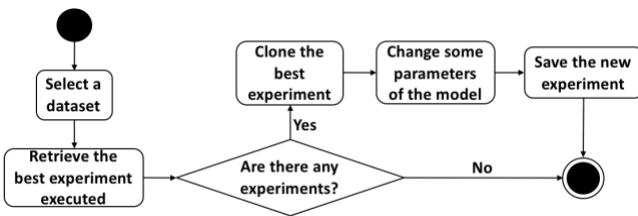


Figure 5. Optimizer Agent Activity Diagram.

To complete this task, the agent starts by selecting a dataset, because the performance and the accuracy are directly related with the dataset used in the training process. Once the dataset is selected the agent retrieves the best experiments of a given type and, based on the parameters, it generates and saves a new model. Notice here that for each type of experiment the *Optimizer Agent* was extended in order to create specific agents which selected the correct algorithm in each case. Fig. 5 describes the workflow of the *Optimizer Agent*. Observe that the *Optimizer Agent* needs a different strategy to create the new model, depending on the type of the experiment.

#### D. Optimizers

Each machine learning strategy comes with a lot of tricks and techniques to improve the performance of the model. Some of the techniques can include mathematical operations, such as transpose, reverse, etc., that can increase the dataset and have a direct impact on the performance as a result. Other techniques aim at increasing the number of features in the dataset to facilitate the training process and obtain a better model. Some examples include multiplication of numeric fields or the use of trigonometrical functions. In addition, there is a group of techniques that filter the outliers to obtain a more general model. All these approaches work directly on the dataset, but our focus here is to work with the existing datasets and calibrate the model's parameters. Each one of the techniques has its own unique parameters, so, it was necessary to create an optimizer for each one. Namely: *SVM Optimizer*, *DT Optimizer* and *NN Optimizer*. The *SVM Optimizer* takes advantage of the kernel trick [27] and creates a new model based only on the best SVM experiment executed. If the best model memorizes the dataset, it then decreases the kernel to compact the data. On the other hand, if the model's accuracy is low, then the agent increases the kernel to separate the data by adding new dimensions. The *DT Optimizer* uses a similar criterion to increase or decrease the *max\_depth* of the decision tree while the *NN Optimizer* creates a new model by randomly combining the two best experiments executed.

#### E. Details of the API

Finally, we created an Application Programming Interface (API) that contains the new objects and functionalities required to set up an environment: create, train and validate the experiments; test the results, and use the best models for prediction.



Figure 6. API Class Diagram.

Fig. 6 shows the API class diagram. The *Task* class defines a collection of experiments of the same problem and refers to the same machine learning problem. Every machine learning problem requires the analysis of data. The *Dataset* class represents a collection of data to be used and contains features such as the path in which it is stored. The data can be stored in different file formats. For this reason, each *Dataset* contains a *DatasetType* class to specify its type, such as *RData*, *CSV*, etc. An *Experiment* class represents the training process of a model and contains general variables being measured, such as time. It also contains more specific features, depending on the particular model being trained. In order to specify the types of experiments allowed to run within the platform, all the Experiments contain

an *ExperimentType* class. The Predictor class defines an object to evaluate a model and the Committee class defines a collection of Predictors and contains a parameter to set the number of members. First, to use the API, we need to select a Task to work with and after that the experiments can be created, linked to the selected task. Each Experiment has a type defined in *ExperimentType* and can have training, validation and testing datasets associated to it, respectively. Each Dataset has a type defined in *DatasetType*. Finally, to predict, based on previously trained models, there are two possible classes: (i) Predictor, which selects the best trained model based on accuracy and uses it to predict, and (ii) Committee, which has a collection of predictors and returns a consensus among them.

## V. USER SCENARIO

This section details an exploratory study taken as a proof of concept for the framework. The experiment is divided into two stages. First, we set up the environment and create the proper conditions to run the experiment — in this case, it was necessary to launch the agents’ platform, to configure the database access and to establish the initial experiment. Second, the agents start their work by training the first model and writing the results. The variables that were measured were the training and validation accuracy, as well as the start and end time. At this point, the *Optimizer Agent* analyzes the results of the finished experiments and proposes a new experiment using the same dataset.

### A. The Dataset

The data used in this example was the IRIS dataset found in the UCI Machine Learning Repository [28]. It contains 150 instances of three classes of iris plants. The predictable attribute is the type of plant, based on four other attributes: sepal length, sepal width, petal length and petal width — all the measurements are in centimeters (cm). This dataset has no missing values and two of the three types of iris are not linearly separable. Table 1 shows a brief summary of the data.

**Table 1.** Summary of the IRIS Dataset

IRIS	Sepal Length	Sepal Width	Petal Length	Petal Width
Min	4.3	2	1	0.1
Median	5.8	3	4.35	1.3
Mean	5.843	3.057	3.758	1.199
Max	7.9	4.4	6.9	2.5

### B. Results

The framework was instantiated as shown in Figure 7. The *TrainingAgent* and the *OptimizationAgent* were extended into the *SVMTrainingAgent* and the *SVMOptimizationAgent* respectively in order to implement specificities about how to train and optimize an SVM model [29]. In this case, for the optimization agent, we use a grid search approach allowing the parameter C the values 1.0, 1.5, and 2.0 and for the Degree the values 1, 2 and 3. The class *SVMExperiment* inherits for the *SpecificExperiment* hotspot and adds the parameters needed to train an SVM experiment. Finally, the *FileData* class and the *CSVData* are classes created to store a reference to the dataset. The starting point is an instance of the *SVMExperiment* class and we choose the following parameters, as shown in Table 2 (first line).

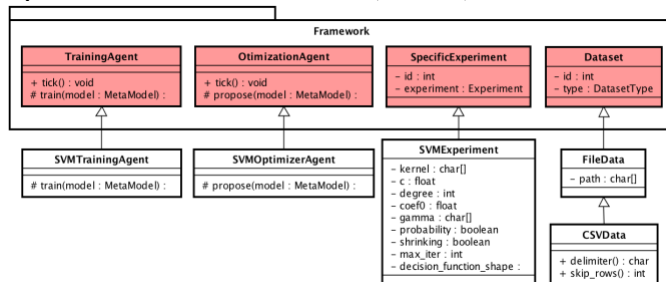


Figure 7 Framework instance for the Iris experiment

**Table 2.** Parameters of the executed experiments

Id	Kernel	C	Degree	Coef0	Gamma	Probability	Shrinking	Max Iterations	Decision Function
1	poly	1	1	0	Auto	0	1	-1	odr
2	poly	1.5	1	0	Auto	0	1	-1	odr
3	poly	1.5	2	0	Auto	0	1	-1	odr

**Table 3.** Measures of the executed experiments

Id	Started	Ended	Time (in seconds)	Validation Accuracy
1	2017-02-27 19:09:43	2017-02-27 19:09:43	0.006163	0.96
2	2017-02-27 19:09:53	2017-02-27 19:09:53	0.004834	0.96
3	2017-02-27 19:10:03	2017-02-27 19:10:03	0.005739	0.97

The training agents were essentially training the new models proposed, while the optimizer agents were trying to tune the parameters of the previously executed models and proposing new

ones that might have a good accuracy. Table 2 in rows 2 and 3 shows the experiments proposed by the *Optimizer Agent* and Table 3 shows the variables measured. The first row in table 2

shows the beginning of the second stage where only the first model had been proposed. Then, the *Trainer Agent* trained the model, resulting in an accuracy of 0.96 (first row in Table 3). The *Optimizer Agent* performed a query to retrieve the trained models and based on the best one, it modified the allowed error (parameter C in Table 2) from 1.0 to 1.5 and proposed the second model. The *Trainer Agent* realized that there was a model to train and then trained it, resulting in an accuracy of 0.96, as well. Once again, the *Optimizer Agent* modified the degree of the function to propose the third model (parameter degree in Table 2) based on the first and the second models. As a result, the *Trainer Agent* trained the new model and obtained a better accuracy of 0.97.

To obtain new models the *Optimizer Agent* balanced the allowed error and the degree of the polynomial function. It is possible to see in Table 3 that the last trained model performed better in the validation. Thus, in the next KDD phase the prediction algorithm will use the best models, based on their accuracy.

## VI. CONCLUSION AND FUTURE WORK

This paper proposes a software framework based on multiagent systems to automate the process of calibrating machine learning models and reduce the amount of human time dedicated to the parameters adjustment. By means of this framework, the agents will tune the parameters of the models while the data mining experts are focused on providing the framework with potential parameter insights. Together, the agents and the data mining experts can complement each other's capabilities in a shared software environment. We conclude that it is possible to take advantage of the characteristics of the software agents to train machine learning models, and also to make decisions about new models that might have good accuracy. The multiagent system inside the proposed solution is the core of the application because it requires autonomy to make decisions, proactivity to create new experiments, and reactivity to deal with overfitting and low accuracy. By automating this process, the users only need to set up the initial battery of experiments, which reduces the time dedicated to train a successful model.

For future work, we have two goals. First, Selection of the first model: The framework needs initial models as inputs to begin the calibration processes, but it would be interesting if the system was capable of auto-generate starting models. Second, Features Selection: Another interesting problem is how to improve the performance of the training by first selecting the most important attributes. This could significantly impact the time spent to train a model. Other possible approaches to improve performance include the use of heuristics such as Principal Features Analysis (PFA) [30] or methods, such as Sequential Forward Selection (SFS) [31] and Sequential Backward Selection (SBS) [31].

## REFERENCES

[1] J. James, Data never sleeps 2.0. 2014.  
 [2] P. Ranganathan, The data explosion. IEEE Computer Society Press, 2011.  
 [3] C. Lucena and I. Nunes, "Contributions to the emergence and consolidation of Agent-oriented Software Engineering," J. Syst. Softw., vol. 86, no. 4, pp. 890–904, Apr. 2013.  
 [4] M. E. Markiewicz and C. J. de Lucena, "Object oriented framework development," Crossroads, vol. 7, no. 4, pp. 3–9, 2001.

[5] K. M. Khalil, M. Abdel-Aziz, T. T. Nazmy, and A.-B. M. Salem, "MLIMAS: A Framework for Machine Learning in Interactive Multi-agent Systems," Procedia Comput. Sci., vol. 6, Jan. 2015.  
 [6] M. Wooldridge, An Introduction to MultiAgent Systems. John Wiley & Sons, 2009.  
 [7] E. Alonso, M. D'inverno, D. Kudenko, M. Luck, and J. Noble, "Learning in multi-agent systems," Knowl. Eng. Rev., vol. 16, no. 3, pp. 277–284, 2001.  
 [8] H. E. Nouri, O. B. Driss, and K. Ghédira, "Hybrid Metaheuristics within a Holonic Multiagent Model for the Flexible Job Shop Problem," Procedia Comput. Sci., vol. 60, pp. 83–92, Jan. 2015.  
 [9] Y. Shoham, R. Powers, and T. Grenager, "If multi-agent learning is the answer, what is the question?," Artif. Intell., vol. 171, no. 7, pp. 365–377, May 2007.  
 [10] P. Stone, "Multiagent learning is not the answer. It is the question," Artif. Intell., vol. 171, no. 7, pp. 402–405, May 2007.  
 [11] A. I. R. L. Azevedo and M. F. Santos, "Kdd, semma crisp-dm: a parallel overview," IADS-DM, 2008.  
 [12] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From Data Mining to Knowledge Discovery in Databases," AI Mag., vol. 17, Mar. 1996.  
 [13] A. Kunft, A. Alexandrov, A. Katsifodimos, and V. Markl, "Bridging the gap: towards optimization across linear and relational algebra," 2016, pp. 1–4.  
 [14] T. Kraska, A. Talwalkar, J. C. Duchi, R. Griffith, M. J. Franklin, and M. I. Jordan, "MLbase: A Distributed Machine-learning System," in CIDR, 2013, vol. 1, pp. 2–1.  
 [15] E. R. Sparks et al., "MLI: An API for distributed machine learning," in Data Mining (ICDM), IEEE 13th International Conference on, 2013..  
 [16] G. Luo, "PredicT-ML: a tool for automating machine learning model building with big clinical data," Health Inf. Sci. Syst., Dec. 2016.  
 [17] S. R. Garner and others, "Weka: The waikato environment for knowledge analysis," in Proceedings of the New Zealand computer science research students conference, 1995, pp. 57–64.  
 [18] M. Abadi et al., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," ArXiv Prepr., 2016.  
 [19] T. Green and others, "Prediction API: Every app a smart app," Google Dev. Blog Apr, vol. 21, 2011.  
 [20] J. Barnes, "Azure machine learning: Microsoft azure essentials," 2015.  
 [21] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," ACM SIGKDD Explor. Newsl., vol. 11, no. 1, pp. 10–18, 2009.  
 [22] I. Guyon et al., "Design of the 2015 ChaLearn AutoML challenge," in International Joint Conference on Neural Networks (IJCNN), 2015.  
 [23] M. Wooldridge and N. R. Jennings, "Pitfalls of Agent-oriented Development," in Proceedings of the Second International Conference on Autonomous Agents, New York, NY, USA, 1998, pp. 385–391.  
 [24] P. P.-S. Chen, "The Entity-relationship Model—Toward a Unified View of Data," ACM Trans Database Syst, vol. 1, pp. 9–36, Mar. 1976.  
 [25] R. Gentleman, R. Ihaka, D. Bates, and others, "The R project for statistical computing," R Home Web Site Httpwww R-Proj. Org, 1997.  
 [26] "Design Patterns by Gamma: Pearson India 9789332555402 Paperback - A - Z Books." [Online]. Available: <https://www.abebooks.com/Design-Patterns-Gamma-Pearson-India/17320714110/bd>. [Accessed: 12-Apr-2017].  
 [27] B. Scholkopf, "The kernel trick for distances," Adv. Neural Inf. Process. Syst., pp. 301–307, 2001.  
 [28] K. Bache and M. Lichman, "UCI machine learning repository," 2013.  
 [29] C. Cortes and V. Vapnik, "Support-vector networks," Mach. Learn., vol. 20, no. 3, pp. 273–297, Sep. 1995.  
 [30] Y. Lu, I. Cohen, X. S. Zhou, and Q. Tian, "Feature Selection Using Principal Feature Analysis," in Proceedings of the 15th ACM International Conference on Multimedia, New York, 2007.  
 [31] J. Doak, "CSE-92-18 - An Evaluation of Feature Selection Methods and Their Application to Computer Security," UC Davis Dept Comput. Sci. Tech Rep., Jan. 1992.

# Accompanying Observation Modes and Software Architecture for Autonomous Robot Software

Zhe Liu, Xinjun Mao, and Shuo Yang  
College of Computer,  
National University of Defense Technology  
Changsha, Hunan, P. R. China  
Email: {liuzhe, xjmiao, yangshuo}@nudt.edu.cn

**Abstract**—To support robust task execution in open environment, autonomous robots (AR) should include reactive capabilities to cope with the dynamics and uncertainties from real-world environment. The uncertainties pose great challenges for robots being sensitive to the environmental changes and flexible to adjust self-behaviors. To this end, this paper aims to improve the sensing and acting capabilities of autonomous robots by novel behavioral theories, observation modes and software architectures. Specifically, this paper has three main contribution: (1) presents an accompanying model that specifies a novel accompanying pattern for interacting robot behaviors; (2) proposes four types of accompanying observation modes that coordinate multiple robot sensing behaviors; (3) proposes a concrete multi-agent software architecture that implements aforementioned accompanying model and accompanying observation modes. To demonstrate the applicability and validity of our accompanying modes and MAS-based software architecture, this paper conducts a case study to implement a domestic service example, which requires the robot to run in a highly dynamic environment and can adapt its behaviors to unexpected situations.

\*

**Keywords**—Autonomous robot software; accompanying model; accompanying observation mode; multi-agent system

## I. INTRODUCTION

Nowadays, more robots are increasingly applied in open environments (e.g., family, hospital, battlefield) and expected to autonomously behave without human beings intervention to achieve assigned tasks [1], [2]. Typically, we call them as autonomous robots (AR), which is a complex software-driven system. The software (Autonomous robot software, ARS) expects to (1) manage and control physical devices (e.g., arm, motor, leg) of robots; (2) make decisions on robots behaviors and drive robots to act; (3) perform computations on robot sensing data.

In presence of dynamics and uncertainties, the autonomous robot software is challenged with the following issues: (1) no explicit behavior theories that specify interacting processes between robot sensing and acting behaviors; (2) limited observation modes that effectively coordinate various robotic sensing behaviors; (3) lack of suitable software architectures that suitable to implement the novel behavior theories.

DOI reference number: 10.18293/SEKE2018-088.

\*This research is supported by research grants from Natural Science Foundation of China under Grant No. 61532004.

In the field of the behavior of autonomous robots, there are many researches focus on behavior-based model for robot [3]–[5]. And in recent year, [6] presented a behavior-based hierarchical architecture and defined fourteen robot behaviors to assist telepresence control a humanoid robot. [7] reproduced the kind of individual recognition and attention that a human can provide in robot based on observed behavior. However, the relationship between robotic behaviors and behavior theories are not hot research directions. Sony company had simply classified and described robotic behaviors in their behavior module [8], but we think their behavior classification cannot be used for AR’s behaviors because their method didn’t meet the characteristics of AR. And for software architectures of ARS, there are several researches has combined ARS with MAS (multi-agent system). VOMAS (Virtual Operator Multi-Agent System) [9] is developed to support spontaneous generation of a task without the need to re-plan. COROS [10] is a multi-agent software framework for cooperative robots in which each of the agents represents a robot machine or a monitoring/control workstation.

This paper aims to improve the sensing and acting capabilities of autonomous robots by novel behavioral theories, observation modes and software architectures. Specifically, this paper has three main contribution: (1) presents an accompanying model that specifies a novel accompanying pattern for interacting robot behaviors. (2) proposes four types of accompanying observation modes that coordinate multiple robot sensing behaviors. (3) proposes a concrete multi-agent software architecture that implements aforementioned behavior models and observation modes.

The rest of this paper is organized as follows: Section 2 analyzes the features of autonomous robot behaviors through a motivating example and presents an accompanying model. Section 3 proposes four types of accompanying observation modes. Section 4 introduces a MAS-based architecture. Section 5 illustrates a case study and conclusion is made in Section 6.

## II. ACCOMPANYING BEHAVIOR THEORY

In this section we analyze the features of autonomous robot behaviors through a motivating example, and present the accompanying behavior model that specifies the novel behavior patterns.



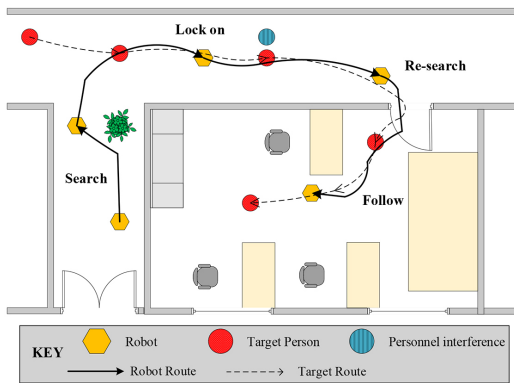


Figure 1. The motivating sample of autonomous robot

### A. A motivating sample

Let us consider a domestic service robot example that motivates our research (see Figure 1). The service robot operates in an open home environment with several rooms and is designed to care for elderly inhabitants. Specifically, the robot expects to identify whether they have fallen off and provides the necessary information services, e.g., calling an ambulance or notifying the family. As the elderly person moves around, the service robot should follow the person within a safe distance to timely perceive his moving information, determine his safety status, and answer his service requests.

- *Scenario 1 (searching for the elderly person)*: the robot should search for the target by moving through the rooms autonomously and recognizing his body features. If the target is lost, the robot repeats the search for the target.
- *Scenario 2 (following the elderly person)*: when the target is found, the robot follows the target when he moves. The robot should follow the target closely to avoid losing the target.
- *Scenario 3 (lock on to the elderly person)*: when there is someone else next to the target, the robot should lock on to the target and avoid mistakes on recognizing the target person.

Through the example, we have found some common features of the autonomous robot behaviors: (1) the robot needs to plan a rational route and timely process multi-feedback from multiple robotic behaviors; (2) the robot should cope with the environment uncertainties by adjusting its behaviors to avoid obstacles or lock on to the right target. From the concrete example, we can claim that: (1) close connection and interaction between different behaviors of robots is important for acting robustly in open environments; (2) massive sensing information is necessary for robotic behavioral adjustment and adaptation.

### B. Accompanying behaviors

For a complex task, robots usually need to carry out diverse behaviors to cooperatively reaching a common task goal. Some of the behaviors are responsible for observing the environment and sensing changes, whereas others drive physical actuator devices to achieve physical tasks. In this view, robot behaviors can be classified into two types: task behaviors that performed

by actuators (e.g., arms, legs, and motors), and observation behavior typically performed by sensors or probes (e.g., sonar and cameras).

Although the two types of behaviors are relatively independent, the robot should enhance their synergy when achieving tasks, so that task behaviors can obtain on-demand feedback to adjust, optimize and self-manage the planned behaviors. Such a synergistic relationship between task behaviors and observation behaviors was defined as *accompanying behavior*.

### C. Accompanying model for ARS

Furthermore, we found that accompanying relation doesn't only occur between observation and task behaviors, but can also occurs between observation behaviors. Multiple observation behaviors is the basis of accompanying behavior. For example, when robot follows the target, the robot should use the camera to lock on the target and use the radar to measure the distance between it and the target. There are several features in accompanying relations with observation behaviors:

- *Interactivity*: the observation and task behaviors of autonomous robots must interact with each other frequently and rapidly. The sensing information from observation behaviors allows the task behaviors informed of the status of task execution and make adaptation timely.
- *Concurrency*: when accompanying relation occurs, there are always more than one observation behaviors executing. Multiple observation behaviors can observe the environment in multi-dimension and improve the perception ability.
- *Temporality*: when multiple observation behaviors execute in parallel, the processing of massive observation information will show a kind of timing. Usually, these information will be processed sequentially. However, sometimes, one kind of information is more important for task execution and it should be given a high processing priority.
- *Dependence*: different accompanying behaviors reply on different types of information while interacting with each other. Some interactions depend on data-type messages, and others may depend on event-type messages.

Based on the accompanying behavior and the analysis above, we present an accompanying model for autonomous robot software (see Figure 2). In the model, we divide these behaviors into system-level behaviors that perform common robotic capabilities (such as planning behavior, scheduling behavior, etc), and application-level behaviors that relate to specific tasks (task behavior and observation behavior).

As shown in Figure 2, multiple observation behaviors are the core pf the accompanying model: (1) when accompanying relation occurs between observation behaviors, it can let software receive more accurate environmental information. (2) when accompanying relation occurs between task behavior and observation behaviors, it can let software know task execution more accurately and rapidly. Therefore, accompanying model have achieved two purposes for ARS: (1) On the one hand,

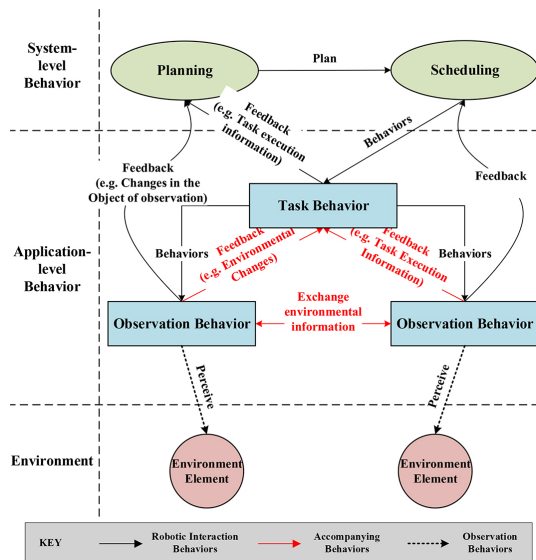


Figure 2. The accompanying model for autonomous robot software multiple observation behaviors can enhance the sensitivity of the robot to the dynamic environment by more than one sensors. The accompanying behavior can help the robot make plan more accurately by multi-feedback. (2) On the other hand, accompanying model can assist autonomous robot software adjust its behaviors and tasks more flexibly with interaction between different behaviors.

### III. ACCOMPANYING OBSERVATION MODES

This section identifies four types of accompanying modes for the autonomous robot software in runtime phase.

#### A. Classification of accompanying observation modes

As robots operate in open environments, different observation behaviors can form into diverse accompanying relations. In this section, we try to analyze the essential types of accompanying observation in following cases:

- Purpose-oriented: in aforementioned sections, there are two main purposes for accompanying relations based on multiple observation behaviors. One purpose is to improve the perception ability to the dynamic environment, while the other one is to facilitate flexibility and adaptivity for robot behaviors. From this view, we divide the accompanying observation modes into two major categories: (1) environment perception modes, (2) task assistant modes.
- Multi-dimension consideration: as illustrated in section 2, there are many characteristics for accompanying behaviors, and these characteristics of different dimensions can also result in diverse accompanying observation modes. For example, for interactivity, sometimes observation behavior needs to interact with other observation behaviors and sometimes neednt, considering this we design a cooperation mode for the former situation.

Based on aforementioned classification principles, we have identified four types of accompanying observation modes: *accident mode*, *observation behaviors cooperation mode*, *priority*

*mode*, *non-priority mode*. And the characteristics of these modes in multiple dimensions as show in Table I. In Table I, **Interactivity** means the types of behavior which interact with each other in the mode; **Concurrency** means whether there are multiple observation behaviors occurs at the same time; **Temporality** means whether there are behaviors with high processing priority; **Dependence** means the type of message between behavioral accompanying in the mode.

TABLE I.  
THE CHARACTERISTICS OF ACCOMPANYING OBSERVATION MODES IN MULTIPLE DIMENSIONS.

Dimension	Environment Perception Modes		Task Assistant Modes	
	AM	OM	PM	NPM
<b>Interactivity</b>	OB, TBs and SBs	OB and OB	OBs and TB	OBs and TB
<b>Concurrency</b>	No	Yes	Yes	Yes
<b>Temporality</b>	No	No	Yes	No
<b>Dependence</b>	Event-type	Data-type	Data-type	Data-type

<sup>1</sup> AM=Accident Mode, OM=Observation behaviors cooperation Mode, PM=Priority Mode, NPM=Non-Priority Mode

<sup>2</sup> OB=Observation Behavior, TB=Task Behavior, SB=System-level Behavior

#### B. Four types of accompanying observation modes

1) *Accident mode*: This mode is designed for some emergencies in runtime for software operation safety, which is a kind of event-type message dependence mode between one observation behavior, task behaviors and system-level behaviors. This mode can improve robotic emergency response to sudden changes from the environment. Autonomous robot software usually operates in a complex environment where many unexpected statuses may occur. These statuses sometimes have side effects on robotic task execution, and the software may not open all sensors to these uncertain statuses all the time.

For possible accidents from the environment, the software expects to plan corresponding observation behavior to detect them. When accident occurs, the observation behavior will send the event-type message to system-level behaviors. Then, the software will let the robot block current execution task and switch to pre-designed corresponding task to deal with the accident. As soon as the accident is removed, the software will let robot resume the task.

As described in the motivating example, when the depth sensor finds that there is an obstacle in front of the robot, it will send event-type message “Obstacle” to planning behavior, and the software will let robot stop to avoid the obstacle until there are no obstacle.

2) *Observation behaviors cooperation mode*: This mode is designed for the interaction between observation behaviors in runtime. The active interaction can help observation behaviors adjust themselves rapidly and on this basis robot can receive environmental information with more accuracy and efficiency. This mode is a kind of data-type message dependence mode.

Under observation behaviors cooperation mode, when the software needs multi-observation behaviors to observe a specific target or environment, observation behaviors can interact with each other to adjust their observation strategies.

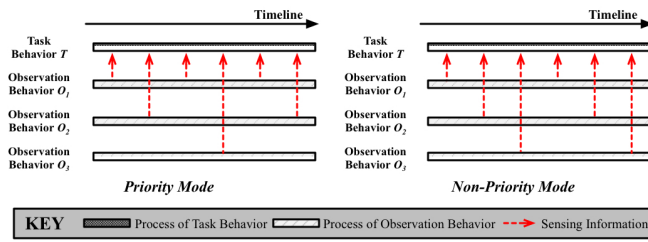


Figure 3. The processing sequence diagrams of Priority mode and Non-priority mode

3) *Priority mode*: This mode is designed for observation behaviors with different priorities. This mode is a kind of data-type message dependence mode between one task behavior and multi-observation behaviors. Generally speaking, different priorities mean different processing orders. In some tasks, the software should give some sensing information a higher processing priority than others. In other words, the software will process these sensing information more frequently and rapidly.

Under priority mode, for specific tasks, the processing priority of observation behaviors will be predefined in task behaviors or assigned by planning behavior. After collecting the sensing information, task behaviors will process these information according to their processing priorities.

4) *Non-priority mode*: This mode is designed for behaviors with the same processing priority and their information will be fused by task behaviors. This is a kind of data-type message dependence mode between one task behavior and multi-observation behaviors.

Under non-priority mode, observation behaviors have the same processing priority. The task behavior just need to receive the sensing information sequentially and process them as predefined according to tasks.

Figure 3 shows processing sequence of the same series of behaviors under priority mode and non-priority mode. In Figure 3, under priority mode, observation  $O_1$  has a higher processing priority than  $O_2$  and  $O_3$ . And under non-priority mode, all observation behaviors have the same processing priority and are processed sequentially.

#### IV. SOFTWARE ARCHITECTURE AND IMPLEMENTATION APPROACH OF ACCOMPANYING OBSERVATION MODES

##### A. MAS-based software architecture

The Multi-Agent System (MAS) provides an effective solution to address the development issues of high-level autonomous robot software for both modeling and implementation aspects. The autonomous robot software is a complex system that consists of a number of diverse and interacting components. All of these components work together to achieve the robots design objectives. Therefore, the autonomous robot software can be decomposed and organized as multiple autonomous agent entities. Furthermore, MAS maintains strong concurrency of plan execution, flexible interactions among diverse agent behaviors, and strong robustness of system functionalities. These features of MAS can greatly benefit

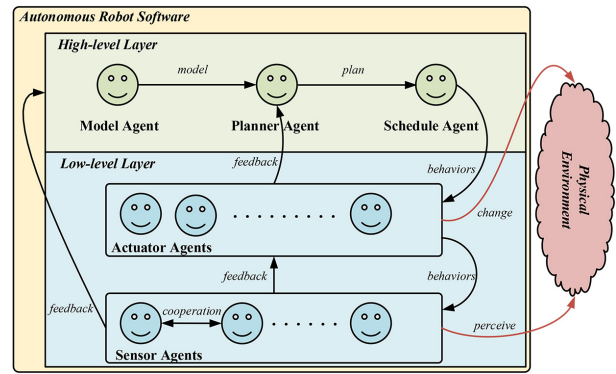


Figure 4. MAS-based software architecture for ARS

the implementation of accompanying model and observation modes, which require both concurrency and interaction mechanisms from implementation software architecture.

In this paper, we propose a multi-agent software architecture to implement the autonomous robot software and provide support for accompanying modes (Figure 4). In the architecture, each agent plays different role, takes distinct behaviours, and cooperates with each other to achieve the common task goals. Integrating with our previous works for the dual-loop control model [11], the multi-agent software architecture provides support for accompanying model and observation modes through agent communication mechanisms. However, in this MAS-based architecture, we don't consider the non-deterministic characteristic of MAS such as learning [12], self-adaptation and self-organization [13]. The description of the role that each agent plays in this multi-agent model is as follows:

- The *modeler agent* establishes the world model on the sensor inputs from sensor agents and offers the specifications of planning domain and problem to the planner agent for task planning.
- The *planner agent* performs planning jobs, including activities of establishing world models and planning over a specific problem domain.
- The *schedule agent* acts as a mediator that dispatches the generated plans to a specific actuator agent capable of the corresponding action.
- An *actuator agent* is implemented as the abstraction over the robot physical actuator and maintains a simple reactive structure, which will dispatche sensing tasks to sensor agents.
- A *sensor agent* controls an independent sensor device of the robot and is implemented to perform a stimulus-response behaviour aimed towards external state changes.

##### B. Implementation of accompanying observation modes

As shown in Figure 4, in our multi-agent software architecture, sensor agents (observation behaviors) can feedback sensing information to actuator agents (task behaviors) and high-level layer agents (system-level behaviors). Besides, sensor agents can interact with each other. Furthermore, multiple agents can be executed concurrently. These designs for the

architecture guarantee the implementation of accompanying modes.

In the implementation of this MAS-based architecture, we developed a multi-agent software framework *AutoRobot* [16] for developing autonomous robot software applications. In *AutoRobot*, MAS-based software architecture can be implemented under JADE [14] and robot controllers can be implemented under ROS [15]. For accompanying implementation, JADE has provided three communication behaviors for agents: One-shot, Cyclic and ThreeStep. Besides, we also design two interaction mechanisms for agents' communication: topics-based mechanism and services-based mechanism. These above communication behaviors and mechanisms guarantee the implementation of the accompanying modes.

## V. CASE STUDY

In this section, we implement the motivating example to validate the effectiveness and applicability of accompanying model and observation mode.

The hardware platform we adopted in the case study is the Turtlebot2 mobile robot, including a mobile base and a Kinect module that contains an RGB camera and a depth camera. The software infrastructure consists of the *AutoRobot* server and the ROS server.

### A. Implementation of the motivating example

The case study for the aforementioned example is implemented using a multi-agent system prototype from the *AutoRobot* framework, and the concrete architecture for the case study is illustrated in Figure 5.

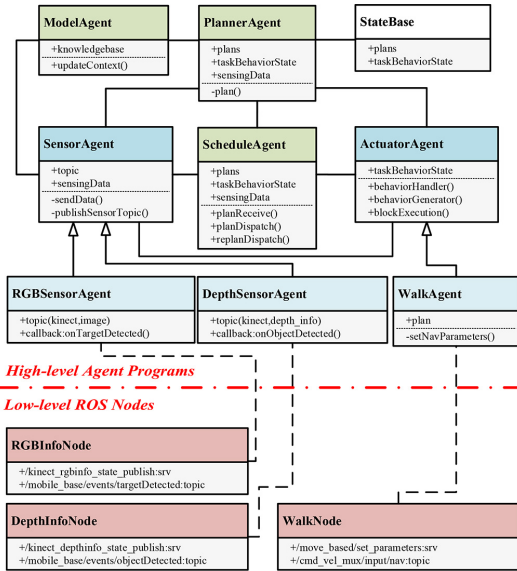


Figure 5. The multi-agent implementation architecture of the case study

For this specific example and the Turtlebot2 capability, we designed the following agent roles: 1) a WalkAgent to drive the mobile base to move around, 2) an RGBSensorAgent and DepthSensorAgent to obtain the RGB and depth images from the RGB and depth cameras.

Moreover, we implement the low-level robot controllers into ROS nodes that offer the fundamental robot functionalities. Each of the ROS node programs, such as the RGBInfoNode, corresponds to a high-level agent entity used to fulfil the agent's capability. The ROS nodes communicate with the high-level agent programs through the Java nodes that implemented by RosBridge middleware.

Under the implementation architecture, we implemented the service scenarios of *searching*, *following* and *locking on* the elderly inhabitants. In each scenario, the agent roles involved in the scenario include all agents in Figure 5. Figure 6 shows the actual effect of the implementation of these scenarios in human's view and robot's view.

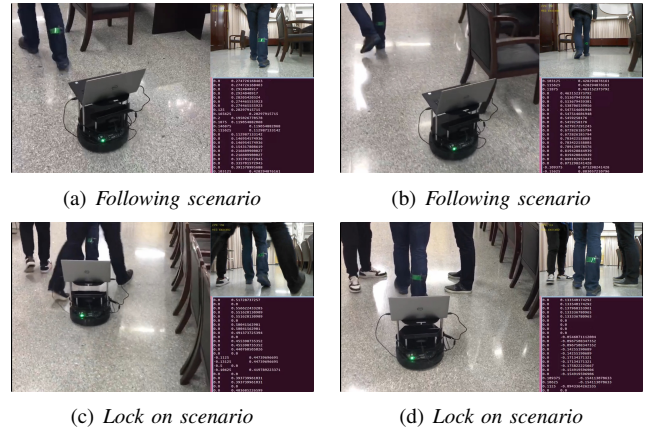


Figure 6. The implementation of scenarios in real world. In (a) and (b) robot follows the target and avoids a chair. In (c) and (d) robot locks on the correct target when other people is surrounding.

### B. Implementation of accompanying model and accompanying observation mode

The implementation details of accompanying model is shown in Figure 7 which is the sequence diagram of agent collaboration in the *Following* and *Lock-on* scenario. The accompany behavior is occurs between WalkAgent, RGBSensorAgent, DepthSensorAgent.

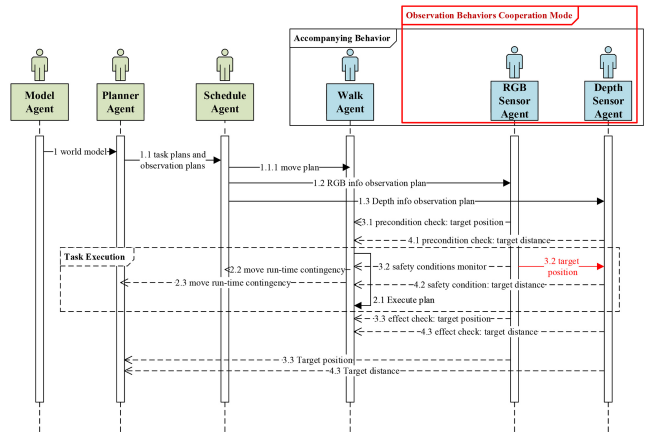


Figure 7. The sequence diagram of agent collaboration based on accompanying model in the *Following* and *Lock on* scenario

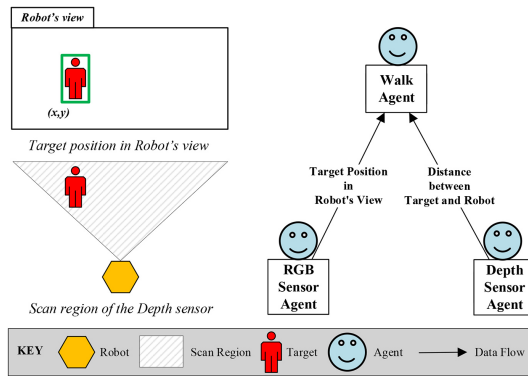


Figure 8. The scanning way of Depth sensor and the cooperation between agents in *Follow Scenario* in traditional way

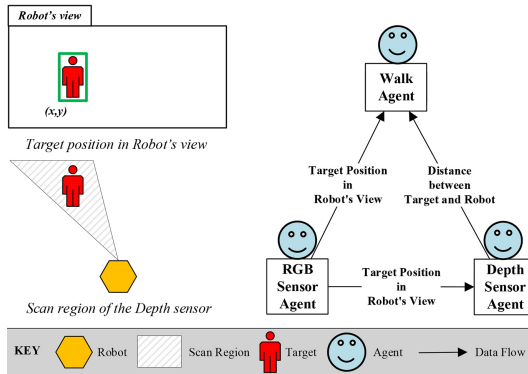


Figure 9. The scanning way of Depth sensor and the cooperation between agents in *Follow Scenario* with observation behaviors cooperation mode

The implementation details of one accompanying observation mode, observation behaviors cooperation mode, is shown in Figure 8 and Figure 9. In our case study, when robot is following the target, the software will open the observation behaviors cooperation mode between RGBSensorAgent and DepthSensorAgent. In *Following Scenario*, RGBSensorAgent is responsible to determine the position of the target in robot's view, while DepthSensorAgent is responsible to determine the distance between robot and the target. In a traditional way, RGBSensorAgent and DepthSensorAgent will feedback different sensing information to WalkAgent. Then WalkAgent will handle this information to calculate the location of the target and adjust the moving parameters. In real-world environments, the target may appear in any position in front of the robot and the DepthSensorAgent should scan a very large region to avoid losing of the target such as illustrated in Figure 8. The process of scanning and calculating the distance will cost lot of computing resource and spend more time than RGBSensorAgent, which will cause the robot losing the target.

When under observation behaviors cooperation mode, firstly, RGBSensorAgent will inform DepthSensorAgent where the target is to reduce the scan region as illustrated in Figure 9. And then RGBSensorAgent and DepthSensorAgent will send sensing information to WalkAgent. In this way, the processing time for the target location is greatly reduced and the sensitivity of the robot to the location of the target is greatly improved.

## VI. CONCLUSION

This paper presents an accompanying model and four types of accompanying observation modes for autonomous robot software to solve the challenge about being sensitive to the environmental changes and flexible to adjust self-behaviors. Our contributions are threefold. First, we investigate the features of behaviors in autonomous robot software and present an accompanying model based on accompanying relations. Second, we identify a number of important accompanying observation modes. Third, we propose a MAS-based software architecture to examine and specify the accompanying observation modes.

## REFERENCES

- [1] Mao, X. J. (2016). Autonomous Robot Software Technologies, *Communications of the CCF*, 12(7): 60-65.
- [2] Ziafati, P. (2013). Programming autonomous robots using agent programming languages. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems* (pp. 1463-1464). International Foundation for Autonomous Agents and Multiagent Systems.
- [3] Ringert, J. O., Rumpel, B., and Wortmann, A. (2014). A requirements modeling language for the component behavior of cyber physical robotics systems. *arXiv preprint arXiv:1409.0394*.
- [4] Topalidou-Kyniazopoulou, A., Spanoudakis, N. I., and Lagoudakis, M. G. (2013). A CASE tool for robot behavior development. In *RoboCup 2012: Robot Soccer World Cup XVI* (pp. 225-236). Springer, Berlin, Heidelberg.
- [5] Liu, C., and Tomizuka, M. (2017). Designing the Robot Behavior for Safe HumanRobot Interactions. In *Trends in Control and Decision-Making for HumanRobot Collaboration Systems* (pp. 241-270). Springer, Cham.
- [6] Zhao, J., Li, W., Mao, X., Hu, H., Niu, L., and Chen, G. (2017). Behavior-Based SSVEP Hierarchical Architecture for Telepresence Control of Humanoid Robot to Achieve Full-Body Movement. *IEEE Transactions on Cognitive and Developmental Systems*, 9(2), 197-209.
- [7] Glas, D. F., Wada, K., Shiomi, M., Kanda, T., Ishiguro, H., and Hagita, N. (2017). Personal Greetings: Personalizing Robot Utterances Based on Novelty of Observed Behavior. *International Journal of Social Robotics*, 9(2), 181-198.
- [8] Hoshino, Y., Takagi, T., Di Profio, U., and Fujita, M. (2004). Behavior description and control using behavior module for personal robot. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on* (Vol. 4, pp. 4165-4171). IEEE.
- [9] Hsu, H. C. H., and Liu, A. (2007). A flexible architecture for navigation control of a mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 37(3), 310-318.
- [10] Koubâa, A., Sriti, M. F., Bennaceur, H., Ammar, A., Javed, Y., Alajlan, M., ... and Shakshuki, E. (2015). COROS: a multi-agent software architecture for cooperative and autonomous service robots. In *Cooperative Robots and Sensor Networks 2015* (pp. 3-30). Springer International Publishing.
- [11] Liu, Z., Mao, X., and Yang, S. (2017). A Dual-Loop Control Model and Software Framework for Autonomous Robot Software. In *Asia-Pacific Software Engineering Conference (APSEC), 2017 24th* (pp. 229-238). IEEE.
- [12] Briot, J. P., de Nascimento, N. M., and de Lucena, C. J. P. (2016). A multi-agent architecture for quantified fruits: Design and experience. In *28th International Conference on Software Engineering & Knowledge Engineering (SEKE'2016)* (pp. 369-374). SEKE/Knowledge Systems Institute, PA, USA.
- [13] do Nascimento, N. M., and de Lucena, C. J. P. (2017). FloT: An agent-based framework for self-adaptive and self-organizing applications based on the Internet of Things. *Information Sciences*, 378, 161-176.
- [14] Bellifemine, F., Caire, G., Poggi, A., and Rimassa, G. (2008). JADE: A software framework for developing multi-agent applications. Lessons learned. *Information and Software Technology*, 50(1-2), 10-21.
- [15] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., ... and Ng, A. Y. (2009, May). ROS: an open-source Robot Operating System. In *ICRA workshop on open source software* (Vol. 3, No. 3.2, p. 5).
- [16] Yang, S., Mao, X., Yang, S., and Liu, Z. (2017). Towards a hybrid software architecture and multi-agent approach for autonomous robot software. *International Journal of Advanced Robotic Systems*, 14(4), 1729881417716088.

# An Architecture for the Development of Ambient Intelligence Systems Managed by Embedded Agents

Carlos Eduardo Pantoja  
CEFET/RJ  
Universidade Federal Fluminense  
e-mail: pantoja@cefet-rj.br

Heder Dorneles Soares and  
José Viterbo  
Universidade Federal Fluminense  
e-mail: hdorneles,viterbo@ic.uff.br

Amal El-Fallah Seghrouchni  
Sorbonne Universités  
UPMC Univ Paris 06, LIP6  
e-mail: amal.elfallah@lip6.fr

**Abstract**—Ubiquitous systems consider the use of electronic components for enhancing daily objects with some kind of computational intelligence for aiding users in their tasks pervasively. Ambient Intelligence (AmI) is a branch of ubiquitous computing that provides an environment full of interconnected devices and it can provide data communication, inference mechanism based on context information and collaboration among system's devices. Similarly, the Internet of Things (IoT) provides uniquely identified devices or things in a network for helping users in their activities. Multi-Agent Systems (MAS) are intelligent systems where agents are responsible for reasoning, competing and using resources to achieve desirable goals pro-actively and autonomously. Agents have been employed in some approaches and works during the last years, but none of them considered embedded MAS responsible for smart devices in an AmI system running over an IoT network. Besides, it is also interesting that agents of the embedded MAS can interact, sharing information with agents situated in another embedded MAS using the IoT network to learn from user's experiences. This paper proposes an architecture for the development of AmI systems using embedded MAS for interfacing with sensors and actuators in a heterogeneous network using an IoT middleware.

## I. INTRODUCTION

Ubiquitous Computing or pervasive computing is the capability of embedding intelligence in everyday objects in a way that the person who interacts with this object reduces the level of interaction with the device or even does not notice it [1]. The tendency of Ubiquitous Systems, supported by the advances in communication and network interconnection, is to allow everyday objects to interact with humans pervasively and to communicate with each other [2]. There are several fields and applications that will be impacted by the use of the Internet of Things (IoT), such as Ambient Intelligence (AmI).

Some questions about the development of AmI solutions refer to technologies that are necessary in the process of communication between devices, node synchronization, collecting and storing context data and inferences for decision making. In general, the objects that participate in this system are usually sensors or embedded devices, which have several limitations regarding hardware resources and the communication capability [3]. In an AmI system where the number of devices can grow exponentially, the concern with scalability becomes latent. To address these limitations, there is

*ContextNet* middleware [4], which treats the communication and connectivity of Mobile Nodes (MN) in a scalable way using data distribution protocols based on the DDS standard of OMG [5].

Intelligent agents are entities that can be constructed in both hardware and software and they are able of performing actions in certain environments autonomously and pro-actively. A Multi-Agent System (MAS) is composed of intelligent agents capable of communicating and collaborating — or even competing — for using resources in an environment to achieve conflicting or common goals [6]. The use of the MAS approach applied in AmI is justified by the autonomous characteristics of agents and their application in complex systems, both found in AmI [7]. However, some traditional smart objects are mainly data gathers and senders, and the data is stored and processed in servers, compromising the autonomy of these objects and because of the high dependency on centralized technologies to provide communication and reasoning in such kind of system [8]. The idea of decentralized MAS responsible for cognitive intelligence in distributed computing is being discussed instead of using centralized MAS for creating real autonomous smart objects [9].

This paper proposes an architecture for developing intelligent systems integrating devices using embedded MAS as smart objects for IoT to be used in the AmI domain. In this architecture, it is possible to develop embedded MAS for controlling devices composed of sensors and actuators. Besides, some characteristics of these nodes are: the ability to communicate with other devices apart of the technology employed in them; truly autonomous; and resilient from the IoT network. For this, our smart objects use Jason framework [10] adopting a specific kind of agent able of communicating with others devices using internally the *ContextNet*. A case study will be presented in a laboratory with several devices employing MAS, Java and Android applications. The contributions of this work are: an architecture for developing solutions for IoT and AmI using different devices supported by intelligent agents; and an extension of a well-known Agent-Oriented Programming Language (AOPL) for programming intelligent devices.

This work is structured as follows: in section 2 is presented some necessary concepts for the understanding of the proposal of this paper; section 3 presents the proposed architecture, a

extension of Jason framework integrating the *ContextNet*; in section 4 it is presented a case study e some experiments; in section 5 we discuss some related works and; finally, the conclusions and references are presented.

## II. BACKGROUND

In this section, it is presented the middleware *ContextNet* for IoT, which is responsible for communicating and gathering data from several nodes in a network. Next, some characteristics of the Jason framework [10] is discussed once it is used to develop agent-oriented nodes in the proposed architecture.

### A. The Middleware *ContextNet*

The *ContextNet* is a service for providing context data in stationary and mobile networks. It counts with a Scalable Data Distribution Layer (SDDL) layer [4], which is used for tracking applications in vehicles, industrial automation and data spreading. This service deals with major questions in data communication such as fault tolerance, network load balancing, support for node disconnection, and security. It also provides creation resources and dynamic management for groups. The *ContextNet* middleware works in a *publish/subscriber* model. The data transferring occurs by using two protocols: the MR-UDP [11] and the OMG DDS [5]. The MR-UDP treats messages between a client and a gateway; the DDS is responsible for distributing data in the core of the network. By using *ContextNet*, it is possible to enable the growing of a network, ensuring the scalability of the content distribution between millions of devices.

### B. Framework Jason

The Jason is a *framework* for developing MAS using a cognitive model named *Belief-Desire-Intention* (BDI) [12] and has an interpreter for the BDI based AOPL *AgentSpeak* in Java language. The BDI allows agents to reason based on perceptions captured of the environment, beliefs that represent the knowledge obtained during their existence and they are able of communicating with each other in order to exchange information or achieve mutual or conflicting goals. Besides, they can have plans composed of actions that are activated depending on beliefs on their belief bases.

Specifically, a Jason's standard agent has a reasoning cycle [10] responsible for the processing of all perceptions and beliefs to generate events which activate plans and actions. It is important to understand the reasoning cycle of a standard agent because several extensions (including a proposal in this paper) modify some characteristics of this reasoning cycle to enhance specific kind of agents with new customized abilities. First, the agent captures the perceptions from a simulated environment where agents can interact with virtual objects that may have information represented as perceptions. The agent verifies its mailbox at the beginning of each cycle for existing messages to be read. It is important to remark that the original distribution of Jason does not have any access to real environments. Afterwards, a function updates the *Belief Base* using the captured perceptions from the environment. For each

modification in the *Belief Base* it is generated an event that an agent has to deal with to achieve its goals. Then, an event is selected from a list of generated events and when it is selected, it retrieves all the relevant plans of the agent's plan library. After that, a verification is performed to identify which plans can be executed based on its current beliefs and perceptions and a function selects only one plan to be executed. Finally, an action of the selected plan is executed one at a time.

There is an extension of Jason's agents capable of controlling devices such as sensors and actuators connected to microcontrollers named ARGO [13]. This customized architecture is able of capturing perceptions and send them to agents without any interference of the programmer and agents are able of executing actions using actuators without worrying about what kind of hardware is being employed. The hardware and software layers are uncoupled. The reasoning cycle of an ARGO agent uses the Javino [14] for capturing perceptions. The Javino provides a serial communication between microcontrollers and Jason using a basic protocol to guarantee the correct information exchange between the transmitter and the receiver.

An ARGO agent has the abilities, at runtime, of: selecting the microcontroller which it desires to control; deciding whether or not to block the perceptions coming from sensors releasing processing time for other tasks, for example; filtering undesirable perceptions; and limiting the time interval of perceiving the environment. However, ARGO agents only communicate with agents hosted in its MAS. This implies that if a MAS is embedded into a device, the communication will be limited to this device. Therefore, in this paper we propose a new kind of agent that is able of communicating with other agents (and devices) using Jason and *ContextNet* to be used in an architecture for programming pervasive solutions in AmI.

## III. THE OVERALL ARCHITECTURE FOR AMI SYSTEMS USING MAS

In this section, it is presented our architecture for developing AmI systems using real autonomous and embedded MAS as devices. The architecture considers an open and dynamic environment where devices using the agent approach and others devices can enter or leave anytime. This approach leads us to a decentralized and a collective reasoning since every embedded MAS is considered an autonomous and independent thing capable of communicating and negotiating — or even acting as a group pursuing common goals — with others devices. By independent, it means that a smart object embedded with MAS are able to keep running and reasoning even if communication and interaction technologies stop. In fact, we consider this embedded smart object a MAS as a Thing.

Basically, the AmI system uses the *ContextNet* as IoT middleware where every device should connect as a Mobile Node (MN) to be part of the system and to communicate with others devices. Some of these devices can be MN with sensors and actuators, Android devices or electronic smart objects (tv, refrigerator, etc.). In this approach, we assert that MN can employ embedded MAS for managing sensors and actuators

in the AmI systems. This MN is composed by a cognitive layer using Jason framework with agents responsible for interfacing with hardware using ARGO; and agents responsible for communication in the network using an instance of *ContextNet* nodes. The hardware layer is composed by: the platform for embedding the MAS, which could be any tiny computer such as Raspberry Pi; sensors and actuators; and microcontrollers. Figure 1 depicts an overview of the proposed architecture.

In some cases, the use of MAS can bring advantages compared to MN that only work as data repeaters sending information from sensors to a server for discovering context about a situation and from MN which need stimulus from other devices to act upon the environment. Agents are pro-active, autonomous and are capable of reasoning about information from the environment that they are situated. These characteristics allow an improved information or even a previous context discovery before sending it to a server, releasing its processing power, for example. Besides, agents can make decisions and act autonomously without depending of a third part processing (if they have sufficient processing power).

Another important characteristic is that a MAS can be programmed individually as a MN that is able to interact with other devices (including another MAS) using a special kind of agent named **Communicator**, which has an instance of *ContextNet*. There is another type of agent responsible for controlling sensors and actuators that can be used along with the **Communicator** one. Therefore, it can exist four types of agents in a project:

- **Standard**: the standard agent is able to communicate with others agents of its MAS but it is not possible to communicate with agents from a different MAS and it is not able of controlling any kind of hardware. It is the basic unit of a MAS.
- **Argo**: it is a customized architecture of agents capable of controlling microcontrollers independently of its type and the domain applied in the solution. ARGO agents have all the abilities of a standard agent but are not able of communication with agents from a different MAS.
- **Communicator**: this agent is able of communicating with agents from a different MAS or any device using *ContextNet*. It has the same abilities of a standard agent but but it is not able of controlling hardware devices.

The *ContextNet* is a scalable architecture, which guarantees a great number of devices transmitting data at same time. In this approach, we propose the use of MAS developed using Jason to exploit some advantages of using a robust middleware for IoT applications and a well-know framework for agents solutions. Jason already counts with implementations of **Standard** and **ARGO** agents. Then, we propose an extension of Jason framework creating a customized architecture for agent **Communicator** with a *ContextNet* instance embedded into its implementation in order to facilitate the use of such kind of agent. If some device needs to communicate with a MN with a MAS, it should send messages in KQML format and translate the received KQML messages. It is important to

remark that Jason uses KQML as communication language. We also propose a mechanism for processing messages based on KQML performatives in the following section.

#### A. Extending Jason using an IoT Middleware

In order to allow the programming of agents that are able of communicating through IoT, a special kind of agent named **Communicator** was proposed. This agent is responsible for exchanging messages between agents hosted in different MAS or in any other device. For this, all the devices must be connected to the *ContextNet* middleware and the **Communicator** agent must have a communication mechanism for sending and receiving messages through the IoT. Thus, the reasoning cycle of the **Communicator** agent was extended with the *ContextNet* embedded in its architecture (Figure 2).

The first modification happened in the beginning of the reasoning cycle and it is capable of receiving messages from others devices using *ContextNet* and messages coming from agents of its own MAS using the *checkMail* method. All messages received are processed to generate events and update the agent's *Belief Base*. The next modification was inserted at the end of the reasoning cycle after the *sendMsg* step. In this moment, the agent can send a message to agents hosted in its MAS or to another device in IoT using *ContextNet*.

A message can be sent to another **Communicator** agent or any device able of understanding the message format. Every agent must have a unique identification number provided by *ContextNet* and to send any message, the agent uses an internal action named **sendOut** likewise the original internal action **send** from Jason. Both of them send a message to an addressee using a illocutionary force. The major difference between them is that **sendOut** sends a message to a mobile device or a **Communicator** agent in another MAS. In this paper, the available illocutionary forces are:

- **achieve**: sends a goal to be accomplished by the addressee. The content of the message sent will be inserted in the base of intentions of this agent.
- **unachieve**: drops a goal in case it has not been reached yet. The content of the message will be removed from the base of intentions of the addressee.
- **tell**: sends a belief of the sender that the addressee believes to be true. The content of the message must be a literal, which represents a belief and will be inserted into the belief base of the addressee.
- **untell**: the sender agent informs the addressee agent that the belief is no longer to be believed. The content of the message is removed from the belief base of the addressee.

In order to integrate *ContextNet* into Jason architecture, some modifications were performed. First of all, the *Communicator* class for creating an agent with the ability of communicating was added as an agent's customized architecture. This class has an attribute *commBridge*, which is responsible for sending and receiving messages from *ContextNet*. This class also has a function for adding the received message from *ContextNet* to the agent's mail box. The *commBridge* implements a process for mounting and verifying a message to



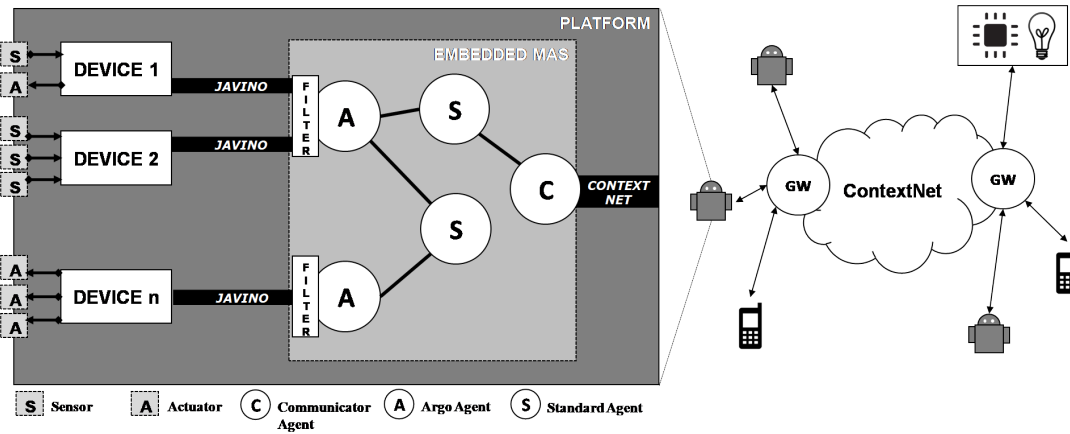


Fig. 1. An overview of the proposed architecture.

guarantee no losses of data in the communication. A message is composed of the following fields: a pre-amble to identify the origin of the message with a length of 4 bytes; fields to identify the sender and the receiver of the message with 32 bytes each; the identification of the illocutionary force with 32 bytes; and the message content with 256 bytes.

When the sender starts sending the message, the size of all fields are calculated to identify the beginning and the end of each field. After that, the pre-amble is added at the beginning of the message to verify the origin of the message. The message is mounted adding all the fields in a single string message that is sent by *ContextNet*. When the receiver receives the message, the pre-amble is verified to guarantee the origin of the message. Then, all the fields' size is verified to guarantee no losses in the communication process. If everything is ok, the message is mounted and processed as a Jason's message. Otherwise, the message is discarded.

The native *TransitionSystem* class of Jason was modified in the reasoning cycle function for allow to check if exist messages to be read coming from *ContextNet*. The existing messages are added to the mail box of the agent to be processed as beliefs or intentions depending on the illocutionary force related to the message as explained before. After that, the agent can send a message using an internal action named *sendOut*, which uses the *commBridge* to send a message using the *ContextNet*. Another internal action named *setMyCommId* is responsible for setting the identification string used by *ContextNet* to identify uniquely a device in the IoT. It is important to remark that all modifications proposed do not interfere in the original Jason distribution nor in ARGO.

#### IV. CASE STUDY

In this section, we present an initial study case based on the proposed architecture using the Jason and the *ContextNet* middleware in order to control some functionalities in a laboratory. The following scenario explicit the general behavior of our approach: Kate is the head of the laboratory and she wants to get information in her smartphone about some features such as the temperature, luminosity and the status of the lights (on

or off). She also wants to know if there are students using the laboratory while she is away receiving an updated list every time she requests. Only students which have an authorization can enter in the laboratory. So, the student's will request access to the laboratory and use its smartphone. Besides, every student when entering the laboratory should connect to the network to inform of his or her presence.

The laboratory is equipped with some devices managed by MAS and MN using Java language and Android. The temperature and luminosity sensors (LM35 and LDR respectively) are controlled by a MAS using a Galileo Intel Gen 2. For controlling status and activation of lights, a MAS running in Raspberry Pi Zero and an Arduino UNO assembled with an ATMEGA328 microcontroller were employed. The electrical installation of the laboratory was modified to accept commands coming from the microcontroller. The door is controlled by a MAS embedded into a Raspberry Pi and is responsible for registering the students entrance and to verify if they have the permissions to enter the laboratory accessing a remote database. The Android applications were developed implementing Android MN from *ContextNet*.

Every device has only one MAS embedded in a tiny computer such as Raspberry and Galileo since they have enough processing power to host embedded MAS. ARGO agents are responsible for controlling actuators and sensors that are connected to one or more microcontrollers. In our case, we assembled an Arduino UNO board, an ATMEGA microcontroller, and the Galileo GEN's GPIO. The **ARGO** agents send serial messages to the microcontroller to some action to be performed in the laboratory. On the other hand, every MAS can employ **Communicator** agents. It is important to remark that every MAS should have only one **Communicator** agent, because it will be responsible for the identification in the IoT network of the device; to communicate with other agents; and to send and receive data from *ContextNet* gateway.

The MAS responsible for controlling the temperature and luminosity sensors are composed of one **ARGO** agent and one **Communicator** agent. The former one is responsible

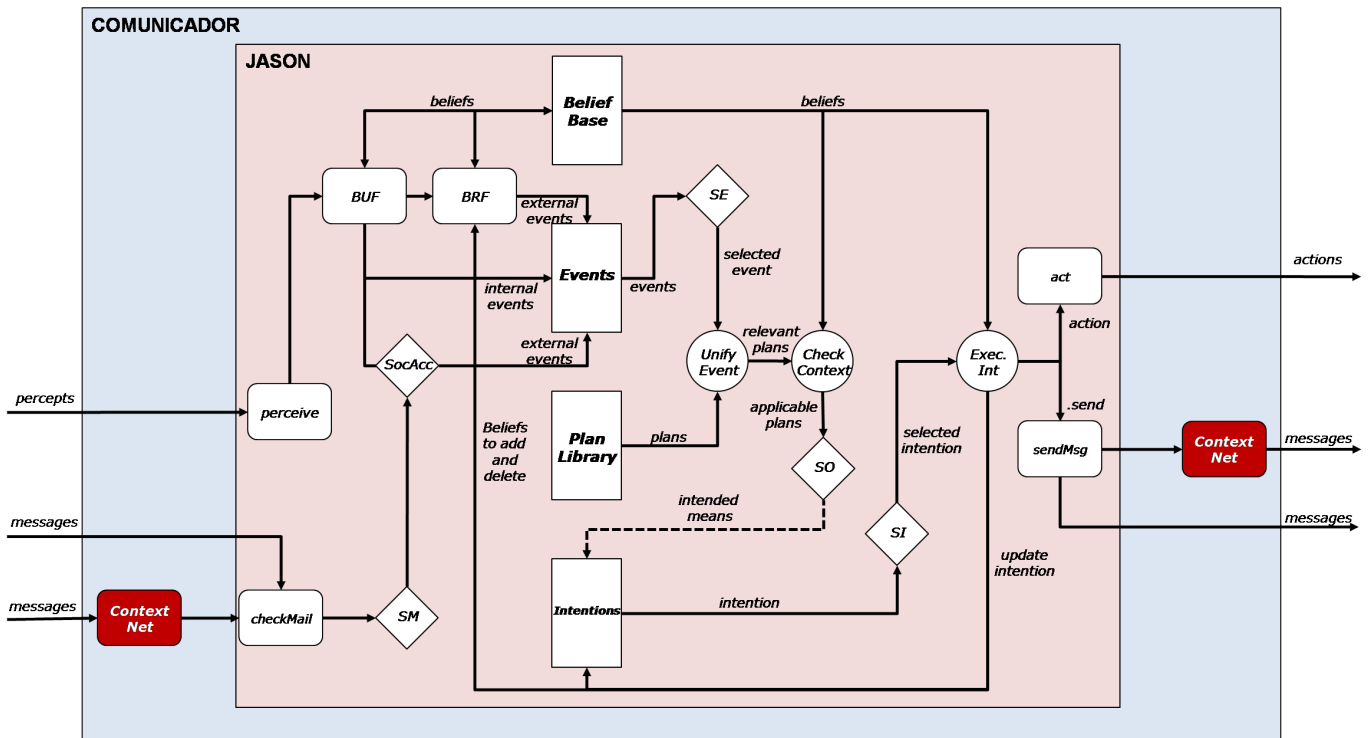


Fig. 2. The Reasoning Cycle of a Communicator agent.

for capturing the perceptions from sensors and to transmit them to the later one, which manages the received requisitions from other smart devices and respond to them with the most current perceptions using *ContextNet*. The MAS responsible for controlling the door is composed of three agents: **ARGO**, **Communicator**, and **Standard**. The **ARGO** agent is able of opening the door if it is closed and for retrieving the status of the door (opened or closed) when requested. The **Communicator** receives the requisitions from other devices and it sends to the **Standard** agent the requisitions that need to be verified if the user has access permission using a customized internal action for accessing the database. If the user has the access permission, the agent sends a message to the **ARGO** agent for opening the door.

The MAS that controls the activation of lights and other electrical stuff has an **ARGO** and a **Communicator** agent. The former one interfaces with the hardware that controls the electrical installation, and it is capable of retrieving perceptions about the status of the lights or it receives commands for activating resources (light, air-conditioning, etc.). The **Communicator** just manages requests coming from other devices like the previous examples. The Android application for the head of the laboratory employs functionalities for listing every access registered in the laboratory; status of electrical stuff and door; commands for activation and deactivation; and some administrator function such as registering a new user and setting new permissions. The user's Android application communicates with the MAS of the luminosity and temperature sensors in order to get these information and to define the

preference for the temperature (warm, hot or cold); and it asks for accessing the laboratory. In fact, every Android application uses an instance of *ContextNet* for MN.

## V. RELATED WORK

It is possible to find several works that try to integrate MAS in AmI systems. However, these solutions only provide communication with agents originally from its MAS, avoiding communication with other agents or embedded MAS that can enter in the AmI system eventually The Agent of Things (AoT) [15] is a definition for devices or things that are managed by a single agent in dynamic environments. The authors suggest that in such kind of environments, the communication between devices are highly programmable and depends on a previous interaction configuration. It is proposed a conceptual framework which consider a direct physical interaction between devices using the hardware layer and using a software layer where every agent represents a device. However, the framework is conceptual and agents are centralized in a software layer. In our approach, we consider a MAS as a thing and a real laboratory implementation as proof-of-concept.

In [16], it is discussed the use of MAS in IoT rising questions about communication and the use of protocols where the main objective is to implement functionalities for access control in a agent-based IoT. The architecture uses a central server for controlling and coordinating agents and it is also presented a hybrid system containing intelligent agents and IoT devices in traffic scenario. There is one embedded agent

for each device and the authors do not explicit how agents are organized. The Agent-based Cooperating SO (ACOSO) is used as IoT middleware for providing an IoT network where agents are devices and the whole group of agents is a MAS [17]. The ACOSO supports the development of MAS in the level of things. So, every smart object can be abstracted to a cooperating agent using Jade as AOPL. The agents are able of managing sensors and actuators; reasoning and decision making using local and distributed databases; and a communication system for the interaction between smart objects. However, the environment is closed and it is not possible of new devices to enter in the system and there is only one agent per device.

A decentralized approach with a framework and an architecture for engineering IoT applications based on autonomous smart objects is proposed in [8]. The authors defend that traditional smart objects are data gathers and senders and the data is stored and processed in central servers. The proposed smart objects are capable of running even if remote technologies (i.e. gateways) are not available. There is one agent for smart object and it is programmed using XML technologies (it does not use AOPL) in a web platform named Eve agent.

## VI. CONCLUSION

This work presented an architecture for the development of AmI systems employing the agent approach and supported by an IoT middleware named *ContextNet*. In this architecture, it is possible to assemble devices, which have embedded MAS responsible for controlling sensors and actuators and for communicating with other devices. Every device is an independent solution and it is free to enter and leave in the architecture. The proposed approach uses Jason framework for the development of the MAS. Besides, we enhanced a real laboratory with such technologies to use it as a proof-of-concept application. This architecture aims to provide a kind of framework for the development of AmI systems.

By using the proposed architecture, it is possible to create an AmI system, which allows communication among several devices at real time, where all devices connected can change information, offering flexibility in the choice of which technology employed in the solution. From MN — which do not use agents — the data information is transmitted to the *ContextNet* gateway in raw format. So, the discovery of context situations coming from these nodes only will happen exclusively in the server. In some cases, it could be necessary that the context and the reasoning for specific situations happen directly in the devices. In cases where the device is embedded with a MAS, a previous reasoning can be performed using the raw information captured as perceptions from **ARGO** agents reducing the processing in the server and avoiding bottlenecks.

This work also presented an extension of Jason to allow embedded MAS in mobile devices to communicate with other agents hosted and embedded in different devices. It provides a specific and new kind of agent that has the ability to communicate with other agents using *ContextNet*. In this type of agents, the middleware is part of the reasoning cycle of

the agent, which uses internal actions for sending messages to agents with the same ability and hosted in a different MAS. For future works, the architecture will be employed in other laboratories containing different devices and technologies. We also aim to improve and formalize the ability of infer rules in the *ContextNet* middleware. Besides, we also aim to create a *testbed* for automation of experiments and resource sharing, in order to assist in the validation of new proposals involving IoT technologies and MAS.

## REFERENCES

- [1] M. Weiser, "The computer for the 21st century," *IEEE pervasive computing*, vol. 1, no. 1, pp. 19–25, 2002.
- [2] D. Surie, O. Laguionie, and T. Pederson, "Wireless sensor networking of everyday objects in a smart home environment," in *Intelligent Sensors, Sensor Networks and Information Processing, 2008. ISSNIP 2008. International Conference on*. IEEE, 2008, pp. 189–194.
- [3] H. D. Soares, R. P. de Oliveira Guerra, and C. V. N. de Albuquerque, "Ftsp+: A mac timestamp independent flooding time synchronization protocol," in *XXXIV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC*. Sociedade Brasileira de Computação, 2016, pp. 820–832.
- [4] M. Endler, G. Baptista, L. Silva, R. Vasconcelos, M. Malcher, V. Pantoja, V. Pinheiro, and J. Viterbo, "Contextnet: context reasoning and sharing middleware for large-scale pervasive collaboration and social networking," in *Proceedings of the Workshop on Posters and Demos Track*. ACM, 2011, p. 2.
- [5] G. Pardo-Castellote, "Omg data-distribution service: Architectural overview," in *Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on*. IEEE, 2003, pp. 200–206.
- [6] M. Wooldridge, *An Introduction to MultiAgent Systems*. Wiley, 2009.
- [7] C. Maciel, P. C. de Souza, J. Viterbo, F. F. Mendes, and A. El Fallah Seghrouchni, *A Multi-agent Architecture to Support Ubiquitous Applications in Smart Environments*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 106–116.
- [8] M. E. P. Hernández and S. Reiff-Marganiec, "Towards a software framework for the autonomous internet of things," in *Future Internet of Things and Cloud (FiCloud), 2016 IEEE 4th International Conference on*. IEEE, 2016, pp. 220–227.
- [9] M. P. Singh and A. K. Chopra, "The internet of things and multi-agent systems: Decentralized intelligence in distributed computing," in *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*. IEEE, 2017, pp. 1738–1747.
- [10] R. H. Bordini, J. F. Hübner, and M. Wooldridge, *Programming Multi-Agent Systems in AgentSpeak using Jason*. John Wiley & Sons Ltd, 2007.
- [11] L. Silva, M. Endler, and M. Roriz, "Mr-udp: Yet another reliable user datagram protocol, now for mobile nodes," *Monografias em Ciência da Computação, nr*, vol. 1200, pp. 06–13, 2013.
- [12] M. E. Bratman, *Intention, Plans and Practical Reasoning*. Cambridge Press, 1987.
- [13] C. E. Pantoja, M. F. Stabile, N. M. Lazarin, and J. S. Sichman, "Argo: An extended jason architecture that facilitates embedded robotic agents programming," in *Engineering Multi-Agent Systems: 4th International Workshop, EMAS 2016*, M. Baldoni, J. P. Müller, I. Nunes, and R. Zalila-Wenkstern, Eds. Springer, 2016, pp. 136–155.
- [14] N. M. Lazarin and C. E. Pantoja, "A robotic-agent platform for embedding software agents using raspberry pi and arduino boards," in *9th Software Agents, Environments and Applications School*, 2015.
- [15] A. M. Mzahm, M. S. Ahmad, and A. Tang, "Enhancing the internet of things (iot) via the concept of agent of things (aot)," *Journal of Network and Innovative Computing*, vol. 2, no. 2014, pp. 101–110, 2014.
- [16] D. Rivera, L. Cruz-Piris, G. Lopez-Civera, E. de la Hoz, and I. Marsa-Maestre, "Applying a unified access control for iot-based intelligent agent systems," in *Service-Oriented Computing and Applications (SOCA), 2015 IEEE 8th International Conference on*. IEEE, 2015, pp. 247–251.
- [17] C. Savaglio, G. Fortino, and M. Zhou, "Towards interoperable, cognitive and autonomic iot systems: an agent-based approach," in *Internet of Things (WF-IoT), 2016 IEEE 3rd World Forum on*. IEEE, 2016, pp. 58–63.

# Understanding Normative BDI Agents Behavior

Francisco Cunha, Marx Viana, Tassio Sirqueira, Marcio Rosemberg and Carlos Lucena

Pontifical Catholic University of Rio de Janeiro (PUC-Rio)

Software Engineering Laboratory (LES)

Rio de Janeiro, Brazil

{fcunha, mleles, tmartins, mrosemberg, lucena} @inf.puc-rio.br

**Abstract**— Testing the autonomy of, and the interaction between, the agents in Multiagent Systems (MAS) is the frontal challenge of traditional software testing approaches. When we study MAS governed by norms – mechanisms created to restrain the behavior of agents – this challenge increases even further. However, agents are autonomous and it is not guaranteed that they will fulfill all norms. Given the fuzzy notion of “test”, especially in the context of MAS, in addition to the difficulties of dealing adequately with normative constraints, the overall understanding of how to handle the creation of tests for normative MAS is still vague. This paper proposes a testing tool to build and run MAS test scenarios and it relies on the use of aspect-oriented techniques to monitor the behavior of autonomous agents. We demonstrated our tool with a simulation of a traffic intersection scenario, based on the Brazilian Transit Code. Our experience shows that the tool can be used to build test scenarios that can achieve high fault detection effectiveness.

**Keywords** – BDI Agent; Autonomous Behavior; Normative Agents; Testing in Multiagent Systems.

## I. INTRODUCTION

Multiagent Systems (MAS) are societies in which autonomous, heterogeneous and independently designed entities work toward a common goal [9]. To reach this common goal, it is necessary to deal with the agents’ autonomy and establish a strategy that will allow open systems to provide social control mechanisms to ensure the desired order [9]. Agent autonomy is very important in MAS, however, from a testing perspective the characteristics of normative agents add many new challenges to software testability. Traditionally, software behavior can be easily tested and understood when compared to a reference behavior, whereas in multiagent systems, the behavior depends on the interactions with other agents in a dynamic environment.

This means that if on one hand agent technology helps to address application requirements of complex systems, on the other hand, its characteristics, such as the autonomy and the use of norms in the environment, bring obstacles to software testability [2]. According to Voas and Muller [2], testability has two facets: (i) controllability – the ability to control the test input, and (ii) observability – the ability to observe the output of the component under test. Agent autonomy impairs observability since agents may employ some degree of nondeterministic behavior. Consequently, it is hard to define (control) the test input that is not only derived from environment data but also from the messages received from concurrent conversations among agents – this is made worse with the use of norms.

This paper presents a tool named N-JAT4BDI: a JUnit-like testing tool implemented in Java and Aspect-Oriented Programming, which is a technique to improve the modularization of crosscutting concerns. N-JAT4BDI has been developed with the purpose of testing agents built in NBDI4JADE [12], a framework for normative agent-based applications that follows the BDI architecture. We can point out the following contributions: (i) an adaptation of JAT4BDI [13] that adds mechanisms to test the relevant properties of normative BDI agents and their interactions with others; (ii) a tool to support the implementation and automatic execution of test cases; (iii) a real test showcasing a Brazilian traffic scenario and (iv) a quality assessment of this test scenario by using a fault injection technique.

The remained of this paper is organized as follows: Section 2 presents the background and related work. Section 3 presents the testing approach to normative MAS. Section 4 presents design and implementation details of the N-JAT4BDI tool. Section 5 presents the usage scenario. Section 6 presents the evaluation of the results. Finally, Section 7 presents our conclusions and future work.

## II. BACKGROUND

This section summarizes the concepts of norms and their use in Multiagent Systems as well as MAS Testing approaches and their limitations.

### A. Norms and Normative Multiagent Systems

In MAS, norms are mechanisms commonly accepted as efficient means of regulating agents behavior and representing the way in which agents understand the responsibilities of other agents [4] [9]. The definition of the norms used in this work is represented by the following properties: Addressee, Condition (for example, Activation, Expiration), Motivation (for example, Rewards, Punishments), Deontic Concept, and State. The description of each property is given as such: (i) *addressee* is used to specify the agents or roles responsible for norm compliance; (ii) *activation* is the condition for the norm to become active; (iii) *expiration* is the validity condition for the norm to become inactive; (iv) *rewards* is used to represent the set of rewards to be given to the agent for norm compliance; (v) *punishments* is the set of punishments to be given to the agent for violating a norm; (vi) *deontic concept* is used to indicate whether the norm establishes an obligation, a permission, or a prohibition, and (vii) *state* is used to describe the set of states or actions that are being regulated [10].

## B. Multiagent Systems Testing

According to Nguyen *et al.* [17], the full testing process of a multiagent system consists of the following levels: unit, agent, integration (or group), system and acceptance.

Several approaches have proposed unit testing for individual agents in multiagent systems [6] [5] [7] [13] [16], whereas few studies deal with the issue of testing a MAS at group level [6] [15] [18] [1] [3]. According to Serrano [18], most approaches have focused on capturing and visualizing messages exchanged among agents, and do not provide ways of tracking the correlation among the agents' behavior. It is important to emphasize that none of the papers mentioned above provides an approach to verify the behavior of normative agents. The main focus of this work is to test the capability of an individual agent to fulfill a norm in order to reach its goal.

## III. THE APPROACH PROPOSED

This section presents a testing tool – the N-JAT4BDI – that proposes test cases to test normative BDI agents encoded in the NBDI4JADE framework [12]. The N-JAT4BDI tool resorted to the main ideas of the JAT4BDI approach [13] and added new features capable of monitoring the behaviors of normative NBDI4JADE agents.

### A. Overview

Our tool simulates real agent interactions by using *mock agents* [14]. The *mock agents* interact and exchange messages with the agent under test (AUT) in order to verify the AUT response and to check whether the environment was affected as expected.

Figure 1 depicts all the participants used in our testing approach: (i) *Agent Under Test (AUT)*: agent whose behavior is verified by the unit test execution; (ii) *Mock Agent*: a fake implementation of a real agent that interacts with the AUT; (iii) *Monitor*: responsible for monitoring agents' behaviors (reasoning cycle); (iv) *Synchronizer*: manages the test scenario execution by defining the order in which the mock agents interact with the AUT; (v) *Test Scenario*: defines a set of conditions to which the AUT will be exposed and verifies whether this agent obeys its specification under those conditions. Each scenario encompasses only one AUT and one, or more, *mock agents*, and (iv) *Test Case*: a particular situation that requires verification.

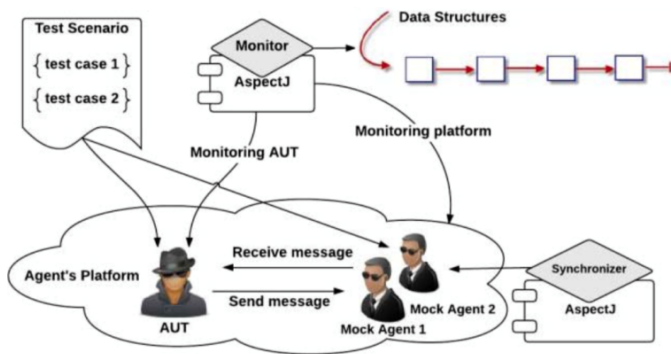


Figure 1. Flow among the participants of the unit test [13]

The workflow steps used by the tool are: (i) to create a *Test Scenario* that will define the test cases involved; (ii) to start the *Test Scenario*; (iii) the test case creates and starts the AUT and *Mock Agents*. The component Monitor starts to observe the AUT's reasoning cycle (its beliefs, executed plans, goals, and norms fulfilled, or refused), and (iv) mock agents exchange messages with the AUT. During this interaction process, the Monitor keeps track of the information gathered during the execution. To do so, it uses a set of data structures to store that information. Both the Monitor and Synchronizer were implemented by aspects [8].

### B. Normative assertions

Aiming to support norm fault identification in NBDI4JADE agents, we provide a set of assertive methods that follow the JUnit style and are capable of verifying the agents' decisions. These assertion methods check the information stored by the tool during the agent's execution.

The main assertions are described below: (i) ***assertNormActive***: It checks whether a norm is active in the environment; (ii) ***assertNormFulfillment***: It checks whether a norm is fulfilled by the agent; (iii) ***assertNormAffectGoal***: It checks whether a norm affects an agent's goal; (iv) ***assertNormAffectPlan***: It checks whether a norm affects an agent's plan; (v) ***assertNormAddressee***: It checks whether a norm is addressed to the agent under test; (vi) ***assertNormExpired***: It checks whether a norm has expired during the AUT's execution; (vii) ***assertNormReward***: It checks whether the AUT has received a reward for fulfilling the norm; (viii) ***assertNormPunishment***: It checks whether the AUT has received a punishment for violating the norm; (ix) ***assertNormDeonticConcept***: It checks the type of norm constraint (obligation, permission or prohibition) that affects the AUT, and (x) ***assertNormState***: It checks the internal state of an element that has been regulated.

## IV. N-JAT4BDI: DESIGN AND IMPLEMENTATION DETAILS

This section discusses the main classes of the N-JAT4BDI tool.

### A. NBDI4JadeMockAgent

The *NBDI4JadeMockAgent* class implements the mock agent concept in N-JAT4BDI and is an instance of the NBDI4JADE class; it has a simple plan that executes a mock agent's single action. The messages exchanged between the mock agents and the AUT is also stored in the internal data structures and can be accessed by using assertive methods.

### B. Monitor

The *Monitor* defines the *pointcut* that will intercept both the normative agents and the NBDI4JADE framework in order to observe the agents' reasoning, their decisions and all the changes that occurred in the environment.

### C. Synchronizer

The *Synchronizer* intercepts the code of the *NBDI4JadeMockAgent* class and orchestrates the sequence of interaction between the AUT and mock agents.

#### D. NBDI4JadeTestCase

This class extends the JUnit framework features to support the NBDI4JADE agent tests. As a result, developers can create agent tests more easily since they will be using their own experience with JUnit tests. This is possible because *NBDI4JadeTestCase* provides a set of JUnit-based assertive methods to verify the normative agent’s behavior and to manage the execution environment before a test scenario starts.

#### V. USAGE SCENARIO

Our case study focuses on the simulation of a traffic scenario in Brazil. This section summarizes our experience with the testing tool and its use in this scenario.

##### A. Motivation

According to **Article 29** of the Brazilian Transit Code (BTC), the right of way rules for vehicles arriving at an uncontrolled intersection are: (i) *Norm1*: vehicles moving on main thoroughfares have the preference; (ii) *Norm2*: in the case of a traffic circle, the ones circulating around it have the preference, and (iii) *Norm3*: in all other cases, vehicles coming from the right have the preference. In addition, **Article 38**, in its sole paragraph, states that before making a right or left turn, or merging onto traffic, the driver must yield to oncoming pedestrians, cyclists and vehicles, always respecting the norms of preference described in **Article 29**.

##### B. Usage Scenario

This simulation of Brazilian traffic rules was implemented to briefly demonstrate how N-JAT4BDI can be used to test a normative agent. The complete simulation involves autonomous cars (agents), highways, traffic circles, traffic intersections, and traffic rules. The goal of the autonomous cars is to arrive at their destination without accidents, following local traffic rules.

Figure 2 presents the scenario implemented with the NBDI4JADE framework: three cars arrive at an intersection at the same time. The goals of the autonomous cars are: (i) the pink car wants to proceed on street 1; (ii) the yellow car wants to proceed on street 2 and will have to cross street 1, and (iii) the red car is on street 1 and wants to turn left onto street 2. In this scenario, however, there are no traffic signs and the agents need to make decisions to avoid collision among the cars, taking into account Brazil’s traffic rules.

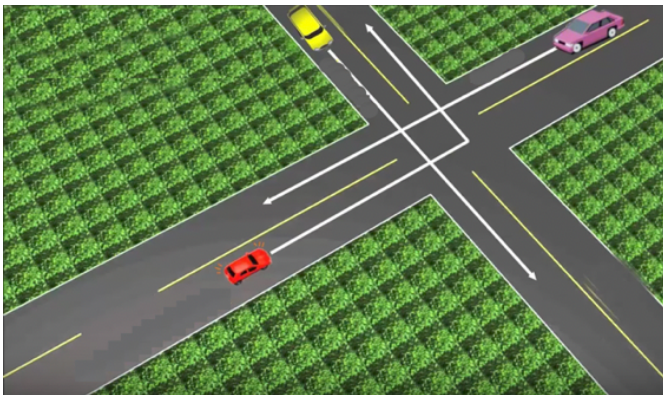


Figure 2. Traffic Intersection rules in Brazil

In our scenario, neither *Norm1* nor *Norm2* of **Article 29** of the BTC can be applied. Therefore, the agents need to decide whether they will fulfill, or violate, *Norm3* of **Article 29**. In our simulation, they all fulfilled *Norm3*, as follows: (i) The PINK car arrives at the intersection and stops because the YELLOW car is on its right; (ii) The YELLOW car arrives at the intersection and stops because the RED car is on its right; (iii) The RED car arrives at the intersection and there is no car on its right, therefore, the agent’s reasoning cannot comply with **Article 29** and must, instead, comply with **Article 38**.

Because the BTC does not deal with similar situations at uncontrolled intersections, it creates an impasse, and requires that the agent’s reasoning process be improved.

Due to space limitation, we describe only one simple test scenario and its implementation, as demonstrated in Table 1.

TABLE I. TEST SCENARIO EXAMPLE

Agent Under Test	Test Scenario Description
The RED autonomous car (Red car)	The RED car arrives at the intersection and there is no car on its right, therefore, the agent’s reasoning cannot comply with article 29.

##### C. Encoding test scenario

Figure 3 illustrates the implementation of the test scenario. Line 13 of the test case starts the agent under test (Red car) and line 15 configures the concurrence from the test environment execution. Line 17 creates a local norm that emulates a real norm in the environment. Line 18 checks whether the norm is *active* in the environment and line 19 checks whether the norm is *addressed* to the agent under test.

```

7 public class RedCarTestCase extends NJAT4BDITestCase {
8
9     static final long DELAY = 5000L;
10
11     public void testNormAddressee() {
12
13         startAUT("RedAutonomousCar", new RedAutonomousCarAgent());
14
15         waitUntilIAUHasFinished(DELAY);
16
17         Norm norm = new Norm("article29");
18         assertNormActive(norm);
19         assertNormAddressee(norm);
20     }
21 }

```

Figure 3. Checks if the norm is active and was addressed to the Red agent

Figure 4 depicts the result of the test case execution visualized in a JUnit style.

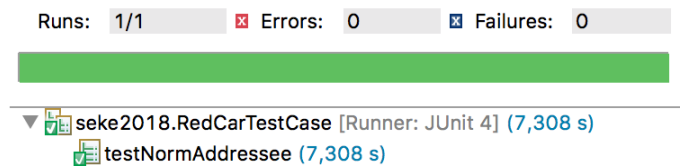


Figure 4. The result of test casse execution

#### VI. EVALUATION

Fault injection is considered a very useful technique to evaluate the effectiveness of testing approaches. The key idea

is to introduce faults during system execution and verify whether the testing approach precisely detects the injected fault [11], which depends on the fault model associated with a testing approach.

In order to estimate the effectiveness of the test cases development, we implemented a module in the tool that uses Java Annotations and aspect-oriented programming that intercepts the execution of the N-BDI4JADE agents and introduces faults in our normative agent.

### A. The Fault Injector

The fault injector component adds specific faults defined by the annotation. Each annotation describes one type of fault and aims to check how agents react to the fault. For instance: (i) *@ActivateNorm*: forces the activation of a norm in the environment. The attribute of this annotation is the norm that will be activated; (ii) *@DeactivateNorm*: forces the deactivation of a norm in the environment. The attribute of this annotation is the norm that will be deactivated; (iii) *@IncreaseReward*: forces the increase in the agent’s reward score even when a norm is not fulfilled; (iv) *@IncreasePunishment*: forces the increase in the agent’s punishment score even when a norm is not fulfilled; (v) *@ChangeAddressee*: forces a change in the addressee of a norm, and (vi) *@ChangeFulfillment*: forces the agent to fulfill, or not, a norm.

### B. Results

We have injected 22 faults inside our simulation to check whether the test scenarios were able to diagnose the injected faults. According to the results, the N-JAT4BDI tool helps the developer in the identification of these types of faults. We attribute these results to the use of the testing driven development technique during the development of our testing tool. Thus, the test cases became consistent and accurate whereas the injection of faults that involved reward and punishment, failed completely. Table 2 summarizes the results.

TABLE II. FAULTS INJECTED AND DETECTED BY THE TEST SCENARIOS

Fault Type	Faults Injected / Detected
Activate Norms	9 / 9
Deactivate Norms	1 / 1
Reward Norms	3 / 0
Punishment Norms	3 / 0
Addressee Norms	3 / 3
Fulfillment Norms	3 / 3

## VII. CONCLUSION AND FUTURE WORK

This work presented N-JAT4BDI, a testing tool for building and running automated test cases for normative agents with N-JAT4BDI to verify a Brazilian traffic simulation involving traffic intersections. To evaluate our approach, we used a fault injection technique to assess the quality of the test scenarios developed for this simulation. The results have shown that

N-JAT4BDI can effectively uncover bugs in normative agents. As future work, we plan to improve the normative fault model and add features when testing other normative properties such as reward and punishment.

## REFERENCES

- [1] A. Ferrando, D. Ancona and V. Mascardi, “Decentralizing mas monitoring with decamon”, Proceedings of the Conference on Autonomous Agents and MultiAgent Systems, pp. 239–248, 2017.
- [2] J. M. Voas and K. W. Miller, “Software testability: The new verification.”, IEEE software, v.12, n.3, pp. 17–28, 1995.
- [3] N. M. do Nascimento, C. J. M. Viana, A. von Staa and C. J. P. de Lucena, “A Publish-Subscribe based Architecture for Testing Multiagent Systems”, 2017.
- [4] M. Alberti, A. Gomes, R. Gonçalves, J. Leite, and M. Slota, “Normative systems represented as hybrid knowledge bases,” Computational Logic in Multi-Agent Systems, pp. 330–346, 2011.
- [5] R. Coelho, E. Cirilo, U. Kulesza, A. von Staa, A. Rashid and C. J. P. de Lucena, “Jat: A test automation framework for multi-agent systems”, in IEEE International Conference on Software Maintenance, pp. 425–434, 2007.
- [6] D. T. Ndumu, H. S. Nwana, L. C. Lee and J. C. Collis, “Visualising and debugging distributed multi-agent systems”, in Proceedings of the third annual conference on Autonomous Agents. ACM, 1999, pp. 326–333.
- [7] Y. Abushark, J. Thangarajah, T. Miller, J. Harland and M. Winikoff, “Early detection of design faults relative to requirement specifications in agent-based models”, in Proceedings of the International Conference on Autonomous Agents and Multiagent Systems, pp. 1071–1079, 2015.
- [8] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J. M. Loingtier and J. Irwin, (1997, June). “Aspect-oriented programming”, In European conference on object-oriented programming (pp. 220-242). Springer, Berlin, Heidelberg.
- [9] F. L. y López, “Social power and norms: Impact on agent behavior”, Ph.D. dissertation, University of Southampton, June, 2003.
- [10] V. T. da Silva, “From the specification to the implementation of norms: an automatic approach to generate rules from norms to govern the behavior of agents,” Autonomous Agents and Multi-Agent Systems, vol. 17, no. 1, pp. 113–155, 2008.
- [11] J. Voas and G. McGraw, “Software Fault Injection: Inoculating Programs Against Errors”, Wiley, 1998.
- [12] F. J. P. da Cunha, T. F. M Siqueira, M. L. Viana and C. J. P. de Lucena, “Extending BDI Multiagent Systems with Agent Norms”, International Conference on Intelligent Agent Technology, 2018 – In Press.
- [13] F. J. P. da Cunha, A. D. da Costa, M. L. Viana and C. J. P. de Lucena, “JAT4BDI: An Aspect-based Approach for Testing BDI Agents”, Web Intelligence and Intelligent Agent Technology (WI-IAT), 2015 IEEE/WIC/ACM International Conference on. Vol. 2. IEEE, 2015.
- [14] R. Coelho, U. Kulesza, A. von Staa and C. J. P. de Lucena, “Unit testing in multi-agent systems using mock agents and aspects”, In Proceedings of the international workshop on Software engineering for large-scale multi-agent systems, 2006, pp. 83–90, ACM.
- [15] J. J. Gomez-Sanz, J. Botía, E. Serrano, and J. Pavón, “Testing and debugging of mas interactions with ingenias,” in International Workshop on Agent-Oriented Software Engineering. Springer, 2008, pp. 199–212.
- [16] V. J. Koeman, K. V. Hindriks and C. M. Jonker, “Automating failure detection in cognitive agent programs”, Proceedings of the International Conference on Autonomous Agents & Multiagent Systems, 2016, pp. 1237–1246.
- [17] C. D. Nguyen, A. Perini, C. Bernon, J. Pavón and J. Thangarajah, “Testing in multi-agent systems,” in International Workshop on Agent-Oriented Software Engineering. Springer, 2009, pp. 180–190.
- [18] E. Serrano, A. Muñoz and J. Botía, “An approach to debug interactions in multi-agent system software tests”, Information Sciences, vol. 205, pp. 38–57, 2012.

# Towards a Representation of Enterprise Architecture based on Zachman Framework through OMG Standards

Miguel Ehécatl Morales-Trujillo<sup>a,b</sup>, Boris Escalante-Ramírez<sup>b</sup>, Pilar Ángeles<sup>b</sup>, Hanna Oktaba<sup>c</sup> and Guadalupe Ibargüengoitia-González<sup>c</sup>

<sup>a</sup>University of Canterbury, Christchurch, New Zealand  
miguel.morales@canterbury.ac.nz

<sup>b</sup>Facultad de Ingeniería, Universidad Nacional Autónoma de México, Mexico City, Mexico  
{boris, pilarang}@unam.mx

<sup>c</sup>Facultad de Ciencias, Universidad Nacional Autónoma de México, Mexico City, Mexico  
{hanna.oktaba, gig}@ciencias.unam.mx

**Abstract** — This paper presents a rearrangement of the Zachman framework carried out in order to facilitate the representation of an enterprise architecture (EA). The approach proposes a diagram-based alternative to model EA using two well-known Object Management Group standards: UML and BPMN. The proposal is directed to organizations in charge of developing and maintaining software systems under the premise that the people who develop software are highly familiarized with both standards. The proposal reduces 30 elements required by Zachman to 17, which are 7 diagrams and 10 documents, providing organizations with a viable alternative to model their structure, processes and environment in a business-oriented vision.

**Keywords**— Enterprise Architecture, Zachman framework, software development, UML, BPMN, modeling language

## I. INTRODUCTION

Organizations constantly look for an improvement and one of the strategies for a more efficient management is process automatization through Information Technology (IT) [1]. Automatization of processes aims at fulfillment of objectives, metrics and requirements of an organization. However, for a process to be correctly aligned, it should be well-defined in a common language and amendable to detailed analysis.

Considering this premise, it becomes an acute necessity to establish a mechanism for representing the elements that comprise an organization. This representation of the elements, their relationships and executional context constitute the Enterprise Architecture (EA) of the organization. EA is a structured and aligned collection of plans for the integrated representation of a given business and IT landscape [2]. EA can be a useful tool for aligning the IT application and organization's activities. Besides, it can facilitate business success to the effectiveness by using information of management strategic and IT resources [3].

In the process of creating an EA representation for software development organizations some obstacles are faced. In the first place, it is imperative to define a method to build the elements of an EA. Even though, as exposed in [4], the way in which an EA specification would be built is not relevant, this premise does not hold at the moment of integrating the elements. If the EA structure is not properly defined since the beginning, it will be difficult to maintain relationships between future elements, which may substantially weaken the cohesion among them.

Moreover, in [5] it is established that the current EA implementation methods have a broad scope and a lack of structure; this usually causes complication and difficulty in implementation. Besides, the fact that there is no comparison between existent EAs complicates the initiative to define EA and leaves those defining it for the first time without a point of reference or comparison.

Last but not least, there is a necessity to offer alternatives to manage an EA evolution. Nowadays there is a considerable gap between defining an EA and its management over time. Any changes within the organization directly affect the previously defined EA, which is why there is a need for a specific mechanism to be able to manage the change and to allow for the EA to evolve alongside the organization [6].

Addressing this necessity, this paper presents an alternative to represent the EA of an organization, which is based on Zachman framework [7] in terms of two widely used Object Management Group (OMG) standards: the Unified Modeling Language (UML) [8] and Business Process Model and Notation (BPMN) [9].

This paper is organized as follows: Section II presents the background of the proposal and its fundamentals. Section III details the proposed adaptation of the Zachman framework and the mechanism to represent the EA. Section IV contains preliminary results related to the applicability of the proposal. Finally, section V presents conclusions and future work.



## II. BACKGROUND

In this section an overview of EA definitions, the Zachman framework and the relevant for this paper OMG standards are presented. Related to the Zachman framework work and OMG modeling languages are described as well.

### A. Enterprise Architecture

EA is defined as a process of strategic planning that integrates the business management with IT in order to improve the organization's financial and enterprise efficiency. EA is a set of descriptive representations relevant for describing an enterprise so that it can realize management requirements and be maintained over the period of its useful life [10]. EA is an approach to enterprise information systems management that relies on models of the information systems and their environment [11]. EA supports the analysis, design and engineering of business-oriented systems through multiple views [12].

In order to create an EA, frameworks like Zachman, TOGAF ADM [13], DoDAF [14] or MODAF [15] have been created. An EA framework is a model used by an organization to develop good corporate governance, creating added value for their business [16]. For the purposes of this proposal, the Zachman framework will be the base to represent an EA and is detailed in the next subsection.

### B. The Zachman framework

The Zachman framework is an ontology that represents EA concepts and their relationships. The ontology is developed through an empirical approach and answers the questions *who? when? why? what? how? and where?*. This question-based approach, according to Zachman, allows for a full and understandable description of complex ideas, which is the case with EA.

Each question aims at finding the necessary Data (what?), the Function (how?), the People (who?), the Network (where?), the Time (when?) and the Motivation (why?) involved in the EA. According to [17] each element is defined as follows:

**Data (Thing—Relationship—Thing):** this element focuses on the material composition of the product.

**Function (Process—Input/Output—Process):** this element focuses on the functions or transformations of the product.

**People (People—Work—People):** this element focuses on the people, the manuals and the operating instructions or models they use to perform their tasks.

**Network (Node—Line—Node):** this element focuses on the geometry or connectivity of the product.

**Time (Event—Cycle—Event):** this element focuses on the life cycles, timing and schedules used to control activities.

**Motivation (End—Means—End):** this element focuses on goals, plans and rules that prescribe policies and ends that guide the organization.

In addition, the framework proposes 5 models, which, according to [17], are defined as follows:

**Scope (Contextual):** describes the models, architectures and representations that provide the boundaries for the organization.

**Business model (Conceptual):** describes the models, architectures and descriptions used by the individuals who are the owners of the business process.

**System model (Logical):** describes the models, architectures and descriptions used by engineers, architects and those who mediate between what is desirable and what is technically possible.

**Technology model (Physical):** describes the models, architectures and descriptions used by technicians, engineers and contractors who design and create the actual product.

**Detailed representations (Out-of-context):** describes the actual elements or parts that are included in, or make up, the final product.

The 6 questions and the 5 models comprise a two-dimensional matrix of 30 cells. Each cell describes or represents a particular element, which can be defined by means of diagrams, documents or work products, according to the organization preferences.

The EA representation that is achieved following this framework is a static view of the organization, as a consequence it is impossible to model its operational processes. The major advantage, however, is the possibility to represent the EA fundamental elements in a precise and well-defined manner.

The Zachman framework is a well-known alternative for modeling an organization's EA, however, it faces three considerable weaknesses: low cohesion of its elements, lack of a method to use it and lack of specificity in the cells description.

In addition, there are no detailed examples demonstrating the successful practical application of the Zachman framework [4], which is a strong limiting factor.

### C. OMG standards and Related work

OMG develops IT standards for a broad variety of industries. Two of the most well-known and broadly used in software engineering related industries are UML and BPMN. UML is one of the most used specification in IT industry; software engineering practitioners know the specification and use it daily in their projects.

In order to provide a unified language architecture, the OMG developed the Unified Profile for DoDAF and MODAF (UPDM) [18]. The UPDM specification reuses a subset of UML and provides additional extensions to allow the representation of architecture models. UPDM is based on UML class diagrams, where each class represents common elements of DoDAF and MODAF; however, the only way to differentiate between classes is through stereotypes. As a result, the main drawback of UPDM is a lack of expressiveness.

A more expressive language is ArchiMate [19] that is an EA specific modeling language created by the Open Group. ArchiMate is aligned with TOGAF. In this case, an important obstacle that hinders its spreading is its level of complexity as well as its lack of familiarity to practitioners. Besides, ArchiMate is targeted to big companies.

Based on the assumption that one quarter of all EA representations is done through the Zachman framework, the OMG published a proposal that represents the Zachman framework cells through OMG modeling specifications [20].

The proposal reuses UPDM, BPMN and UML mainly. However, there are also Zachman's cells that are not represented at all.

### III. PROPOSED SOLUTION

The enterprise willing to model its EA has to cope with two challenges: it must define procedures for gathering the needed information and must devise a conceptual model defining the necessary information [21].

Modeling EA requires representing multiple diagrams of an enterprise, which typically shows the multiples business entities, IT systems and the services they offer [22]. Therefore we propose to model these aspects by using UML and BPMN mainly. On the one hand, the proposed solution pursues the goal of adjusting the EA elements to the organization's context. In other words, we intend that the created EA views become meaningful to the organization and fulfill its necessities or objectives.

On the other hand, the representation mechanism should use a language that is close to the organization's members, so that the EA representation can be easily assimilated and applied, demanding as little effort as possible. It is also possible to use a more complex notation, which may allow to keep more aspects together thus reducing the number of cells; however, our main goal was to increase comprehension and simplicity in the representation.

In the following lines, we describe the proposed readjustment to the Zachman framework together with the alternative that was chosen to represent each of the elements.

#### A. Rearrangement

The first step taken to create this proposal was to rearrange the elements of the Zachman framework. This was done in order to reduce the number of cells proposed by Zachman. A grouping of the cells of the framework in first place would reduce the complexity and the number of work products to represent the EA.

On the other hand, taking into account software engineers' reasoning, several elements were combined. For example, Business model and System model of the Data column are represented by an Entity-Relationship diagram in the context of software development, while, within the Zachman framework, they are separated cells.

Another example is that it comes natural for software engineers to model processes in terms of three fundamental elements: activities, work products and roles. Even though the Zachman framework places these elements on the layer Scope, they are separated into Function (what is done) and People (who does that).

This separated representation renders incomplete and loses the perspective of what is being modelled. Therefore, the proposed rearrangement merges both elements into one, which might be represented by a BPMN diagram integrating what is being done and who is in charge of it.

During the software design phase, the database is often modelled by the means of tables and class diagrams. The class diagrams represent a table through its attributes and methods. Therefore, the Physical Data Model and the System Design, attached to the questions *What?* and *How?* in the Technology model, can be combined and represented as a class diagram. This class diagram will show the database and the tables that express the persistency of the system.

We propose to integrate and rearrange several of the 30 Zachman's cells, finally obtaining 17. For the proposed model to be practical, we name each element as presented in [17]; in case the elements are merged, their names are introduced through a slash. In the following subsections the 17 elements are described in more detail.

Figure 1 shows the obtained rearrangement, the first work products to be created are lists, shown in green; the blue ones are diagrams and red ones are documents.

#### B. Representation using diagrams

In this subsection we describe the elements represented through diagrams.

1) *Semantic model / Logical data model*: represents entities (things) and their relationships that are involved in the Business model and its logic representations in the System model. It is carried out by means of an Entity-Relationship diagram.

2) *Physical data model / System design*: represents the databases and the domains of the Technology model. The representation is carried out by means of a class diagram.

3) *Business process model / Work flow model*: represents the processes, resources, persons and work products involved in the business model. The representation of these elements is carried out by means of a BPMN or a UML activity diagram.

4) *Application architecture / Human interface architecture / Presentation architecture*: it is a combination of the System model with the Technology model and represents the functions of the system and its users. Both aspects are represented by means of a Use case diagram.

5) *Distributed system architecture / Technology architecture / Network architecture*: the nodes, communication protocols, hardware and software that are necessary to allow communication between different locations of the organization are represented by means of a deployment diagram. This diagram represents a physical distribution of objects alongside with how they relate and communicate with each other.

6) *Master schedule / Processing structure / Control structure*: the organization's time-related aspects can be represented through a statechart diagram. The states that business or organizational systems go through as well as the events that cause a change of a state are the main components of the Business, System and Technology models.

7) *Timing definition*: This element describes events and their times. It is possible to represent the active state of each organizational process by means of a time diagram.

	<b>DATA</b> <i>What</i>	<b>FUNCTION</b> <i>How</i>	<b>PEOPLE</b> <i>Who</i>	<b>NETWORK</b> <i>Where</i>	<b>TIME</b> <i>When</i>	<b>MOTIVATION</b> <i>Why</i>
<b>SCOPE</b> (Contextual)	Things important to the business List of things • Things • Entities	Processes the business performs List of processes • Processes • Functions	Organizations important to the business List of organizations • Organizations • People	Locations in which the business operates / Business Logistics System List of locations • Locations • Offices • Network	Events significant to the business List of events • Events • Time	Business goals / Business Plan List of business goals • Business objectives • Business strategy
<b>BUSINESS MODEL</b> (Conceptual)	Semantic Model / Logical Data Model E/R diagram • Entities • Relationships	Business Process Model / Work Flow Model BPMN diagram • Resources • Work products • Persons			Master Schedule / Processing Structure / Control Structure UML Statechart diagram • States • Events	
<b>SYSTEM MODEL</b> (Logical)		Application Architecture / Human Interface Architecture / Presentation Architecture UML Use Case diagram • Functions • Users		Distributed System Architecture / Technology Architecture / Network Architecture UML Deployment diagram • Nodes • Protocols • Software systems • Hardware		Business Rule Model / Rule Design / Rule Specification List of business restrictions • Assertions • Hypotheses • Constraints • Conditions
<b>TECHNOLOGY MODEL</b> (Physical)	Physical Data Model / System Design UML Class diagram • Databases • Tables (Domains)					
<b>DETAILED REPRESENTATIONS</b> (Out-of-Context)	Data Definition Data dictionary • Attributes • Constraints	Program Software	Security Architecture Organigram • Persons • Roles • Privileges		Timing Definition UML Timing diagram • Events • Timing	
<b>FUNCTIONING ENTERPRISE</b>	Actual Business Data	Actual Application Code	Actual Business Organization	Actual Physical Networks	Actual Business Schedule	Actual Business Strategy

Fig. 1. Rearrangement of the Zachman framework

### C. Representation using lists and documents

The Scope layer in particular describes the context in terms of lists, 4 of the 6 Scope layer elements did not undergo any rearrangement nor integration with other elements:

1. List of things important to the business.
2. List of processes the business performs.
3. List of organizations important to the business.
4. List of events significant to the business.

The remaining two were merged with the cell from an inferior row.

1) *List of locations / Business logistics system*: this document enlists the organizational offices and their descriptions.

2) *List of business goals / Business plan*: this document enlists the business objectives as well as the strategies for achieving them.

3) *Data definition*: contains a detailed representation of data carried out by means of a data dictionary containing fields, their descriptions and restrictions applied to them.

4) *Program*: the programs that the organization uses to carry out its functions are represented by means of a tools and software systems list.

5) *Security architecture*: the detailed representation of the people involved should contain the persons or identities, and their roles and privileges. This representation is carried out by means of an organigram.

6) *Business rule model / Rule design / Rule specification*: this document gathers the business rules and restrictions. By means of assertions, hypothesis and restrictions the context for achieving the organizational objectives is described.

## IV. RESULTS

The proposal of representing an EA by means of work products, in this case diagrams and documents, originates in the idea from [4]. Ylimäki states that “since the Zachman framework is not a methodology, a method is needed to fill in the framework cells”. However, Ylimäki’s study offers 52 work products to cover the EA definition.

Our proposal, on the other hand, reduces the number of the Zachman’s cells and offers 17 work products to cover them, which we consider an advantage. Besides, although they are created specifically for EA, they are based on the languages that belong and are closely familiar to software development organizations and their work teams. In addition, the UML-based approach allows representing a wider range of enterprise concerns [24], taking advantage of its flexibility and popularity.

The preliminary results are classified in advantages and drawbacks. The following are the advantages that are identified in relation to the proposed solution: (i) software development organizations widely use BPMN and UML. In consequence, people in charge of creating, maintaining or using an EA can understand its language. Besides, diagrams increase the EA expressiveness; (ii) work products and diagrams used in this proposal are fully familiar to members of work teams, which means that minimal effort is required to comprehend and create them; and (iii) the number of the necessary work products is

reduced, making an EA definition simpler and lighter. Once the proposal is validated through case studies, further reductions of work products or more optimal rearrangements could be proposed.

However, several disadvantages are identified: (i) once the EA is created, additional effort is required for its maintenance and evolution. Since there is no explicit framework to guide these processes, the organization has to manage them by its own; (ii) as mentioned before, the Zachman framework faces three considerable weaknesses: low cohesion of its elements, lack of a method to use it and lack of specificity in the cells description. Even though the identified disadvantages are significant, they are all inherited from the original Zachman framework.

## V. CONCLUSIONS

This proposal is an initial approach towards how software development organizations can build their EA. EA is a complex concept that aims at modeling the structure and behavior of an organization and enables its stakeholders to make decisions. There is no one-size-fits-all template for EA. However, the Zachman framework is widely acknowledged to encompass all the concepts necessary to describe an organization [20].

The Zachman framework provides an alternative for modeling an organization's EA; however, as its own creator said: "so, if you ask who is successfully implementing the whole framework, the answer is nobody that we know of yet" ([http://archive.visualstudiomagazine.com/ea/magazine/spring/online/druby3/default\\_pf.aspx](http://archive.visualstudiomagazine.com/ea/magazine/spring/online/druby3/default_pf.aspx), accessed 01/05/2018). Up to now there is no solid evidence to reject this statement. The intention of our proposal is to simplify the framework, taking advantage of the fact that it is well-known in the industry although rarely used.

This proposal integrates the Zachman framework and a work product based approach. The work products are mainly created by means of UML and BPMN, and represent the fundamental components of an EA. In addition, populating the Zachman framework with OMG modeling specifications is widely supported by software tools [20]. We believe that this joined approach will allow more software development organizations to get involved into the subject of EA and, what is more important, be able to define their own by using reachable and well-known to their work teams tools.

As future work we consider the following: (i) to create real-life examples applying the proposed rearrangement; (ii) to create a guide for work teams to be able to define their own EA; and (iii) establish a framework for managing the future evolution of already defined EAs.

## ACKNOWLEDGMENT

This work has been developed under the "Programa de Becas Postdoctorales en la UNAM" of the Dirección General de Asuntos del Personal Académico (DGAPA) of the Universidad Nacional Autónoma de México (UNAM).

## REFERENCES

[1] S. Townson. Why does Enterprise Architecture Matter? The Open Group (2008)

[2] M. Zhang, H. Chen and A. Luo. A Systematic Review of Business-IT Alignment Research with Enterprise Architecture. *IEEE Access*, Vol. 6, DOI: 10.1109/ACCESS.2018.2819185 (2018)

[3] D. Rusli and Y. Bandung. Designing an Enterprise Architecture based on TOGAF ADM and MIPI. In: *Proc. Of the Intl. Conf. on Information Technology Systems and Innovation*, pp. 38–43, DOI: 10.1109/ICITSI.2017.8267915 (2018)

[4] T. Ylimäki and V. Halttunen. Method Engineering in Practice: A Case of Applying the Zachman Framework in the Context of Small Enterprise Architecture Oriented Projects. In: *Proc. of the Information, Knowledge, Systems Management*, Vol. 5, No. 3, pp. 189–209 (2006)

[5] S. Leist and G. Zellner. Evaluation of current architecture frameworks. In: *Proc. of the ACM Symposium on Applied Computing*, pp. 1546–1553 (2006)

[6] B. Rouhani, M. N. Mahrin, F. Nikpay, R. B. Ahmad and P. Nikfard. A systematic literature review on Enterprise Architecture Implementation Methodologies. *Information and Software Technology*, No. 62, pp. 1–20, DOI: 10.1016/j.infsof.2015.01.012 (2015)

[7] J. Zachman. A framework for information systems architecture. *IBM Systems Journal*, Vol. 26, No. 3 (1987)

[8] OMG. Unified Modeling Language (UML). Technical report, Object Management Group, Needham, MA, USA (2017)

[9] OMG. Business Process Model and Notation (BPMN) 2.0. Technical report, Object Management Group, Needham, MA, USA (2011)

[10] J. Zachman. Enterprise architecture: The issue of the century" *Database Programming and Design*, vol. 10, no. 3, pp. 44–53 (1997)

[11] A. Källgren, J. Ullberg and P. Johnson. A Method for Constructing a Company Specific Enterprise Architecture Model Framework. In: *Proc. of the 10th Int. Conf. on Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing*, pp. 346–351, DOI: 10.1109/SNPD.2009.103 (2009)

[12] M. Bakhshandeh, G. Antunes, R. Mayer, J. Borbinha and A. Caetano. A Modular Ontology for the Enterprise Architecture Domain. In: *Proc. of the Int. Enterprise Distributed Object Computing Conference Workshops*, pp. 5–12, DOI: 10.1109/EDOCW.2013.8 (2013)

[13] Open Group. TOGAF 9.1 Architecture Development Cycle (ADM). Reference Card (2011)

[14] United States Department of Defense. Department of Defense Architecture Framework (2015)

[15] British Ministry of Defence. The MOD Architecture Framework (2012)

[16] I. Alonso, J. Verdún and E. Tovar. The IT Implicated Within the Enterprise Architecture Model: Analysis of Architecture Models and Focus IT Architecture Domain. In: *Proc. of the IEEE Int. Conf. on Service-Oriented Computing and Applications (SOCA)*, DOI: 10.1109/SOCA.2010.5707174 (2010)

[17] D. Frankel, P. Harmon, J. Mukerji, J. Odell, M. Owen, P. Rivitt, M. Rosenm and R. Soley. The Zachman Framework and the OMG's Model Driven Architecture. *Business Process Trends* (2003)

[18] OMG. Unified Profile for DoDAF and MODAF (UPDM). Technical report, Object Management Group, Needham, MA, USA (2013)

[19] Open Group. ArchiMate 3.0.1. Specification (2017)

[20] OMG. OMG's Enterprise Architecture Specifications (white paper). Object Management Group, Needham, MA, USA (2015)

[21] S. Buckl, F. Matthes, C. Schweda. Conceptual Models for Cross-cutting Aspects in Enterprise Architecture Modeling. In: *Proc. of the 14th IEEE Int. Enterprise Distributed Object Computing Conference Workshops*, pp. 245–252, DOI: 10.1109/EDOCW.2010.18 (2010)

[22] H. Dam, L.-S. Lê and A. Ghose. Supporting change propagation in the evolution of enterprise architectures. In: *Proc. of the 14th IEEE Int. Enterprise Distributed Object Computing Conference*, pp. 24–33, DOI: 10.1109/EDOC.2010.23 (2010)

[23] J. Zachman and J. Sowa. Extending and Formalizing the Framework for Information Systems Architecture. *IBM Systems Journal*, Vol. 31, No.3 (1992)

[24] A. Gerber, A. van der Merwe and K. Bayes. An Investigation into UML Case Tool Support for the Zachman Framework. In: *Proc. Of the Enterprise Systems Conference*, DOI: 10.1109/ES.2013.6690080 (2013)

# STEM: A Simulation-Based Testbed for Electromagnetic Big Data Management

Mengyuan Lyu<sup>1,3</sup>, Peiquan Jin<sup>1,2</sup>, Zhou Zhang<sup>1,2</sup>, Shouhong Wan<sup>1,2</sup>, Lihua Yue<sup>1,3</sup>

<sup>1</sup>School of Computer Science and Technology, University of Science and Technology of China

<sup>2</sup>Key Laboratory of Electromagnetic Space Information, Chinese Academy of Sciences

<sup>3</sup>Science and Technology on Electronic Information Control Laboratory, Chengdu 610036, China  
Hefei, China

lmys@mail.ustc.edu.cn, jpq@ustc.edu.cn, zzwolf@mail.ustc.edu.cn, {wansh, llyue}@ustc.edu.cn

**Abstract**—With the development of networked radars and wireless communication technologies, electromagnetic data is becoming a new type of big data. Compared with other kinds of big data, such as Internet big data, financial big data, and healthcare big data, electromagnetic big data has some special properties. For example, they usually contain rich and varying labels that describe the features of electromagnetic space. However, existing big-data benchmark tools cannot support the generation and test of electromagnetic big data. In this paper, we aim at providing a simulation-based testbed for electromagnetic big data. The testbed, named STEM, can simulate real-world electromagnetic big data. It supports generating real-time electromagnetic data streams with varying labels. In addition, it is reusable, reconfigurable, and flexible for users to generate workloads for different scenarios. After a brief introduction on the architecture of STEM, we present the implemental details of STEM. Then, we present a case study as well as performance evaluation to demonstrate the usability and flexibility of STEM.

**Keywords**- *Electromagnetic big data; Testbed; Simulation*

## I. INTRODUCTION

According to an IDC report, the global data will grow to 163 zettabytes by 2025, which is ten times to the 16.1 zettabytes of data generated in 2016 [1]. The increasing of data volumes in different areas, such as web search [2], social networks [3], moving objects databases [4-5], and electromagnetic spaces, leads to the big data era. Electromagnetic big data is a new type of big data, which is advanced with the development of electromagnetic space networks, networked radars, and wireless communication technologies. Compared with other kinds of big data, such as Internet big data, financial big data and healthcare big data, electromagnetic big data has its own characteristics. First, it contains rich label information, e.g., time and location labels, which is very important for the analysis and use of the data. Besides, unlike the common multi-sensor data whose file size is dozens of kilobytes [6], electromagnetic big data is a kind of single-sourced big data, meaning that each electromagnetic device can produce high-speed data streams with a large volume of data. In an electromagnetic space network, multiple radar devices can simultaneously generate high-speed electromagnetic data flows up to 100Gbps. To address these challenges, a testbed for accurately evaluating the performance

of electromagnetic big data related approaches has been a critical and important issue.

Existing benchmarks, such as YCSB [7], BigDataBench [8] and BigBench [9], have their own data generators. However, the generated benchmark data cannot exhibit the characteristics of electromagnetic big data. As a result, it is difficult to accurately measure the performance of electromagnetic big data. Another problem in testing electromagnetic big data is that we lack real environments. So far, it is much hard to construct a real electromagnetic space network that contains multiple satellites and radar devices. Moreover, a specific electromagnetic space network is not able to support the diversity of test requirements.

In this paper, aiming to build a reconfigurable and flexible environment for electromagnetic big data researches, we propose a simulation-based testbed called STEM (*Simulation-based Testbed for Electromagnetic big data Management*). STEM is designed to be reusable and flexible to allow users to customize different electromagnetic environments. In addition, it provides a friendly user interface. The main contributions of the paper are summarized as follows:

(1) We present a simulation-based testbed named STEM for electromagnetic big data. STEM can simulate real-world radar echoes and integrate the characteristics of electromagnetic big data into the simulation process.

(2) The proposed STEM provides a user-friendly interface and flexible configuration options for data generation, which help users to modify field data (such as labels) to satisfy new requirements and customize the electromagnetic environment.

(3) We present a case study of STEM on MongoDB [10] to demonstrate its reusability and configurability.

The remainder of the paper is structured as follows. Section II introduces related work. Section III presents the architecture of STEM. In Section IV, we show the key technologies of STEM. Section V presents a case study of STEM on MongoDB. And finally we conclude the paper in Section VI.

## II. RELATED WORK

Many existing big data benchmark tools have their own data generators. Some of data generators are extensible, while others are inextensible. For example, LinkBench [11] uses an inextensible data generator which is designed to generate synthetic data with similar characteristics to real social graph data, while BigDataBench uses an extensible data generator

named BDGS [12], which contains a text generator, a graph generator and a table generator. No matter using which kind of data generator, they both have pros and cons.

Benchmarks with inextensible data generator usually can produce meaningful results for specific application, such as HiBench [13] (for Hadoop system), LinkBench and TPC-DS [14] (for RDBMS). None of them can generate data with similar characteristics to electromagnetic data. For example, HiBench contains eight workloads, which can be classified into four categories: Micro Benchmarks, Web Search, Machine Learning and HDFS Benchmarks. LinkBench, as mentioned before, can generate social graph data like Facebook network. TPC-DS implements a multi-dimensional data generator—MUDD, which is designed for structured data type. For these application-specific limitations, some researchers turn to extensible data generator.

Though extensible data generator can produce various kinds of big data, it still has limitations. First, the data's veracity is questionable. For example, PGDF [15], a parallel data generation framework applied in BigBench, uses distribution functions to generate distributed big data. But its distribution functions' data source is provided by RNG (Random Number Generator). Second, the lack of flexibility is also a common issue. Besides PGDF, another data generator—BDGS, which is implemented on BigDataBench, uses the data model derived from real data sets. But electromagnetic big data contains a wealth of label information and the labels can be changeable for fitting different requirements. A data generator like BDGS, would lead to more redundant work on deriving data models. Another example is YCSB, a popular benchmark for NoSQL system, also have two limitations mentioned above. Its workloads consist of records, each with several fields which are random strings of ASCII characters and it also need to modify the data configuration every time, which is inconvenient.

An ideal way for electromagnetic big data tests is to run tests in a real electromagnetic environment. However, it is costly and hard. Instead, a simulation-based testbed is more suitable for electromagnetic big data researches, due to its flexibility and reconfigurability. Compared with previous methods, our proposal can simulate real-world radar echoes. In addition, it provides a user-friendly interface and flexible configuration options for users to customize the electromagnetic environment. To the best of our knowledge, our proposal is the first testbed that supports the simulation of electromagnetic big data.

### III. ARCHITECTURE OF STEM

Fig. 1 shows the architecture of STEM. It mainly consists of three modules: the EES (*Electromagnetic Environment Setting*) module, the SDT (*Simulation and Data Testing*) module, and the TEM (*Tools for Easy Management*) module. The detailed architecture is described as follows.

**The EES Module.** This module is designed to customize electromagnetic environments. It contains three components: Target Setting, Label Setting, and Device Setting. Every component is designed to be flexible and configurable. Users can add, delete, or edit elements within each component. The Target Setting component provides two operational modes, namely automatic generation and manual setting. This makes the system

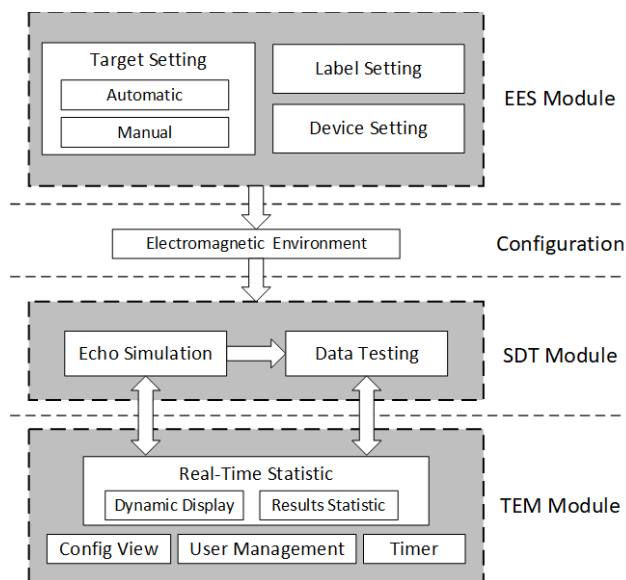


Figure 1. Architecture of STEM

easier for users to operate. After setting up the three components, the *EES Module* can generate a configuration of electromagnetic environment for the *SDT Module*.

**The SDT Module.** This module includes two sub-modules: the Echo Simulation module and the Data Testing module. The Echo Simulation module plays the role of data generator. It can generate electromagnetic data based on the configuration which is set by the EES Module, and then send them to the Data Testing module. The Data Testing module is designed to evaluate the performance of a database in electromagnetic big data. Currently, we've implemented it on MongoDB, which is the most popular NoSQL database system according to the DB-Engines Ranking [16]. Testing results and the data generated by Echo Simulation module will be sent to the *TEM Module* for visualization.

**The TEM Module.** This module provides some management tools for easy use. For example, the Real-Time Statistic in the *TEM Module* can display the results produced by the *SDT Module* in line chart and numbers. The line chart can clearly reflect the trend of the results, while the numbers can accurately show the value. Meanwhile, the Real-Time Statistics can convert the electromagnetic data into signal waveform synchronously in order to express it more vividly. In addition, we design some other management tools, such as Config View, User Management and Timer. Config View can help users check the configuration. User Management divides users into administrators and regular user, and grants different authorities. Timer can set the running time of whole system for saving users' time.

### IV. IMPLEMENTATION OF STEM

In this section, we describe the implemental details of STEM.

#### A. Implementation of the EES Module

The EES Module is a software layer that simulates the real electromagnetic environments. We divide it into three parts

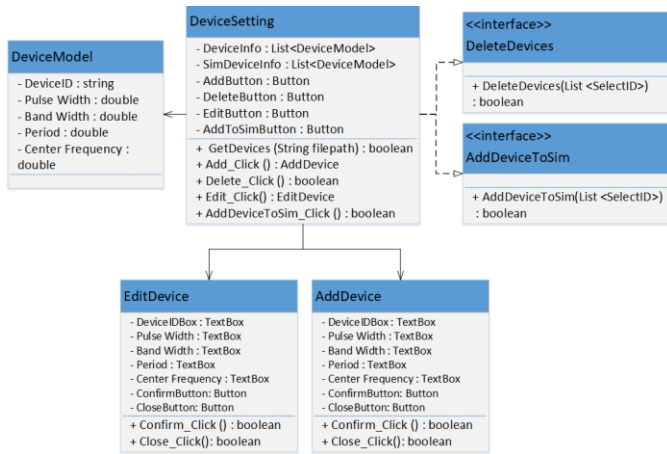


Figure 2. The class diagram of Devices Setting

according to the principle and characteristics of electromagnetic data, which are *Devices Setting*, *Targets Setting*, and *Labels Setting*. These components provide functionalities with similar interfaces for users to customize the environments. We are going to take the Device Setting as an example to introduce the similar parts of these interfaces. Then, we describe the differences between the Labels Setting and Targets Setting.

**Devices Setting.** Fig. 2 shows the class diagram of Device Setting. In our models, a radar device is represented as the following structure:  $\{DeviceID, Pulse\ Width, Band\ Width, Period, Center\ Frequency\}$ . The DeviceID is used to facilitate access to specific device. Pulse Width, Band Width, Period and Center Frequency are the parameters of a device, which generally won't change. In order to prevent these invariant parameters from being transmitted over and over again, we put device information into the database and use "DeviceID" to replace these parameters during transmission. As for some parameters that can be changed during the work, such as pitching angle and azimuth angle, we treat them as labels and put them in Label Setting module.

In order to make the module flexible and configurable, we offer some interfaces for users to construct they own devices. *AddDevice()* and *EditDevice(SelectedID)* provide a new graphical interface for user input, The interfaces and the descriptions are shown in Table 1.

Each interface provides guidance for easy use and can handle users' operation errors, like *DeviceID* is repeated or didn't select the devices to delete. The other two sub-modules offer similar interfaces, like *AddTarget()*, *DeleteLabel(List <SelectedID>)* and so on. The differences will be discussed in their own sections.

**Targets Setting.** Fig. 3 gives a simple class diagram of Target Setting. Considering that there may be a lot of targets needed to be set in one experiment, so we offer two operations mode: Manual and Automatic. The Manual mode is similar to what we use in Devices Setting. It involves the following interfaces like *AddTarget()*, *DeleteTarget(List <SelectedID>)* and so on. If there is no requirement on simulating specific targets, we can leave this to the Automatic mode, which can automatically generate the targets.

TABLE I. DEVICE INTERFACES AND DESCRIPTIONS

Interface	Description
<i>AddDevice()</i>	Create a new radar device
<i>DeleteDevice(List &lt;SelectedID&gt;)</i>	Delete the radar devices selected by users
<i>EditDevice(SelectedID)</i>	Edit a radar device selected by users
<i>AddDeviceToSim(List &lt;SelectedID&gt;)</i>	Add the selected devices into the simulation list

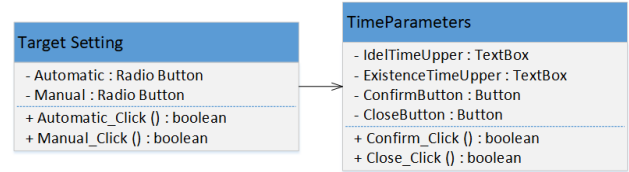


Figure 3. The class diagram of Targets Setting

For the sake of simplicity, each target is modeled as a point, which means that we ignore its size and shape. Thus, the model we use to describe a target can be expressed as  $\{TargetID, Distance, Scattering\ Coefficient, Velocity\}$ . In our previous design, we also need to set the appearance and disappearance timing of the target. But it will bring much inconvenience. For example, suppose that an identical target appears intermittently within the radar field of view, we have to repeatedly set its appearance and disappearance timing while other parameters remain the same. Considering this, we remove these two parameters from the target model and offer other two parameters needed to be set only once for users: Idle Time Upper and Existence Time Upper. The Idle Time Upper defines the maximum time between the disappearance of the last target and the occurrence of next target. And the Existence Time Upper defines the maximum time that a target can be exposed within the radar field of view. Therefore, the appearance and disappearance timing of the target are determined by system while users only need to define the boundaries.

**Labels Setting.** The rich label information is one of the key characteristics of electromagnetic data. The labels can be divided into two categories: some valuable information for the analysis of the electromagnetic data, such as the time and space label, and the parameters that can be changed during the working of a radar device, as mentioned in the Device Setting of this section.

A label can be simply described as:  $\{LabelID, Description, Bytes, Count\}$ . Similarly, we design some interfaces like *AddLabel()*, *DeleteLabel()*, *EditLabel()* and *AddLabelToSim()* for user-friendly extension. The difference is that we add a little limitations due to the importance of the time and space labels, for example, the time and space labels cannot be deleted and they are added to the simulation list by default even if not be selected.

Now we can give a general introduction to the final data structure. The electromagnetic data model can be divided into four parts: *Labels, IdBytes, DeviceID* and *Amplitude*. Fig. 4 gives an example of electromagnetic data.

As mentioned in the Devices Setting of this section, the elements we use to represent a radar device generally remain

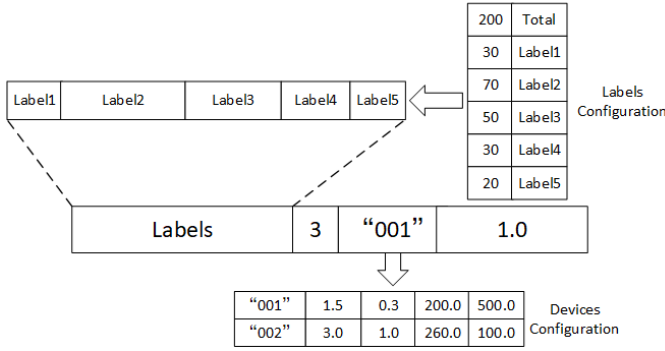


Figure 4. An example of electromagnetic data

stable. Thus, in order to prevent these invariant parameters from being transmitted, we use the *DeviceID* like "001" to replace these parameters for saving space and maintain a *Devices Configuration* in the database for accelerating the search on devices. We also use *IdBytes* to point out the bytes it takes for reading it later, which equals 3 in Fig. 2. *Amplitude* is calculated by the *SDT Module*, we will introduce it in the next section.

The label part is represented by a byte array. After setting up the Labels Setting module, the format of labels will be applied to every piece of data generated by devices. Through this way, we do not need to record the bytes for every label, i.e., the *Labels Configuration* is space efficient, which is similar to the *Devices Configuration*. In the example shown in Fig. 2, we can see from the *Labels Configuration* that the first 200 bytes are the label parts, containing five labels which occupy 30 \* 2, 70, 50 and 20 bytes, respectively. The next four bytes are an integer with the value of 3, which means that the next three bytes are the *DeviceID*. The rest is the *Amplitude*.

### B. Implementation of the SDT Module

The *SDT Module* is the core module of STEM. The most important component in the SDT module is the echo model. Next, we describe the echo model as well as the echo simulation process.

**Echo Model.** In the design of radar echo model, we have borrowed some ideas and methods in the [17]. We select the LFM (linear frequency modulation) pulse as the simulation object, which is the most widely used pulse compression signal and not sensitive to the Doppler frequency shift. For the LFM pulse, the radar emit signal can be expressed as in (1), where *rect()* is a rectangle function which defined in (2), *T* represents pulse width, *f<sub>c</sub>* is center frequency, *K* is frequency modulation rate and *B* is band width.

$$S_E(t) = \text{rect}(t/T) \exp[j2\pi(f_c t + Kt^2/2)], \quad K=B/T \quad (1)$$

$$\text{rect}\left(\frac{t}{T}\right) = \begin{cases} 1 & |t/T| \leq 1/2 \\ 0 & |t/T| \geq 1/2 \end{cases} \quad (2)$$

Suppose that there is a point target approaching a radar device whose distance to the target is *R<sub>0</sub>*, at a radial velocity *v*.

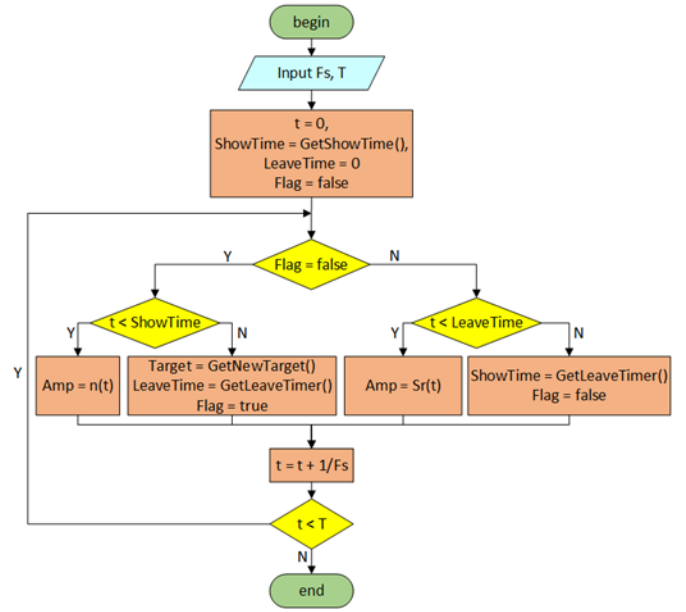


Figure 5. An example of Echo Simulation Process

The pulse repetition frequency of radar is set as *T<sub>r</sub>*. When the *n*th pulse is emitted, the distance between the target and the radar can be expressed in (3).

$$R^n(t) = R_0 - vnT_r, \quad n = \lfloor t/T_r \rfloor \quad (3)$$

Then, we define the echo by (4). Here, the symbol *A* represents the scattering coefficient. *τ(t)* is time delay function and *n(t)* is a random function for simulating the environment noise.

$$S_R(t) = AS_E(t - \tau(t)) + n(t), \quad \tau(t) = 2R^n(t)/(c + v) \quad (4)$$

**Echo Simulation.** Next, we give the process of the echo simulation process. Fig. 5 gives an example with the Timer in the TEM Module and with the choice of Automatic mode in Target Setting of the EES Module. If no Timer is used, the end of system is controlled by the *Pause* Button. If choosing Manual mode in Target Setting, we do not need to get new target during the process.

The parameters we need to input into the system is *F<sub>s</sub>* (sampling frequency) and *T* (time limit set by the Timer). *GetShowTime()* and *GetLeaveTime()* are designed to calculate target's appear and disappear time based on the *Idle Time Upper* and the *Existence Time Upper*, which are set in the *Targets Setting*. *Flag* is used to determine if there is a target.

The process in Fig. 5 can be described as follows:

- (1) *LeaveTime* and *t* (represents time) is set to zero, *Flag* is set to false and *ShowTime* is calculate by *GetShowTime()*.
- (2) If *Flag* = false, which means there is no target, go to (3), otherwise it means that target appears, go to (4).
- (3) If *t* < *ShowTime* (the next target has not appeared yet), the echo that radar has received is just the noise, so *Amplitude* is determined by *n(t)*. Otherwise it means *t* is equivalent to or over *ShowTime*, which also means that there would be a new target in the radar range immediately. Thus, we need to design this new target



by using  $GetNewTarget()$ , calculate its  $LeaveTime$  and set  $Flag$  to true. Then, we go to (5).

- (4) If  $t < LeaveTime$ , which means the target is still in the radar range, the echo is calculate by  $S_R(t)$ . Otherwise it means that this target will disappear. In such a situation, we need to calculate the next target's appear time and set  $Flag$  to false. Then, we go to (5).
- (5)  $t = t + 1/F_s$ .
- (6) If  $t < T$ , return to (2), otherwise the process is ended.

## V. EVALUATION

In this section, we describe a case study of using STEM for evaluating the storage performance on MongoDB in electromagnetic big data. The results show that STEM is easy to reconfigure to simulate different electromagnetic environments and evaluate the performance in electromagnetic big data.

### A. Experiment Setup

The experiment is based on the network architecture as shown in Fig. 6. It consists of five server nodes and several client nodes. The server nodes constitute a distributed storage cluster for simulating the data receiver and each server node is equipped with Intel 2.1GHz dual CPU, 128GB DDR4 memory, twelve 4T 7200RPM SAS drives and two 240GB SSDs. The client nodes are common computers, mainly used for data acquisition. The server nodes and the client nodes are connected by ten gigabit LAN.

Our STEM runs on the clients of Data Acquisition. First, the client nodes run STEM to set the environment. Then, all the client nodes run the  $SDT$  Module at the same time to generate data and transmit the data to the server nodes. The states of the servers are monitored by STEM.

### B. Use Cases of STEM

We present two cases of using STEM. In the first case, the electromagnetic environment set by each client node is ten radars, automatic targets and 300 labels (each is 50 KB, the file size limit in MongoDB is 16 MB), we test the throughput rate of MongoDB under different number of processes.

The second one is designed to test the throughput of MongoDB under different numbers of labels. Its environment is set to ten radars, automatic targets and ten processes. For the sake of simplicity, all the labels are set to having the same size (50 KB). Both experiments are run for five minutes.

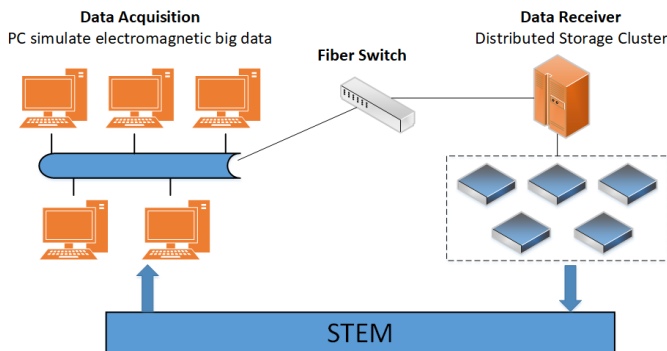


Figure 6. Experiment Network Deployment Architecture

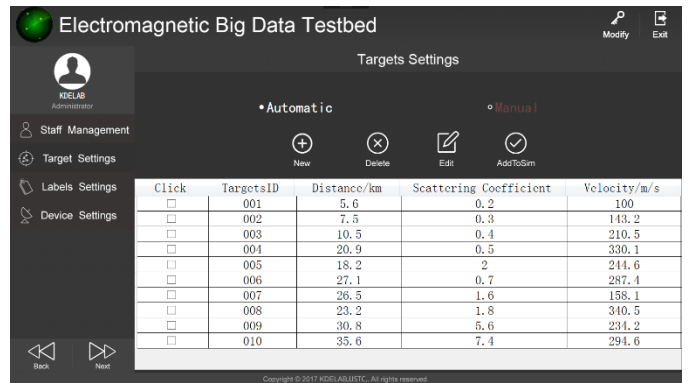


Figure 7. The setting interface of STEM

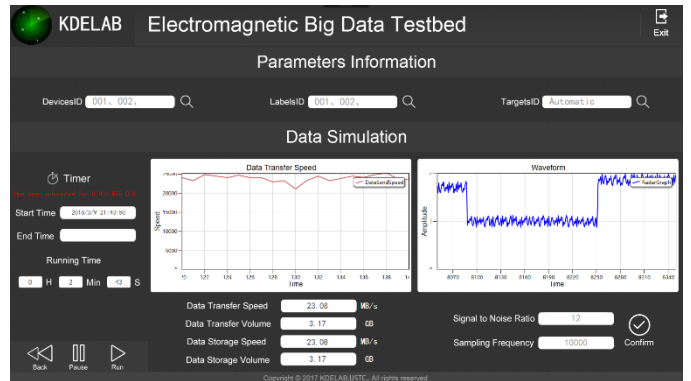


Figure 8. The running screenshot of STEM

The experimental process consists of three steps:

- (1) Every client node finish the experiment environment setting, click the Next button to the running page;
- (2) Every client node sets the Timer to five minutes, then click the Run button to run the system;
- (3) Just wait for five minutes and click the Back button to the setting page, reset the environment setting, and back to (2).

Fig. 7 shows the setting page of STEM, and Fig. 8 shows the running screenshot of STEM.

### C. Results

Fig. 9 shows the highest throughput (top throughput) as well as the average throughput under different processes. We can see that when the number of processes reaches 13, the top throughput reaches 1126 MB/s, which is close to the upper-bound of the network speed. Basically, the top throughput increases linearly with the increase of the process and the average throughput reaches the bottleneck when the number of processes reaches 11, which is nearly 788 MB/s.

Fig. 10 shows the throughput under different numbers of labels when the number of processes is 10. The average throughput increases with the increasing of the numbers of labels. On the other hand, the top throughput does not show an apparent trend. Specially, the top throughput reaches 1105 MB/s when the number is 20, which is close to the result of 11 processes and 300 labels. A possible reason is that the small numbers of labels make the number of files that can be generated and inserted per second increase, as shown in Fig. 11.

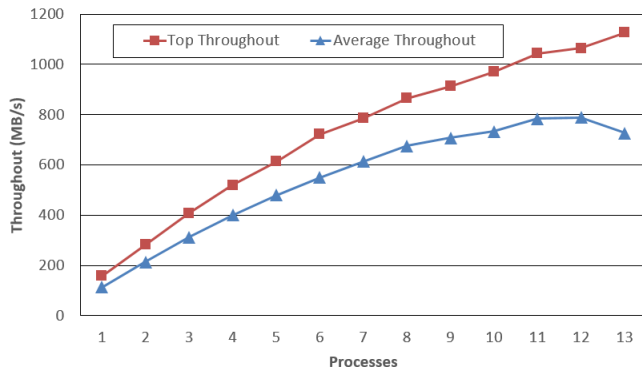


Figure 9. Throughput under different processes

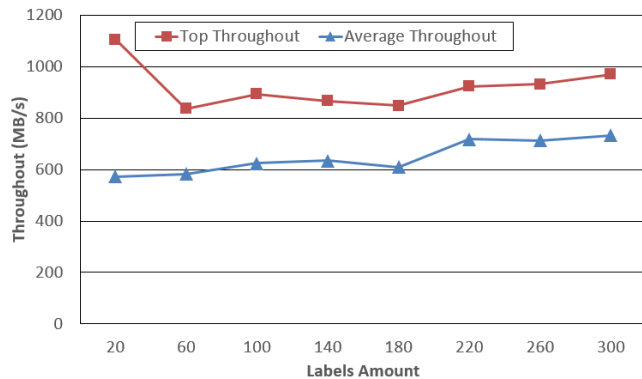


Figure 10. Throughput under different size of labels

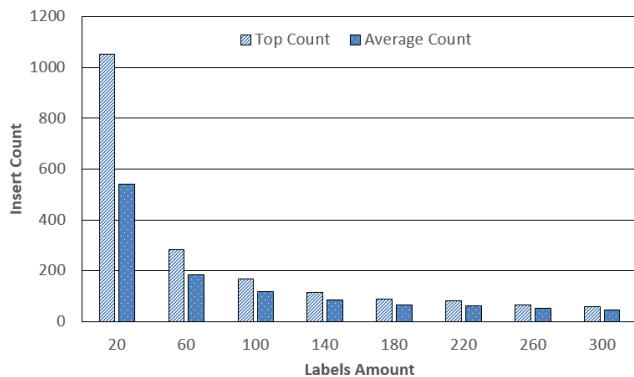


Figure 11. Insert Count under different size of labels

## VI. CONCLUSIONS AND FUTRUE WORK

In this paper, we propose a simulation-based testbed for electromagnetic big data researches, which aims to provide effective support for evaluating the performance of electromagnetic big data management, such as throughput test and real-time processing measurement. The proposed testbed is designed to be reusable, flexible, and reconfigurable. Currently, STEM is able to evaluate the storage performance of electromagnetic big data on MongoDB, such as the test of write throughput, the average throughput, and the insert counts per second.

Our future work will concentrate on supporting multiple types of databases, offering more kinds of performance tests,

and improving the efficiency of data generation. We plan to support HBase, Cassandra and other popular database systems in future. In addition, we will adopt other performance tests like query processing to make the testbed multifunctional and the parallel computing on every radar to make data generation more efficient.

## ACNOWLEDGEMENT

This work is supported by the National Science Foundation of China (61672479 and 61472376), and a fund from the Science and Technology on Electronic Information Control Laboratory. Peiquan Jin is the corresponding author.

## REFERENCES

- [1] D. Reinsel, J. Gantz, and J. Rydning, "Data Age 2025: The evolution of data to Life-Critical," <https://www.seagate.com/files/www-content/our-story/trends/files/Seagate-WP-DataAge2025-March-2017.pdf>, Accessed in April 2018.
- [2] J. Zhao, P. Jin, Q. Zhang, and R. Wen. "Exploiting location information for Web search". *Computers in Human Behavior*, 2014, 30: 378-388.
- [3] L. Zheng, P. Jin, J. Zhao, and L. Yue. "A fine-grained approach for extracting events on microblogs", *International Conference on Database and Expert Systems Applications (DEXA)*, 2014, pp. 275-283
- [4] P. Jin, L. Zhang, J. Zhao, L. Zhao, and L. Yue, "Semantics and modeling of indoor moving objects", *International Journal of Multimedia and Ubiquitous Engineering*, 2012, 7 (2): 153-158
- [5] C. Huang, P. Jin, H. Wang, N. Wang, S. Wan, and L. Yue. "IndoorSTG: A flexible tool to generate trajectory data for indoor moving objects.", *2013 IEEE 14th International Conference on Mobile Data Management (MDM)*, 2013, pp. 341-343
- [6] X. Hao, P. Jin, and L. Hua, "Efficient Storage of Multi-Sensor Object-Tracking Data," *IEEE Transactions on Parallel and Distributed Systems*, 2015, 27(10): 2881-2894
- [7] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with YCSB," *ACM Symposium on Cloud Computing (SoCC)*, 2010, pp. 143-154.
- [8] L. Wang, J. Zhan, C. Luo, Y. Zhu, Q. Yang, et al, "Bigdatabench: a big data benchmark suite from internet services," *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2014, pp. 488-499.
- [9] A. Ghazal, T. Rabl, M. Hu, F. Raab, M. Poess, et al, "BigBench: towards an industry standard benchmark for big data analytics," *ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 2013, pp.1197-1208.
- [10] MongoDB. <https://www.mongodb.com/>.
- [11] T. G. Armstrong, V. Ponnkanti, D. Borthakur, M. Callaghan, "LinkBench: a database benchmark based on the Facebook social graph," *ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 2013, pp.1185-1196.
- [12] Z. Ming, C. Luo, W. Gao, R. Han, Q. Yang, et al, "BDGS: a scalable big data generator suite in big data benchmarking," *Workshop on Big Data Benchmarks*, 2013, pp. 138-154.
- [13] S. Huang, J. Huang, J. Dai, T. Xie and B. Huang, "The HiBench benchmark suite: characterization of the MapReduce-Based data analysis," *IEEE International Conference on Data Engineering (ICDE) Workshops*, 2010, pp. 41-51.
- [14] R. O. Nambiar and M. Poess, "The making of TPC-DS," *International Conference on Very Large Databases (VLDB)*, 2006, pp. 1049-1058.
- [15] T. Rabl, M. Frank, H. M. Sergieh and H. Kosch, "A data generator for cloud-scale benchmarking," *Technology Conference on Performance Evaluation and Benchmarking*, 2010, pp. 41-56.
- [16] DB-Engines Ranking. <https://db-engines.com/en/ranking>.
- [17] M. A. Richards, "Fundamentals of Radar Signal Processing, Second Edition", 2017.

# Towards Reference Architecture for a Multi-layer Controlled Self-adaptive Microservice System

Peini Liu, Xinjun Mao, Shuai Zhang, Fu Hou

College of Computer  
National University of Defense Technology  
Hunan, China 410073

Email: peini.liu@foxmail.com, xjmao@nudt.edu.cn, zhangshuai16a@nudt.edu.cn, houfu@nudt.edu.cn

**Abstract**—With the features of high distribution in deployment and independence in running, the microservice systems that operate in heterogeneous infrastructures and open Internet environment are expected to be self-adaptive to adapt to various changes of both operating contexts and application requirements. This requires the adaptability of the microservice systems to be diverse and flexible, and independent of implementation technologies and platforms. This paper presents a reference architecture for self-adaptive microservice systems with the abilities of multi-layer controlled self-adaptations, including infrastructure-controlled layer and application-controlled layer. Such reference architecture presents a blueprint to cope with diverse changes from different levels in microservice systems and supports the interactions between layers. We have implemented a practical platform called *SAMSP* based on the reference architecture and Kubernetes and evaluated our approach using a sample. The experimental results are promising, and demonstrate the feasibility and effectiveness of our proposed reference architecture.

**Keywords**—microservice system; reference architecture; self-adaptive microservice system; multi-layer control loops

## I. INTRODUCTION

Microservice, a popular architectural style, attracting more and more attention in both academia and industry areas, is widely adopted now by many large companies such as Amazon [1], Netflix [2], LinkedIn [3]. This architecture style is considered to be the best efforts for cloud computing and service-oriented system engineering [4].

In the early years of service-oriented architecture, the monolithic architectural style has been an approach to build web applications. These applications were built as a single unit, and all the logic for handling a request runs in a single process [5]. As system under this architecture is lack of independence and flexibility, services have to get scaled and evolved together, which result in a huge waste of server resources. Hence, the monolithic architectural style is not suitable enough to construct ultra-large-scale information system anymore. To overcome the challenges, microservice has become a new architectural style to build such complex system [6]. It decomposes a large complex software application into a suite of small services, with each service running in its own process and communicating by lightweight mechanisms [7]. The microservice architecture style brings many benefits for service-oriented systems, such as

scalability [8], functional separation [9], loose coupling [7] and fast delivery [10].

However, microservice system still faces several challenges. Firstly when turning into microservice, since these highly distributed microservices often run on containers deployed on cloud and are organized to realize an application, we have to take a software architecture with adaptability. Secondly, as the microservice system is evolving because of changes of context and requirements, system must adapt to handle the challenges. In addition, the agile and DevOps methodology expect the system runs without downtime and integrates continuously. Therefore, system is no longer running in a known context and with static requirements, meaning that the system needs to reconfigure and restructure themselves to meet the dynamic changing world [11].

Obviously, it is infeasible for operators to take all the changes into consideration and manually control such system. However, self-adaptive system brings some inspiring approaches to adapt system at runtime, which help to preserve and optimize the system's operation in dynamic changes [12, 13]. For example, Rainbow [14], an architecture-based approach, allows the self-adaptive system to be aware of the software structure and drives the self-adaptation by the external control. Another reflection approach can use the reflected ability of software to examine and possibly modify its structure (structural reflection) or behavior (behavioral reflection) at runtime [15]. These two methods inspire us to build MAPE control loops as an autonomic manager to monitor the states of microservice system, to analyze the changing and to plan and execute the actions at runtime [16].

Our work presents a reference architecture for a multi-layer controlled self-adaptive microservice system. The self-adaptation idea comes from self-adaptive systems which can manage system itself according to the high-level goals [16]. The reference architecture makes microservice system to be aware of its dynamic contexts continuously and the changing requirements in order to adjust its behavior and structure at runtime. The innovation of this architecture is using autonomic computing MAPE control loop to form multi-layer control loops. On the one hand, self-adaptive microservice system might run on third-party provided infrastructure servers, and the system workload at the infrastructure layer has to be adapted through infrastructure-controlled loop. On the other hand, self-adaptive

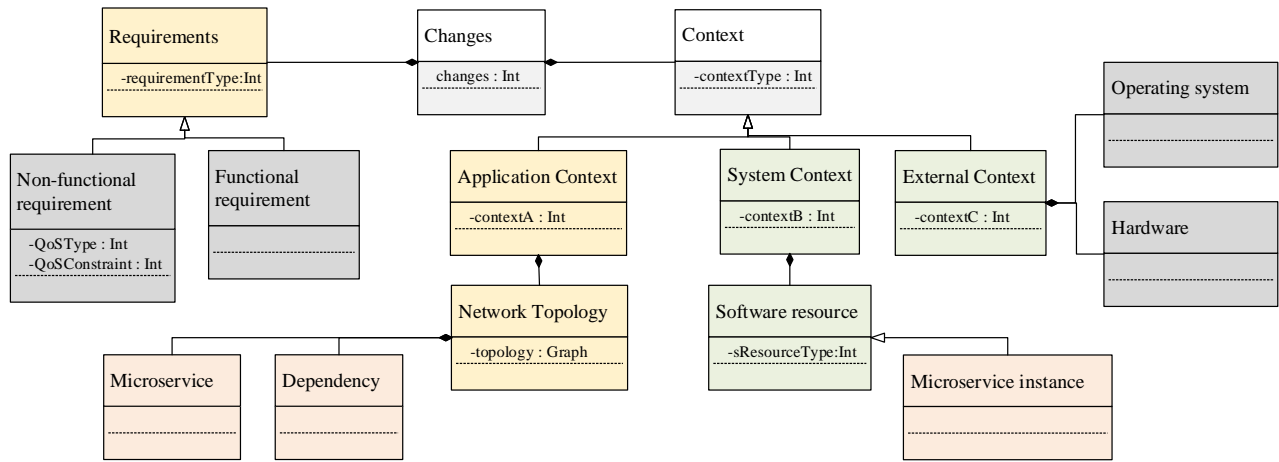


Figure 1. Diversity of changes in self-adaptive microservice system.

microservice system has multiple microservices organized to realize an application, and the applications' requirements and performance at the application layer have to be assured through application-controlled loop.

The remainder of this paper is organized as follows: Section 2 analyses the diverse changes and multiple layers in self-adaptive microservice system. Section 3 proposes a reference architecture for a multi-layer controlled self-adaptive microservice system; Section 4 presents a practical implementation of the reference architecture; a sample and a promising experiment based on reference architecture are given in Section 5; Section 6 compares with related work and section 7 concludes and discusses the future work.

## II. DIVERSE CHANGES AND MULTIPLE LAYERS IN SELF-ADAPTIVE MICROSERVICE SYSTEMS

Self-adaptive microservice system has challenges to facing the diverse changes. Inspired by Cherif Sihem et al.[17] who bring a context model of SOA that divided into several parts—Infrastructure, Platform, and Application. Figure 1 represents the diversity of changes in self-adaptive microservice system.

Changes in self-adaptive microservice system contain the context and the application requirements, each of which has its own rules and presents a part of self-adaptation. The former changes have three basic elements that provide different levels of context from different perspectives. For example, (1) external context is a part of the external world and includes hardware resource, operating system and other related system. System can sense these context but cannot directly control. (2) system context is internal system environment with software system resources like microservice instance. (3) application context includes system network topology, and they are concerned with specific application. The latter changes are from users and include application functional and non-functional requirements. For instance, (1) functional requirements are about the organization of applications. (2) non-functional requirements are the QoS related to the application performance.

Each part of changes has its own adaptation rules that can be conducted. However, from the result of adaptation, the system needs to achieve adaptation by reconfiguring microservice instances or restructuring the microservice topology. So we can

provide adaptation facilities isolated in the infrastructure and application two layers. Thus, a key challenge is how to structure and coordinate the two layers to handle the diverse changes.

To overcome the challenge, the structure with two control layers upon the microservice system is shown, and a conceptual model of multi-layer controlled self-adaptive microservice system is proposed in order to clarify the layers and their interactions. In Figure 2, the self-adaptive microservice system is composed of multi-layered control layers and the microservice system. Microservice system is a target system that achieves the function of business. Managing system has multiple controlled layers, and each of the layer manages different types of adaptation: (1) the infrastructure-controlled layer (ICL) senses the system context and external context to manage the containers with the platform predefined rules, system adaptation often appears as reconfiguration; (2) the application-controlled layer (ACL) senses the application related changes like requirements and application context, system adaptation can be restructured or with the lower control loops help.

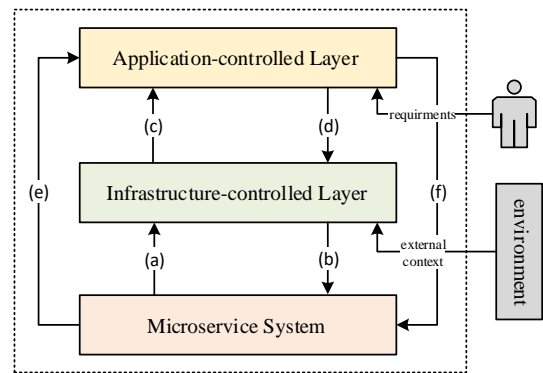


Figure 2. The conceptual model of multi-layer controlled self-adaptive microservice system.

## III. REFERENCE ARCHITECTURE FOR SELF-ADAPTIVE MICROSERVICE SYSTEMS

Nowadays, microservice system needs a reference to support continuous changes in context and requirements. Facing the challenge in Section II, in this section we proposed a reference architecture for self-adaptive microservice system as a guideline

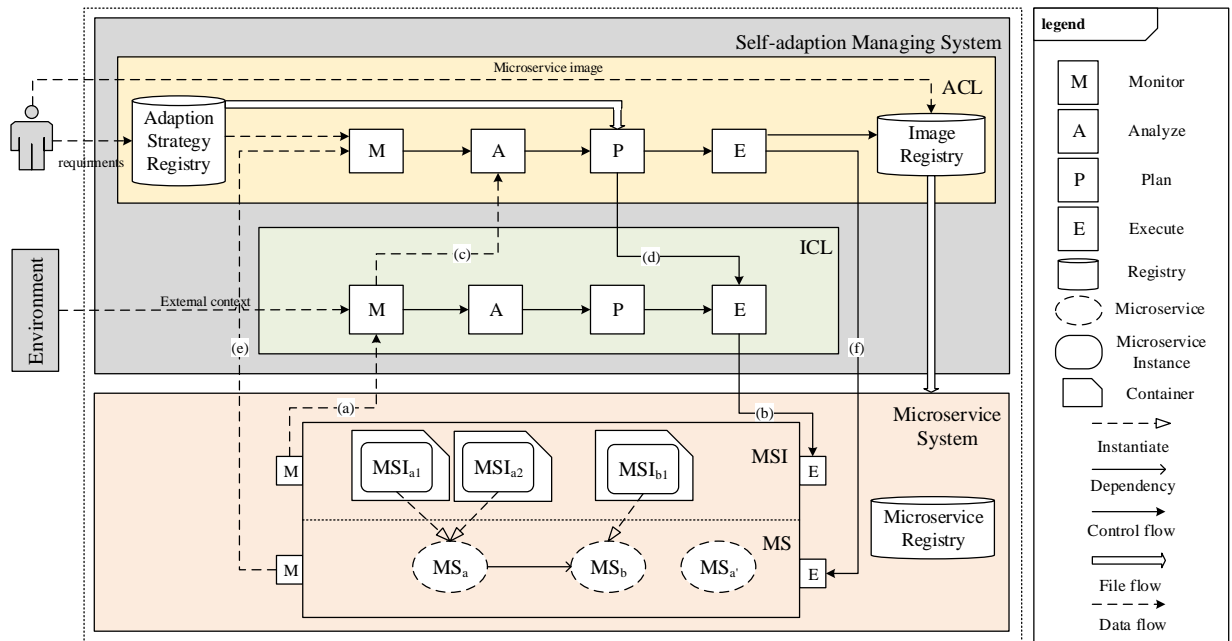


Figure 3. The reference architecture for multi-layer controlled self-adaptive microservice system.

for engineers on how to design, develop and adapt such systems in a specific domain. Our reference architecture is inspired by the control theory and find an effective instantiation –‘MAPE model’ to build control loops. In fact, our contribution is to design the architecture by considering these aspects explicitly: (1) decoupling the different layer control loops required to satisfy each part of changes; (2) achieving self-adaptation goals through the collaboration between two layers.

A detailed view of the reference architecture for multi-layer controlled self-adaptive microservice system is given in Figure 3. The reference architecture defines the functional elements, as well as the control, data or file interactions among the internal elements of each layer and the multiple layers. In addition, the reference architecture characterizes the collaboration among the multi-layer to assure that it can be applied partially in case someone does not need some parts of adaptation. These details are explained in the following sections.

#### A. Microservice system

As a target system, the microservice system consists of a set of microservices, which are organized in applications through lightweight protocols. There are two concepts in a microservice system: one is a microservice instance (MSI), which refers to a real entity that handles requests to accomplish the appropriate functions. The other is a microservice (MS), which can be understood as an abstraction of a set of microservice instances which exactly have the same capabilities.

At run time, the microservices are discovered by each other through the microservice registry. In particular, the microservice itself does not process the request but distribute the request to its corresponding microservice instances to perform the functions. The microservices instance is the smallest running unit that runs in a container and is deployed on cloud, giving us an inspiration to operate the container to manage microservices instances. Meanwhile, microservices also shield the operation details

through the abstract microservices interface and well maintain the topology of the application. Once the topology changes due to the change of the dynamic context or the change of applications’ requirements, the adaptive system can timely observe the microservices and dependencies between them to restructure the application. Therefore, the microservice system needs to sense the running status information of the microservice instance and the topology of the organized microservice application to determine whether the target system is healthy.

#### B. Infrastructure-controlled layer

The infrastructure-controlled layer (cf. ICL in Figure 3) solves the adaptive problem at the infrastructure level of the adaptive microservice system. It consists of a MAPE control loop: the *Monitor* senses the external context from the environment and the system context from the microservices instances (cf. Interaction (a) in Figure 3) and collects monitoring data. *Analyze* analyzes the system-related information, and triggers the system-level policy in *plan* by the event whether the context changes. Finally, *Execute* in the control loop adjusts the system configuration according to the policy, so that the system adaptation can be implemented by scheduling the place that containers deployed, scaling the number of containers and limiting or increasing the resources of the container (cf. Interaction (b) in Figure 3).

#### C. Application-controlled layer

The application-controlled layer (cf. ACL in Figure 3) as the upper layer of self-adaptive microservices system, also consists of a MAPE control loop: *Monitor* senses the context from the application organized by the microservices — the application topology (cf. Interaction (e) in Figure 3) or the requirement changes by users through adaptation strategy, and *Analyze* analyzes the application functional requirements information and triggers the strategy written by the application developer in *Plan* when the topology changes, e.g., changing the dependency

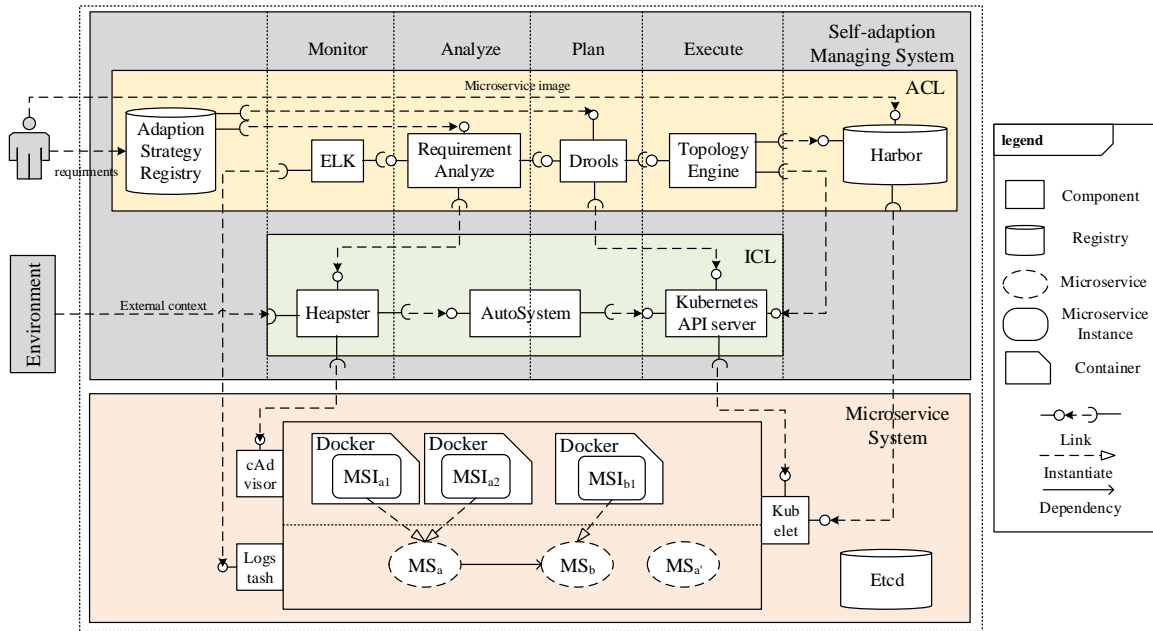


Figure 4. SAMSP - A practical implementation of the reference architecture.

between two microservices. Finally, *Execute* in the loop achieves the system adaptation according to the tactics that adjust the microservices organized in application and their related dependency (cf. Interaction (f) in Figure 3).

Moreover, ACL governs a part of changes with the collaboration of the ICL. We define the application-related runtime information at the ICL layer as variables to be controlled in ACL. In the circumstances, ICL *Monitor* uploads the related data to ACL *Analyze* (cf. Interaction (c) in Figure 3) to judge whether it satisfies the application non-functional requirement and decide to obtain the results in *Plan*. At last, ACL *Plan* sends the approach to ICL *Execute* (cf. Interaction (d) in Figure 3) to mentor the runtime adaptation.

#### IV. A PRACTICAL IMPLEMENTATION OF THE REFERENCE ARCHITECTURE

This section describes a practical implementation of the reference architecture that is built based on Kubernetes<sup>1</sup>. Our implementation provides an extended platform called SAMSP with a toolkit to realize the multi-layer self-adaptation. The whole implementation architecture of the reference architecture is depicted in Figure 4.

In the design stage, application logic and adaptive logic are separated. For one, we develop microservices with independent function and use jersey.jar to implement the Restful interaction protocol. After the development, we structure their environment through Docker<sup>2</sup> images and put them into Image registry Harbor<sup>3</sup>. For the other, some self-adaptation goals are considered by using self-adaptation strategy language to describe and registered in our adaptation strategy registry.

When it comes to runtime stage, in microservice system, microservice instantiate several microservice instances, running in containers and deployed on distributed cloud servers. We use a container orchestration Kubernetes to help us deploy our microservice instances containers, also, some plugins like Etd<sup>4</sup> which is used for microservice discovery. As an important role in monitoring system status, cAdvisor<sup>5</sup> collects the performance of the microservice instances, and the logstash obtains a calling chain in local.

In ICL, in the monitor stage, we use Heapster<sup>5</sup> to collect the status of clusters and microservices instances' performance like CPU usage, memory usage, etc. in local cAdvisor<sup>2</sup>. In analyze and plan stages, the Autosystem component that expanded from HPA (Horizontal pod autoscaler) in Kubernetes to analyze the status of the system and choose optimal values for the configurable parameters. In the execute stage, the Kubernetes API server hands out the new configuration parameters to the cluster kubelet to adapt the changing context.

In ACL, firstly, ACL needs to load the adaptation strategies from the adaptation strategy registry. In the control loop, internally, ELK<sup>6</sup> (ElasticSearch, Logstash, Kibana) are used to collect the organization of the application from local logstash. After the requirement check from requirement analyze, Droools<sup>7</sup> as our application rules engine will fire the self-adaptation strategy we have defined and use topology engine to change the dependency between the microservices or build a new structure. As for collaborating with ICL, the requirement analyze asks the application-related property from ICL and the Kubernetes API server obtains an application required results from Droools to operate the containers.

#### V. CASE STUDY AND EXPERIMENTS

To illustrate the feasibility of our proposed reference architecture and the effectiveness of its self-adaptation, in this section, we use a book information system (BIS) as a running

<sup>1</sup> <https://kubernetes.io/>

<sup>2</sup> <https://www.docker.com/>

<sup>3</sup> <http://vmware.github.io/harbor/>

<sup>4</sup> <https://coreos.com/etcd/>

<sup>5</sup> <https://github.com/kubernetes/heapster/>

<sup>6</sup> <https://www.elastic.co/products>

<sup>7</sup> <https://www.drools.org/>

example [18]. This application provides the book information support to help users to know the book through the internet. Basically, the BIS application shows the information of books by composing a book review microservice with a book content microservice. Meanwhile, the book contents can be provided by some different book content information providers, such as Wikipedia, Baidupedia and several school library systems.

### A. Sample development based on BIS

Here, the generic adaptation scenarios for microservice system are divided in two aspects, the infrastructure level adaptation and the application level adaptation. (see Table 1)

TABLE I. GENERIC ADAPTATION SCENARIOS OF MICROSERVICE SYSTEM

Layer	Type of changes	Scenarios
ICL	External context	S1: Microservice instance unavaliable
	System context	S2: Microservice instance overload
ACL	Application context	S3: Microservice unreachable
	Functional Requirements	S4: Application function enhancement
	Non-functional Requirements	S5: Application-related QoS constraints violation

A real BIS application has been set up based on our reference architecture. This BIS adopts microservice, and the ICL/ACL can be implemented exactly as Section IV. The ICL layer is used to enhance the reliability and the performance of the system by reconfiguring the microservice instances. It takes effect on S1 and S2 in Table 1. In S1, if one of these book review microservice instances is unavailable, the rest of the book review microservice instances need to accept the requests from the unavailable microservice instance, and to restart this failed microservice instance. (2) S2: if one of them is overloaded, reconfiguring the number of the microservice instances or scheduling it to an available server will achieve better system performance.

However, if all instances failed, the microservice will be unreachable. So the ACL is responsible to handle this through restructuring the organization of the application. For instance, in S3, if the book content microservice is unreachable, the ACL will be notified the situation. In response, this layer will register the alternative service (e.g. Wikipedia or Baidupedia) into the system, adjust the dependency between these related microservices and relink that service to the latest available microservices. For requirement changes, in S4, if the application needs a new function, e.g., book rating, ACL will handle the functional requirements by pushing a new microservice into image registry and a new configuration into the ICL to make use of the microservice that is newly added. Finally, as for the collaboration between the ACL and ICL, the scenario is that, if the user requires the average response time of the service to be no more than 1.5s (i.e. S5). To satisfy this, ACL will sense the average response time information from ICL and trade-off the planning when the condition has been violated, at last execute the plan in ICL.

### B. Experiments Analysis

To evaluate the self-adaptation effect and performance in the self-adaptive microservice system compared to the original microservice system without self-adaptation, we conduct an experiment on a distributed testbed Locust<sup>8</sup>. In this experiment, some of the adaptation scenarios from Table 1 have been selected (e.g. S1: Microservice instance failure, S2: Microservice instance overload, S5: Application-related QoS constraints violation). We observe the microservice’s average response time (ART) in 30min to evaluate the performance when microservice faces diverse changes.

Figure 5 shows the results of system performance with and without adaptation during the whole running time. The solid line shows that, without adaptation, once the average response time rises when facing the changes, it never falls again. On the other hand, the dashed line shows that if SAMSP works in the adaptation, average response time rises but soon return to optimal level. Table 2 shows the details of the changes in the periods, and also compared the results of our case. The results indicate that self-adaptation is effective and significantly improves system performance in this experiment.

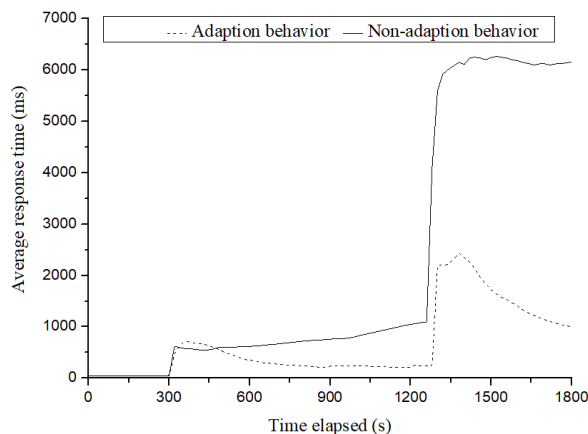


Figure 5. System performance with and without adaptation.

TABLE II. SCENARIOS PERIOD AND RESULTS OF THE CASE STUDY

Time (s)	Scenarios	ART without adaptation(ms)	ART with adaptation(ms)
0-300	System stable running	51.11	47.15
301-900	S2: Book information microservice overload	644.81	408.32
901-1200	S1: Book information microservice instance failure	895.51	231.26
1201-1800	S5: Book information microservice QoS constraints violation	5559.90	1454.25
Total time		2148.94	646.67

## VI. RELATED WORK

We find some work related to self-adaptive microservice system, and also we take a look at some models for designing a self-adaptive system and the microservice architecture nowadays. Some work is briefly described in this section.

<sup>8</sup> <https://www.locust.io/>

This research is supported by research grants from Natural Science Foundation of China under Grant No. 61532004 and 61379051.

## REFERENCES

An architecture for self-managing microservices is presented at [19]. It proposed a novel architecture that enables scalable and resilient self-management of microservices application on cloud. The main approach is using the algorithm to select a leader to assign management functionality to nodes and allowing atomic service to become self-managing. However, the distributed configuration management is much easier to conflict compared to centralized architecture and cannot implement expensive algorithms to elect leaders. A reference architecture from Krasimir based on SOA and autonomic computing [20] provides a way to transform the microservice instances by adding an autonomic manager into a part of the service and build an adaptation registry. However, in this approach, microservice instances need to be transformed into an intelligent instance by weaving code, which is quite difficult and expensive to implement. In [21], Namiot provides an overview of microservices architecture and implementation pattern. However, it treats microservices as components and analyses its communications in a design view. An autonomic computing system supports a continuous process, J. Kephart in [22] discuss a view of autonomic computing that has four elements: Monitor, Analyze, Plan and Execute (MAPE). This model reflects a general method for self-adaptive system.

Our research is different from previous work among the following: (1) We analysis the diversity of changes and divide adaptation into different layers; (2) We present a reference architecture for self-adaptive microservice system with the multi-layer control loops; (3) Our research ensures the coherence between the reference architecture and implementation and deploys a case in real system.

## VII. CONCLUSION AND FUTURE WORK

Microservice system is distributed and deployed on the cloud, which often uses container technology. It needs the capability of self-adaptation to face the diverse changes and challenges. Our contribution is discussing this question from an architecture perspective, and presenting a reference architecture for a multi-layer controlled self-adaptive microservice system, which constitutes a guide to design self-adaptive microservice systems.

Our contributions are threefold: (1) designing a novel reference architecture for multi-layer controlled self-adaptive microservice system, which decouples different layer control loops required to satisfy different parts of changes, and achieves self-adaptation goals through reconfiguring or restructuring the microservice system at runtime; (2) presenting an implementation architecture of microservice systems based on our reference architecture and K8S, and providing an extended platform *SAMSP* with a toolkit; (3) validating our proposed reference architecture in term of sample development and experiments, and showing the feasibility and effectiveness of architecture and platform for self-adaptive microservice systems.

For further research, we would like to improve our reference architecture in the following aspects: (1) enhance the self-adaptation managing system to provide more common self-adaptive abilities; (2) design a context model and relation model for microservice systems as self-adaptation knowledge to improve the control loop; (3) integrate accurate and efficient algorithms to choose adaptation strategies optimally.

- [1] Staci kramer. gigaom - the biggest thing amazon got right: The platform. <https://gigaom.com/2011/10/12/419-thebiggest-thing-amazon-got-right-the-platform/>, 2011.
- [2] Tony mauro. nginx - adopting microservices at netflix: Lessons for architectural design. <http://nginx.com/blog/microservices-at-netflix-architecturalbestpractices/>, 2015.
- [3] Steven ihde. infoq - from a monolith to microservices + rest: the evolution of linkedin's service architecture. <http://www.infoq.com/presentations/linkedin-microservices-urn>, 2015
- [4] Armin Balalaie, Abbas Heydarnoori, and Pooyan Jamshidi. Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture. IEEE Computer Society Press, 2016.
- [5] M Fowler, j lewis. monolith first. <http://martinfowler.com/bliki/MonolithFirst.html>, 2015
- [6] Mario Villamizar, Oscar Garcés, Harold Castro, Mauricio Verano, Lorena Salamanca, Rubby Casallas, and Santiago Gil. Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud. In Computing Colombian Conference, 2015.
- [7] M fowler, j lewis. microservice a definition of this new architectural term. <https://martinfowler.com/articles/microservices.html>, 2015.
- [8] Thomas F. Dillmann and Andr Van Hoorn. Model-driven generation of microservice architectures for benchmarking performance and resilience engineering approaches. In The Acmspec, pages 171–172, 2017.
- [9] Sara Hassan, Nour Ali, and Rami Bahsoon. Microservice ambients: An architectural meta-modelling approach for microservice granularity, 04 2017.
- [10] Tasneem Salah, M. Jamal Zemerly, Yeob Yeun Chan, Mahmoud AIQutayri, and Yousof Al-Hammadi. The evolution of distributed systems towards microservices architecture. In Internet Technology and Secured Transactions, pages 318–325, 2017.
- [11] Schmerl B, Kazman R, Ali N, et al. Managing Trade-Offs in Adaptable Software Architectures. 2017.
- [12] Danny Weyns. Software engineering of self-adaptive systems: an organised tour and future challenges. 2017.
- [13] Krupitzer C, Roth F M, Vansyckel S, et al. A survey on engineering approaches for self-adaptive systems. Pervasive & Mobile Computing, 17(PB):184-206, 2015.
- [14] Cheng Huang, David Garlan, and Bradley Schmerl. Rainbow: Architecture-based self-adaptation with reusable infrastructure. Computer, 37(10):46–54, 2004.
- [15] J. Malenfant, M. Jacques, and F. N Demers. A tutorial on behavioral reflection and its implementation. 1996.
- [16] IBM corp. an architectural blueprint for autonomic computing, tech. rep. <http://www03.ibm.com/autonomic/pdfs/AC>, 2005
- [17] Cherif S, Djemaa R B, Amous I. ReMoSSA: Reference Model for Specification of Self-adaptive Service-Oriented-Architecture. ADBIS. p121-128, 2014.
- [18] Danny Weyns and Radu Calinescu. Tele assistance: A self-adaptive service-based system exemplar. 05 2015.
- [19] Giovanni Toffetti, Sandro Brunner, Florian Dudouet, and Andrew Edmonds. Anarchitecture for self-managing microservices. InInternational Workshop on Automated Incident Management in Cloud, pages 19–24, 2015.
- [20] Krasimir Baylov and Aleksandar Dimov. Reference architecture for self-adaptive microservice systems. pages 297–303, 2017.
- [21] Dmitry Namiot and Manfred sneps sneppe. On micro-services architecture. 2:24–27, 09 2014.
- [22] Jeffrey O. Kephart and David M. Chess. The vision of autonomic computing. Computer, 36(1):41–50, January 2003



# A Heterogeneous Architecture for Integrating Multi-Agent Systems in AmI Systems

Vinicius Souza de Jesus and  
Fabian Cesar Pereira Brando Manoel  
CEFET/RJ  
e-mail: souza.vdj,fabiancpbm@gmail.com

Carlos Eduardo Pantoja and José Viterbo  
Universidade Federal Fluminense  
e-mail: pantoja@cefet-rj.br,viterbo@ic.uff.br

**Abstract**—Several challenges arise when applying Multi-Agent System (MAS) in Ambient Intelligence scenarios such as the heterogeneity of the hardware and the domain where it is applied. There are several applications that use Agent-Oriented approaches but they provide solutions that tie the hardware to the software, and they do not provide generic architectures. So, in this paper, we propose a heterogeneous architecture for applying different microcontrollers in the design of embedded MAS for such kind of systems. An architecture and a small-scale prototype of a smart home assembled with several hardware devices connected to different ATMEGA and PIC microcontrollers are presented as proof-of-concept. Our architecture shows to be effective in several tests performed using different implementation strategies.

## I. INTRODUCTION

Ambient Intelligence (AmI) comprises electronic and intelligent environments characterized by the interconnection of different technologies with the purpose of helping users in their daily tasks in an autonomous, proactive and pervasively way [1]. Multi-Agent Systems (MAS) are composed of autonomous agents situated in an environment and have the capacity of making decisions based on perceived stimuli and interactions between others agents to realize common or conflicting goals [2]. The agent-oriented paradigm is appropriate for implementing AmI systems because agents can be proactive, have social abilities and autonomy, and it is capable of learning from its past experiences [3].

In a common architectural approach to implement AmI systems and MAS, many sensors and actuators are managed by different microcontrollers, while in a more external layer, generic services for accessing sensors and actuators are implemented and provided for cognitive applications running on a top layer. Thus, is important to have an architecture able of controlling microcontrollers supported by an Agent-Oriented Program Language (AOPL) responsible for reasoning. Particularly, Jason [4] is a framework to develop MAS widely used in the agent community for programming cognitive systems interfacing hardware of the same type [5].

An extension of Jason named ARGO aims to facilitate the use of hardware devices by intelligent agents independently of the domain of the solution [6]. It means that it is possible to develop MAS where the software layer is not tied to the hardware layer and it can be used in any domain. Since the

hardware choice in the design of the solution is limited since it is only possible to use ATMEGA microcontrollers, ARGO employs a generic communication interface between the AOPL and the microcontroller [7], and nothing prevents the development of a communication interface for other microcontrollers.

Therefore, the objective of this paper is to present a layered architecture for designing MAS capable of adopting different microcontrollers where all layers are independent from each other. For this, we adopt Javino as communication interface between microcontrollers and the software layer, and we propose Javic for interfacing the software layer with PIC or other C based microcontroller. The PIC was chosen since it is often used in industrial applications because of its reliability. Besides, the Jason framework and ARGO agents were used as the cognitive reasoning in the software layer.

Then, we present as proof-of-concept a small-scale prototype of a smart home for temperature control and a doorbell system for helping the hearing impaired to identify if there is someone in front of the door. In order to evaluate the MAS and the prototype, some performance tests were executed taking into account parameters such as the number of agents and controllers, the agent's reasoning and the amount of environmental perception, to explore different implementation strategies of AmI System development supported by MAS.

Our contributions are: (i) the use of different kind of microcontrollers in the same MAS ; and (ii) a communication interface for working along with ARGO to program MAS for controlling PIC microcontrollers. This paper is structured as follows. In Section 2, we present some related work; In Section 3, the architecture is presented; In Section 4, the Smart Home architecture and its prototype are discussed; and Conclusion and future works are presented in Section 5.

## II. RELATED WORK

Some works exploit existent architectures and middleware to facilitate the connection between hardware and software. Some works try to embed the MAS into hardware platforms to provide real autonomy and other use a central processing unit for controlling the hardware from a distance.

In [8], it is presented an unmanned ground vehicle controlled by a MAS hosted in a computer and programmed in Jason. The agents can control the vehicle using commands that are sent to the hardware using radio transmitters on both sides

(computer and microcontroller), but the MAS only communicates with a single type of microcontroller. In addition, in all cited works, the MAS is not embedded with the hardware.

In [7], a platform for embedding MAS programmed in Jason using a Raspberry Pi board is presented. The MAS controls the functions of a vehicle using external actions. The agents are not able of controlling devices directly, becoming dependent on the simulated environment programmed in Java. Besides, it only communicates with a single type of microcontroller.

Applying MAS in AmI is not a new topic and several proposals integrating devices have already been presented in simulated smart homes such as [9]. However, it does not use AOPL, the agents do not use a cognitive model, and the implementation is tied to the solution. In [10], it is proposed a model for supporting residential accidents using embedded agents and Arduino. For each Arduino, there is a specific agent and a high-level agent replicated in case the former is not capable of processing. In this work, an ARGO agent controls many devices. An embedded approach in real time using Jade for programming MAS is proposed by [11]. The architecture maps each sensor and actuator in the high-level language, and there are six types of generic agents. In this paper, the sensors and actuators are connected to controllers and only ARGO agents can manage such devices.

### III. THE ARCHITECTURE

In this section, we propose a layered architecture for the development of MAS using an AOPL (responsible for the MAS) and a separated and heterogeneous hardware layer, with several microcontrollers, actuators and sensors (a heterogeneous architecture is able of employing different types of microcontrollers in the same solution). To establish communication between both independent layers, a third layer is used as middleware that is responsible for the exchanging data, through serial communication, between the software and the hardware. Using this middleware, the hardware is programmed to send all the perception and execute requests arriving from the MAS. Already in the software layer, agents are designed to interact with the infrastructure available sending actions to the hardware layer and receiving perceptions coming from sensor. In our architecture, the MAS is independent from the hardware and can be modified or changed if it is desirable.

So, the proposed architecture could employ microcontrollers of different types, each of them controlling sensors and actuators in the hardware layer, and a central core responsible for the deliberation based on information perceived by sensors in the software layer. It is used the Jason and ARGO agents for controlling hardware devices. The use of Javino and Javic middleware along with Jason and ARGO agents provides a platform, which supports the developer to deploy AmI systems without concerns about integration issues between hardware and software because of the independence between layers. All perceptions are directly processed by the MAS without intervention of the designer. Besides, the capability of using different microcontrollers in the same project controlled by a MAS is the main contribution of using the architecture.

#### A. The Hardware Layer Implementation

In the hardware layer of the architecture, it can be employed different types of microcontrollers. For this, libraries for capturing data from sensors and sending them to the software layer must be adopted. In this section, we discuss Javino and present the Javic library for interfacing PIC microcontrollers.

1) *The Javino [7]*: Javino is a communication interface used to exchange messages between microcontrollers and programming languages using serial communication. Its main benefit is to ensure that the receiver will not accept messages with errors. The Javino consists of two libraries: one on the hardware-side (microcontroller), and another one on the software-side (programming language).

When using Javino, the message follows a structure with three fields: **Preamble**, composed by 2 bytes of a fixed value to identify the message; **Size** that has 1 byte used to inform the size of the message sent and; the **Message** (up to 256 characters) to be sent. The two firsts fields are both used to identify errors that can occur because of information loss during the message transmission. In this work, Javino is used on the hardware layer for interconnecting ATMEGA microcontrollers or any other platform that uses the Arduino IDE such as Galileo, Galileo Gen 2, and NodeMCU. In this case, Javino is responsible for gathering perceptions from all sensors connected to the microcontroller, and it sends them for the software layer. The designer of the system must be responsible for programming the sensors and actuators stimuli based on messages received from the communication layer.

2) *The Javic*: Javic implements the same protocol as Javino. The Javic is a C-based library for PIC microcontrollers. Depending on the type of the PIC, the amount of available memory can interfere in the functioning of the library, because it was developed to work in PIC with at least 256 bytes of RAM because the size of the message is up to 256 bytes. The methods implemented for Javic and Javino are:

- **sendMsg(String msg)**: Sends a message to the software layer using serial communication;
- **availableMsg()**: Checks if exists messages coming from the software layer, returning a boolean value informing whether there is a message available or not;
- **getMsg()**: If there is a message available, it is used to get the request information sent by the software to perform some action using actuators or to gather perceptions.

When the software side library needs to send a message to the other side, it is necessary to inform the port where the target device is connected. Including Javino and Javic, there are 3 libraries: one for the software side that uses the Java language, and two for the hardware side, one for ATMEGA, and another one for PIC microcontrollers.

#### B. The Software Layer Implementation

In this section, we discuss the technologies used in this work for the development of the software side of the architecture. Jason [4] is a framework used to build cognitive MAS, and when used together with the customized architecture of agents

named ARGO [5], it is possible to develop solutions using actuators and sensors that can interact with the real world.

Jason is a framework that has an interpreter in Java of AgentSpeak for the development of cognitive agents using the BDI. The BDI contains three basic constructions: “Beliefs” (information considered to be truth by the agent acquired internally, with other agents or with the environment), “desires” (agent’s motivation to perform determined goal), and “intentions” (actions that the agent is compromised to execute) [4].

ARGO is a customized architecture of Jason agents for enabling the programming of agents capable of interacting with platforms of prototyping. The ARGO allows the intermediation between the cognitive agents and a real environment (using microcontrollers) through the Javino. A MAS can be composed of traditional Jason agents and ARGO agents working simultaneously. The Jason agents can perform plans and actions only in software level and communicate with other agents in the system (including ARGO agents). An ARGO agent is a traditional agent with additional features, such as the ability to communicate with the physical environment, perceive it, act upon it, and filter information perceived from sensors connected to microcontrollers. For this, ARGO has five internal actions to be used at runtime: (i) the action `.port(Port)`, where the agent chooses which device to control selecting the serial port where the device is connected (e.g. `.port(com8)`); (ii) the action `.percepts(open or block)`, where it is defined if the agent blocks or releases the flow of perceptions from the controller; (iii) the action `.limit(millisecons)`, which defines for how long the environment should be perceived; (iv) the action `.act(message)`, which sends a message through the serial port to execute an action using an actuator and; (v) the action `.filter(XML)`, which selects the XML file responsible for filtering perceptions.

#### IV. THE SMART HOME PROTOTYPE

In this section, we present a Smart Home architecture based on the proposed architecture using the Jason and the ARGO and containing several controllers with sensors (temperature) and actuators (LED lights). In Figure 1, we propose one possible architecture for a Smart Home prototype developed in wood, which has six rooms each one controlled by a microcontroller (three ATMEGA328 and three PIC). Four rooms have light sensors (LDR) and LEDs; one room has a temperature sensor (LM35), an air-conditioner(Peltier), a heater(Peltier) and LEDs and; the last room has the bell door button, the bell’s sound emitter (buzzer), the door motor and a LED. For each room, an ARGO agent is responsible for the cognitive management of sensors and actuators. The ARGO agents have the same communication skills as a traditional Jason agent, where an ARGO agent can communicate with another ARGO agent or with a traditional agent.

The methodology employed in the development of the smart home takes into account three layers where interventions are required: the interconnection of hardware devices, where sensors and actuators should be connected to the controllers in

the desired room; the microcontroller programming, where all the activation functions of the actuators must be programmed in the controllers in response to serial port stimuli, and; the MAS’s creation, where the perceptions coming from the sensors must be prepared to take into consideration the format expected by the MAS. The perceptions are sent to the agent every time an ARGO agent performs its reasoning cycle. Finally, the MAS must be independently programmed to the hardware, taking into account only the actions that must be performed in hardware. If it is necessary to change the device, there is no need to recode the MAS, but the agent should point to the proper serial port it desires to control, and the actions messages to be executed must be available on the new device.

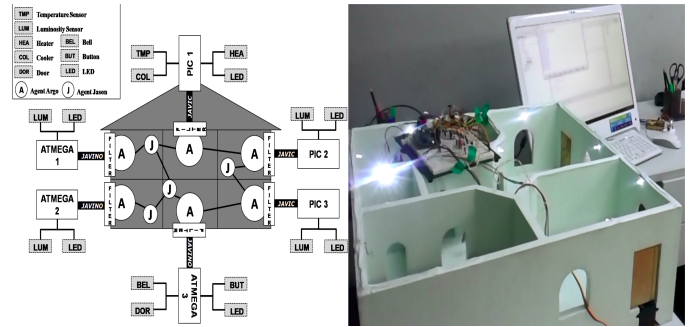


Fig. 1. The Smart home architecture and prototype.

#### A. Performance Tests

In order to test the applicability and to identify strategies for using ARGO agents in the AmI domain, we executed performance tests by performing a circuit of activating and deactivating of LEDs (the agent turns on all the LEDs from room to room and then turn them off). In the Jason, an event is generated for each perception received from the environment. This feature can lead to delays in the reasoning and execution of actions in situations that require fast responses. To deal with this situation, an agent can vary the use of the internal action that blocks and release the perceptual flow from sensors. Thus, tests were performed and the execution time of the activation circuit was measured employing from 1 to 6 microcontrollers being controlled by 1 to 6 agents in the MAS (36 possibilities) and using three different strategies:

- 1) **opened**: the agents open the flow of perceptions at the beginning of the execution to continuously perceive the environment until the end of the MAS execution without blocks its perceptions.
- 2) **just once**: the agent opens the flow of perceptions according to the need to execute each plain to search sensorial information and after that, it closes it again, acquiring just a few perceptions at the moment.
- 3) **lazy**: the agent open and close the flow of perceptions at the beginning of the first plan, repeat the operation at the last plan, and eventually when is necessary to modify controllers in the middle of the execution.

For each strategy, the tests were replicated 3 times totalizing 324 tests, of which 135 resulted in conflict because of the amount of ARGO agents was greater than microcontrollers employed (two or more agents can not use the same serial port at the same time). When there are conflicts by serial port competition, one solution is to implement a negotiation strategy to avoid that they do not use a device at the same time. For the remaining tests, when considering only the number of controllers managed by the MAS, it is noticed that **opened** strategy is the slowest of all because the number of perceptions processed is larger than the other strategies (this strategy never blocks its perceptions). The **lazy** strategy is slightly faster than **just once** strategy, but depending on the agent's programming or the domain, keeping the perception out of date for a period could cause some mistaken decisions.

Analyzing the results and the number of agents, we noticed that when one agent is responsible for controlling all devices, the execution time increases. The **opened** strategy is again the slowest due to the number of perceptions captured from sensors (they are updated in every reasoning cycle) while the **just once** and **lazy** strategy were faster because of the way perceptions were blocked. It is noteworthy that for all tests, the agent's codes were the same, differentiating only where the flow of perceptions was opened or closed. Figure 2 shows the scatter plot of the tests. For the next examples we decided that **just once** strategy should be used, it is slightly slower than **lazy** strategy, but it guarantees that the agents will have up-to-date perceptions of the sensors when compared with the former one. The **opened** strategy was not chosen because it was slower than the others. However, its characteristics always keep up-to-date information from sensors.

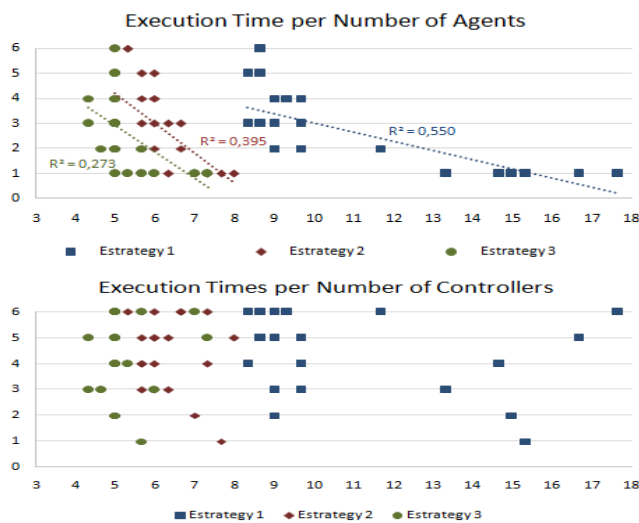


Fig. 2. The scatter plot of the number of agents employed (left) and the scatter plot of the number of controllers employed (right).

## V. CONCLUSIONS

This work presented a layered architecture for programming MAS which uses hardware devices, it uses Jason framework

and ARGO in the software layer, and it also uses libraries for communication to different microcontrollers. Then, we developed a specific library for integrating PIC microcontrollers with Jason framework allowing the design and construction of MAS and prototypes using both PIC and ATMEGA. This heterogeneous characteristic is an important issue for developing AmI systems because it is possible to use devices accordingly to the requirements of the designed MAS. Most of the platforms in the literature are directed to one kind of technology, or it ties to a particular domain. Our proposed architecture does not tie the design of the system to a particular domain, and it allows the use of several types of microcontrollers.

In AmI, we aim to use this heterogeneous characteristic of the proposed architecture to develop smart and proactive environments. So, we presented as proof-of-concept, a smart home example showing that is possible to use Jason to develop such kind of solutions. Besides, an analysis of three strategies for MAS implementation were presented to identify the best strategy to obtain quicker and efficient responses from agents. The results show that the delays generated are acceptable depending on the domain and strategy applied. As future work, it is necessary to test the approach in a real environment using complex scenarios with a high number of sensors.

## REFERENCES

- [1] W. Weber, J. Rabaey, and E. Aarts, *Ambient Intelligence*. Springer, 2005.
- [2] M. Wooldridge, *An Introduction to MultiAgent Systems*. Wiley, 2009.
- [3] C. Maciel, P. C. de Souza, J. Viterbo, F. F. Mendes, and A. El Fallah Seghrouchni, *A Multi-agent Architecture to Support Ubiquitous Applications in Smart Environments*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 106–116.
- [4] R. H. Bordini, J. F. Hübner, and M. Wooldridge, *Programming Multi-Agent Systems in AgentSpeak using Jason*. John Wiley & Sons Ltd, 2007.
- [5] C. E. Pantoja, M. F. Stabile, N. M. Lazarin, and J. S. Sichman, "Argo: An extended jason architecture that facilitates embedded robotic agents programming," in *Engineering Multi-Agent Systems: 4th International Workshop, EMAS 2016*, M. Baldoni, J. P. Müller, I. Nunes, and R. Zalila-Wenkstern, Eds. Springer, 2016, pp. 136–155.
- [6] C. E. Pantoja and J. Viterbo, "Prototyping ubiquitous multi-agent systems: A generic domain approach with jason," in *Advances in Practical Applications of Cyber-Physical Multi-Agent Systems: The PAAMS Collection: 15th International Conference, PAAMS 2017, Porto, Portugal, June 21-23, 2017, Proceedings*, Y. Demazeau, P. Davidsson, J. Bajo, and Z. Vale, Eds. Springer International Publishing, 2017, pp. 342–345.
- [7] N. M. Lazarin and C. E. Pantoja, "A robotic-agent platform for embedding software agents using raspberry pi and arduino boards," in *9th Software Agents, Environments and Applications School*, 2015.
- [8] R. S. Barros, V. H. Heringer, N. M. Lazarin, C. E. Pantoja, and L. M. Moraes, "An agent-oriented ground vehicle's automation using Jason framework," in *6th International Conference on Agents and Artificial Intelligence*, 2014, pp. 261–266.
- [9] K.-I. Benta, A. Hoszu, L. Văcariu, and O. Creț, "Agent based smart house platform with affective control," in *Proceedings of the 2009 Euro American Conference on Telematics and Information Systems: New Opportunities to increase Digital Citizenship*. ACM, 2009, p. 18.
- [10] G. Villarrubia, J. F. De Paz, J. Bajo, and J. M. Corchado, "Ambient agents: embedded agents for remote control and monitoring using the pangea platform," *Sensors*, vol. 14, no. 8, pp. 13 955–13 979, 2014.
- [11] E. Kazanavicius, V. Kazanavicius, and L. Ostaseviciute, "Agent-based framework for embedded systems development in smart environments," in *Proceedings of International Conference on Information Technologies (IT 2009)*, Kaunas, 2009.

# BackPocketDriver – A Mobile App to Enhance Safe Driving for Youth

Catherine Shanly, Michael Ieti

Department of Electrical and Computer Engineering  
The University of Auckland, New Zealand  
{csha289, miet001}@aucklanduni.ac.nz

Ian Warren and Jing Sun

Department of Computer Science  
The University of Auckland, New Zealand  
{i-warren, j.sun}@cs.auckland.ac.nz

**Abstract**—Young drivers are one of the highest risk groups for being involved in car accidents. BackPocketDriver (BPD) is an Android application that aims at encouraging young drivers to adopt and hone safe driving skills. Smartphone sensors are used to monitor driver behaviour, including speed, turning, acceleration and braking. The journey data is analysed for unsafe behaviour with user feedback, which includes journey review, positive reinforcement using textual messages, goal setting and points scoring. To achieve this, behavioural change techniques were studied in relation to gamification, and several features were implemented into BackPocketDriver including achievements, leader board, quizzes, friends system, etc. Evaluations in terms of functional comparisons with related tools were conducted to measure the advantages of the proposed solution. The BPD app provides effective improvements to youth driving.

## I. INTRODUCTION

Youth drivers are one of the most at-risk groups in New Zealand with regards to car accidents. From 2013-2015, drivers from 15-24 accounted for only 13% of all licensed drivers, but were involved in 24% of all minor crashes, 23% of all major crashes, and 19% of all fatal crashes [1]. These crashes resulted in more than 200 deaths in those two years.

Further investigation into the causes of these crashes resulted in a number of primary factors being identified. The main factors were alcohol/drug consumption, losing control, and speed. More than half (53%) of all fatal crashes had alcohol, drug use, or speeding identified as being contributing factors. Young drivers are more than twice as likely to have speed or alcohol as a crash factor compared to drivers over the age of 25 [1]. In a report by Transport Engineering Research New Zealand (TERNZ), it is stated that most fatal crashes occur at speeds over 60 km/hr [2]. These statistics highlight points where action can be taken to reduce deaths - reducing speed and reinforcing safe driving behaviours and habits may be the key to minimising driving-related injuries and deaths in New Zealand.

Avoidable crashes also create a large amount of economic, financial, and intangible cost for society. These costs are largely attributed to reduced quality of life for victims, fatalities, loss of productivity, as well as medical and legal fees, totaling \$3.79 billion in the year 2015. Vehicle damage only accounts for around 5% of this cost [3].

DOI reference number: 10.18293/SEKE2018-011

The ubiquity of smartphones in all demographics opens a channel for engagement with drivers of all ages. Smartphones have reached 70% market penetration in New Zealand, with 91% of 18-34 year olds owning at least one smartphone [4-7]. In order to focus efforts towards younger drivers, the use of a smartphone app can be seen as an appropriate measure to analyse and correct unsafe driving behaviours [8].

BackPocketDriver (BPD) is an Android application that aims at encouraging young drivers to adopt and hone safe driving skills. Smartphone sensors are used to track a user's driving, such as speed, turning, acceleration, braking, etc. Behavioural Change Techniques (BCT) [3,4] and gamification ideas [9, 10] were adopted in the development of the tool. Users can set their own goals to improve the smoothness of their driving and to reduce their speeding. The data is then analysed for unsafe behaviour through user feedback. A journey summary is provided after a trip, detailing the route taken, the user's current goals, and if they have met them. In this summary, specific feedback is also given to the user. An additional feature of BackPocketDriver is a daily messaging system gives the user safe driving tips or instructions daily.

The rest of the paper is organised as follows. In Section II, we present the development of the BackPocketDriver app. Section III presents the evaluation of the tool, including the feature comparison to existing safe driving applications. Finally, Section VI concludes the paper and outline the future work.

## II. DESIGN AND IMPLEMENTATION

### A. Overall Technologies

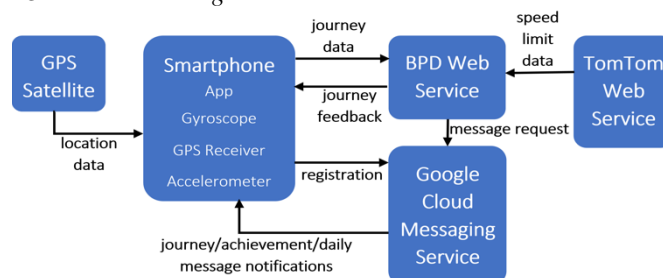


Fig. 1. BackPocketDriver component diagram

Fig. 1 shows the components of BackPocketDriver and its overall architecture. The user's smartphone uses the GPS, accelerometer, and gyroscope to calculate journey data. This is

then sent to the BackPocketDriver web service for processing before journey feedback is sent back to the device. A messaging service is also used for registration purposes and to send daily messages and notifications to the user.

We integrated several technologies for the development of the app and server for BackPocketDriver. The app is an Android smartphone app and the server code was implemented in Java programming language. The MySQL relational database management system was used for the server while SQLite was used for the app. Additionally, the GreenDAO generator was used in the app to map Java objects to the relational database. We used Apache Tomcat, an open-source servlet container that provides an HTTP web server environment. We used Google Cloud Platform to host the application.

### B. Key Features

Fig. 2 shows the *Home* screen that the user first sees when they open BackPocketDriver. Users can start and stop recording their journey on this screen by pressing the car icon. The username of the user is displayed which allows the user to identify themselves and more easily compare themselves with others.

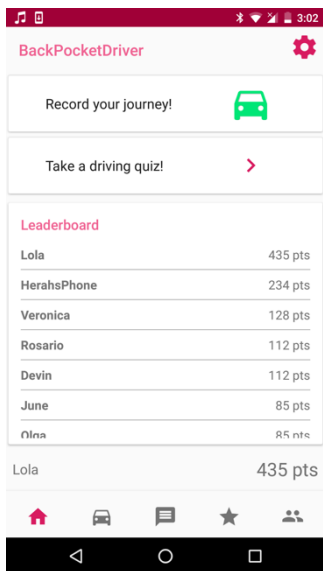


Fig. 3. Home screen implementation

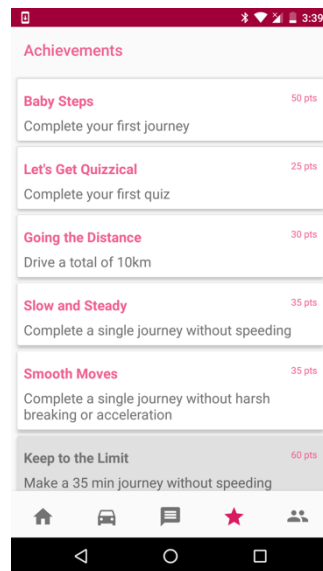


Fig. 3. Achievements implementation

The user can click on the gear icon to access user preferences. This gives the user control over their use of the app, for example, in our implementation users can choose to appear on the global leaderboard. The use of preferences can promote feelings of autonomy, which increases motivation [15], but is also important because BackPocketDriver contains potentially sensitive information about users.

On the *Home* screen the user can also see how many points they have earned. This is immediately available for the user to see and links with the BCT [12] of feedback on behaviour as users will see an increase in their score as soon as they perform a good behaviour, such as recording a journey without harsh braking.

The leaderboard has also been implemented on this screen. The leaderboard shows the top-ten users and ranks them against the number of points they have earned (users can earn points by recording journeys, completing quizzes, and obtaining achievements). Users are motivated to perform point earning activities when they compare their score against others and are motivated to *continue* performing them if they see themselves as a role model on the top of the leaderboard [11].

Fig. 3 shows the *Achievements* screen where users can obtain achievements by carrying out certain tasks, for example, the “Baby Steps” achievement requires that the user complete a journey for the first time. Achievements are motivating as users can set goals for themselves to obtain a specific achievement and they also visually show (achievement items are no longer greyed out) a sense of progress for the user as they obtain more achievements [14].

We have implemented achievements so that users can earn points when they complete a specific task. Each achievement has a number of points associated with it making the use of points more meaningful as users know exactly that they need to do, for example driving a total of 10km, to earn points.

There are two different types of achievements implemented in BackPocketDriver. The “Let’s Get Quizzical” achievement, for example, directs users towards the quiz functionality in the app. This can help direct users to functionality that they may not be aware of. Alternatively, the “Smooth Moves” achievement motivates the users to display good driving behaviours (completing a journey without harsh braking or acceleration) [13]. This links with the BCT of instruction on how to perform a behaviour as it explicitly shows the user what is considered good driving behaviours [16].

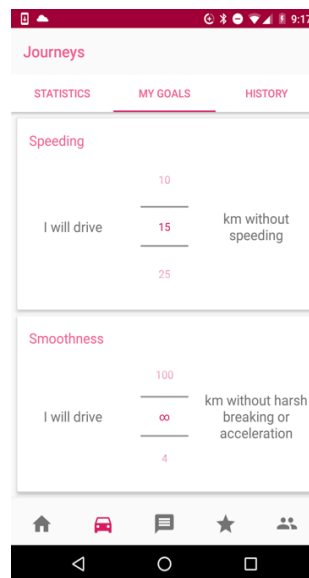


Fig. 4. My Goals screen

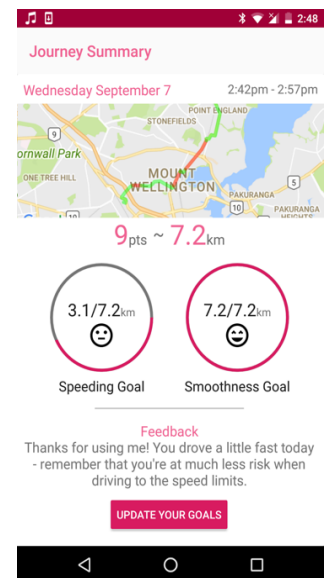


Fig. 5. Journey Summary screen

Fig. 4 and 5 show the screens for the redefining trip goals, reviewing trip goals after a journey, and redesigning journey feedback requirements. Fig. 4, the *My Goals* screen, allows the user to set themselves a speeding goal and a smoothness goal for each journey. In our implementation, the goals are clearly

TABLE I  
SUMMARY OF FEATURES

BCTs	Feature					
	Points	Leaderboard	Achievements	Quizzes	Friends	(Redefined) Trip Goals
Feedback on behaviour	X		X			X
Social comparison		X	X		X	
Goal-setting		X	X			X
Identification of self as role-model		X				
Instruction on how to perform a behaviour			X	X		
Discrepancy between current behaviour and goal						X
Review behaviour and goals						X

defined, for example, on the *My Goals* screen the speeding goal is “I will drive 15km without speeding.” The user can scroll through a list of available numbers to change the goal depending on their ability. The “∞” indicates that the user, for example, will not brake or accelerate harshly for the whole duration of their journey. If the length of the user’s trip is less than their current goal, then the user must also drive the whole trip perfectly.

Fig. 5 shows the *Journey Summary* screen which has been updated from the original app to reflect the changes in the goals. There is now a circular progress bar that shows the user progress towards their goal as well as the exact distance they have driven without speeding and the distance they have driven smoothly. These two features show the discrepancy between the user’s behaviour and their goal, another BCT.

On the *Journey Summary* screen, we have implemented a button that prompts the user to update their goals after they have seen their journey feedback and if they have met their goals. This leads the user back to the *My Goals* screen, allowing them to review their goals. Redefining the journey goals involved many changes across the app and server code. Journeys had to be processed differently so that the distance speeding and driving harshly was calculated, stored, and sent to the app as part of a journey summary object (previously only the proportion of the distance speeding was stored).

### C. Summary of Implemented Features

Table I shows a summary of the features implemented during the timeline of this project and the BCTs they are associated with. Additionally, there are other links between features. For example, users are ranked on the leaderboard based on how many points they have, which they can earn by recording journeys, achievements, and completing quizzes. Achievement activities correspond to several other functionalities of the app and support the behavioural change ideas, e.g., the achievement for appearing on the leaderboard is named “Role Model”. The friends system also allows users to view the achievements of their friends. These links all show how the implemented features are designed to support and reinforce each other.

## III. EVALUATION

### A. Performance Measurements

The performance of the server was briefly analysed using loader.io, a website that can stress test web applications with multiple concurrent connections [17]. Our tests retrieved the top

ten users on the leaderboard where each test had, over a 1 minute period, an increasing amount of client requests.

There were no errors up to 500 client requests over a minute. This test had an average response time of 70ms. 1000 clients had a 0.04% error rate and an average response time of 133ms. For 5000 clients, there was an 0.82% error rate with an average response time of 156ms. 10,000 clients over a 1 minute period had a 25.21% error rate with an average response time of 1224ms. All errors were timeouts (set at 10 seconds).

This shows that as the number of clients increase the longer the response time is and the more errors there are due to timeouts. Our web server has therefore limited scalability. This problem needs to be addressed before BackPocketDriver is made available to more users.

### B. Comparisons of BackPocketDriver to Other Safe Driving Applications

Table II shows a selection of safe driving applications, available on the Google Play Store, and their features (based on the app description). The most similar app to BackPocketDriver is the TOWER Insurance SmartDriver app [18], however, as this has been created by an insurance company users may not be motivated to use the app if they initially identify as a poor driver.

Some gamification features, such as points, appear to be popular across some of the other apps. Other features that BackPocketDriver does not include are a safe driving mode where the app discourages the user from handling their phone while driving. Instead, the app may perform tasks on behalf of the user using voice commands.

The data demonstrates that there is a gap in safe driving apps that BackPocketDriver has the potential to fill. The proposed solution uses a range of gamification features and uses a social network (friends system) which is not otherwise common. In the future, BackPocketDriver could support some of the other features present in the other apps to appeal to a wider audience.

## IV. CONCLUSIONS

The relationship between BCTs and gamification was examined and a clear link was found between these two concepts. The results of this study were applied to BackPocketDriver, a smartphone application created for encouraging safe driving in youth drivers. Limitations of the existing app were addressed by extending existing features and by implementing the following game elements: achievements, leaderboards, and quizzes. Additionally, a friends system was

TABLE II  
COMPARING BACKPOCKETDRIVER TO OTHER SAFE DRIVING APPS

Feature	Application					
	BackPocketDriver	TOWER Insurance SmartDriver [18]	Dash – Drive Smart [19]	Drive Safe [20]	Safe Driving App [21]	EROAD Driver [22]
Journey feedback	X	X	X		X	
Goals	X					
Statistics	X	X		X	X	
Daily messages	X					
Points/Score	X	X	X		X	
Leaderboard	X	X	X			
Quizzes	X					
Achievements	X	X				
Rewards		X				
Social network	X	X				
Logging driving						X
Vehicle inspection details						X
Driving tips						X
Safe driving mode			X	X		

implemented. These features were chosen based on their motivational effectiveness and their relevance to BCTs. They were evaluated by conducting a performance and comparative study to related tools. The results found that proposed solution provides effective means to improve youth driving.

In the future, the validity of the app needs to be evaluated by carrying out a long-term study of users recording their driving with the app. During this study the motivational effectiveness of features could be evaluated by recording how often the user visits different functionality as well as seeing if their driving performance improves over time.

#### REFERENCES

- [1] Young Drivers | NZ Transport Agency. Available: <http://www.nzta.govt.nz/safety/driving-safely/young-drivers/>.
- [2] Young drivers crash facts | Ministry of Transport. Available: <http://www.transport.govt.nz/research/crashfacts/youngdriverscrashfacts>
- [3] S. Michie et al, "The behavior change technique taxonomy (v1) of 93 hierarchically clustered techniques: building an international consensus for the reporting of behavior change interventions," *Annals of Behavioral Medicine*, vol. 46, (1), pp. 81-95, 2013.
- [4] C. Abraham and S. Michie, "A taxonomy of behavior change techniques used in interventions." *Health Psychology*, vol. 27, (3), pp. 379, 2008.
- [5] A. Direito et al, "Do physical activity and dietary smartphone applications incorporate evidence-based behaviour change techniques?" *BMC Public Health*, vol. 14, (1), pp. 646, 2014.
- [6] C. C. Quinn et al, "Cluster-randomized trial of a mobile phone personalized behavioral intervention for blood glucose control. *Diabetes Care* 2011; 34: 1934-1942," *Diabetes Care*, vol. 36, (11), pp. 3850, 2013.
- [7] F. Fylan and S. Stradling, "Behavioural Change Techniques used in road safety interventions for young people," *Revue Europeenne De Psychologie Applique/European Review of Applied Psychology*, vol. 64, (3), pp. 123-129, 2014.
- [8] S. Deterding et al, "From game design elements to gamefulness: Defining gamification," in *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, 2011, pp. 9-15.
- [9] M. Sailer et al, "How gamification motivates: An experimental study of the effects of specific game design elements on psychological need satisfaction," *Comput. Hum. Behav.*, vol. 69, pp. 371-380, 2017.
- [10] J. Hamari, J. Koivisto and H. Sarsa, "Does gamification work?--a literature review of empirical studies on gamification," in *System Sciences (HICSS)*, 2014 47th Hawaii International Conference On, 2014, pp. 3025-3034.
- [11] R. N. Landers, K. N. Bauer and R. C. Callan, "Gamification of task performance with leaderboards: A goal setting experiment," *Comput. Hum. Behav.*, 2015.
- [12] J. Hamari, "Do badges increase user activity? A field experiment on the effects of gamification," *Comput. Hum. Behav.*, 2015.
- [13] C. Cheong, F. Cheong and J. Filippou, "Quick quiz: A gamified approach for enhancing learning." in *Pacis*, 2013, pp. 206.
- [14] E. A. Edwards et al, "Gamification for health promotion: systematic review of behaviour change techniques in smartphone apps," *BMJ Open*, vol. 6, (10), pp. e012447, 2016.
- [15] R. M. Ryan and E. L. Deci, "Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being." *Am. Psychol.*, vol. 55, (1), pp. 68, 2000.
- [16] L. de-Marcos, E. Garcia-Lopez and A. Garcia-Cabot, "On the effectiveness of game-like and social approaches in learning: Comparing educational gaming, gamification & social networking," *Comput. Educ.*, vol. 95, pp. 99-113, 2016.
- [17] Application Load Testing Tools for API Endpoints with loader.io. Available: <https://loader.io/>.
- [18] SmartDriver - Android Apps on Google Play. Available: <https://play.google.com/store/apps/details?id=com.tower.smartdriver>.
- [19] Dash - Drive Smart - Android Apps on Google Play. Available: <https://play.google.com/store/apps/details?id=com.dashlabs.dash.android>
- [20] Drive Safe - Android Apps on Google Play. Available: <https://play.google.com/store/apps/details?id=westport.andrewirwin.com.drivesafe>.
- [21] Safe Driving App - Android Apps on Google Play. Available: <https://play.google.com/store/apps/details?id=com.risktechnology.indirect>.
- [22] EROAD Driver - Android Apps on Google Play. Available: <https://play.google.com/store/apps/details?id=com.eroad.driver>.



# A Real-Time Ride-Sharing Matching Framework Using Simulated Annealing Genetic Algorithm

Jie Xu<sup>1</sup>, Yong Zhang<sup>1</sup>, Chunxiao Xing<sup>1</sup>, Guigang Zhang<sup>2</sup>

<sup>1</sup>Research Institute of Information Technology, Beijing National Research Center for Information Science and Technology, Department of Computer Science and Technology, Institute of Internet Industry, Tsinghua University, Beijing 100084, China

<sup>2</sup>Institute of Automation, Chinese Academy of Sciences

{xuj15}@mails.tsinghua.edu.cn, {zhangyong05, xingcx}@tsinghua.edu.cn, {guigang.zhang}@ia.ac.cn

**Abstract:** Recently, ride-sharing services are becoming more popular in commercial application, which have attracted many researchers' attention. The ride-sharing provides significant economic, societal and environmental benefits in a sharing economy, such as reducing pollution, travel costs and traffic congestions. This kind of problem essentially is to maximize the matched ride-sharing pairs, and obviously is an optimization problem. There are lots of platforms for dynamic peer to peer ride-sharing, however, existing researches can be improved better. For example, how to reduce the algorithm computing time, and maximize the matching effectiveness to gain more economic benefits. In this work, we first introduce the problem of ride-sharing with time guarantee on road network, and then design a novel heuristic simulated annealing genetic algorithm. In addition, we carefully adjust the parameters with different constraints, and conduct extensive verification experiments with realistic datasets derived from Beijing car services, the results demonstrate the advancement of our methodologies.

**Keywords—**Road Network, Ride-sharing, Simulate Annealing Algorithm, Genetic Algorithm

## I. INTRODUCTION

Recently, we witness the rapid growth of the sharing economy, in which the resource owners temporarily transfer their properties to someone to acquire the benefits, the ride-sharing is the one of the most representative applications, that means someone acts as a driver and offers the ride-sharing service to bring together travelers with similar trip. Nowadays, the ride-sharing has become very prevalent application, consequently appearing lots of startups companies, typical like DIDI, Uber, and Lyft. There are lots of increasing researches, however, most existing methods have several limitations, for example, the efficiency is not high, and the methods need deep analysis of specific issues of the problems, or the extendibility is weak. Generally speaking, solving this kind of optimal problem is not easy. People have studied various approaches to address it [1,2,3], such as the integer linear programming, the branch-and-bound problem and the approximation algorithm [4].

In this work, we focus on the ride-sharing problem with time windows. There are many challenges to address such problems: First, it has been proved as an NP hard problem [10]; Second, to calculate the sharing ratio, it needs to calculate the shortest path, it is time-consuming; Third, for the drivers, their main concerns are the economic benefits, thus it is not only needed to minimize detour distance at most in carry

services, but also to maximum the number of matched pairs, it is a bi-objective problem, which can be solved by the simulated annealing (abbr.SA) or genetic algorithm (abbr.GA). Simulated annealing is a probabilistic technique for approximating the global optimum in a large search space, and the genetic algorithm is derived from the process of natural selection and genetic evolutionary. With the above mentioned knowledge, we present a framework which combines the simulated annealing algorithm with the genetic algorithm. The contributions of this work are summarized as follows: First, we establish the map grid, it can effectively reduce the shortest path calculation time. Second, we designed a formulation to describe the problem, and then develop a novel simulated annealing genetic algorithm (abbr. SAGA). Third, we conduct extensive experiments on the real dataset, and our results show the superiority of our approach.

The rest of this paper is organized as follows: we propose the related important work on ride-sharing and kinds of approaches in Section 2, and then model the problem in Section 3, afterwards, we illustrate our algorithm in detail in Section 4. Experimental results are presented in Section 5. We outline the conclusions of the paper in Section 6.

## II. RELATED WORKS

The ride-sharing problem has attracted increasing attention. In this section, we introduce the related work and summarize the key technologies for different algorithms [5,6]. The studies can be classified into three categories, approximation algorithms, filter and refine algorithms, heuristic algorithms.

Ta et al. [7] proposed a new ride-sharing model that each driver has a requirement restriction with the shared route percentage, two variants of the ride-sharing problem are proposed, i.e., multiple drivers and a single rider and multiple drivers and multiple riders. To improve the efficiency, an approximate method with error bound guarantee was proposed, that meant updating the lower and upper bounds of the maximal graph matching until the bounds gap was small. The idea was very innovative and useful.

In [8], a highly generalized model for the taxi and delivery services in the market economy was proposed. The model can efficiently used with surge pricing mechanism. To solve the above problem, they proposed an approximation algorithm which was transferred to the multiple disjoint paths (MDP) problem, and carefully proved their algorithm had a tight approximate ratio with the original problem.

Furuhata et al. [9] proposed a review of ride-sharing to understand the key aspects of existing ride-sharing systems. The main characteristics to describe different aspects of ride-sharing systems were described. They classified the existing problem into six categories depending the target segments and criteria. The objective of the work was to identify key challenges thus helped to build the effective formal ride-sharing mechanisms.

Ma et al. [10] demonstrated a large-scale taxi ride-sharing for a dynamic ride-sharing problem, the framework included two steps: the first was a definition of a searching algorithm by taking advantage of a spatial-temporal index, then a scheduling algorithm was proposed. To address the heavy computational load, a lazy method was described by partitioning the road network into several grids. The approximated distance finally approved the effectiveness of their approach.

Thangaraj et al. [11] described a platform named Xhare-a-Ride (XAR) system for dynamic peer-to-peer ride-sharing. Three kinds of geographical hierarchical discretization including region using grids, landmarks and clusters, were used to help to eliminate shortest path computation during search. The notions were valuable for integration with multi modal trips.

Alarabi et al. [12] presented an effective framework named SHAREK which can embed inside existing approaches to improve the performance and quality. The model took into consideration the users' maximum willing waiting time and the cost of both riders and drivers. Three consecutive phases named Euclidian temporal pruning, Euclidian cost pruning, and Semi-Euclidean skyline-aware pruning were used to avoid the expensive shortest path computations.

Qian et al. [13] introduced SCRAM in order to provide recommendation fairness without sacrificing driving efficiency, the by-product of a framework named SCRAM was the capability of offering actual driving routes rather than rough driving directions. The recommended routes were evaluated from three aspects, i.e., priority principle, decaying principle, sharing principle. The successive probability of picking up customers had higher priority than the driving cost, and the probabilities of road sections were appraised decreasingly with the distance from the starting point.

Huang et al. [14] proposed a large scale service guarantee real-time ride-sharing, and demonstrated kinetic tree algorithms which can satisfy dynamic scheduling requests. They depicted a hotspot-based algorithm to avoid duplicates computation, the drop-off locations were grouped which led to a large number of valid schedules to satisfy the rider-driver constraints.

In [15,16], a genetic and insertion heuristic algorithm for a single rider was proposed. The main idea was to fine the best trips which can be shared by more than one driver with time constraints. They first modeled the ride-sharing problem with the time constraints, then illustrated the detailed implementation steps of the genetic algorithm. They also defined five different mutation operators basing on diverse

services time constraints. The experimentation results indicated their algorithm was feasible.

### III. PROBLEM DEFINITION

In this section, we formally propose the preliminaries and define the notions of our ride-sharing problem, and then depict the framework of our system.

#### A. Preliminaries

Definition 1.  $G = (V, E)$  is used to model a road network, where a vertex set  $V$  is associated with a geographical position, including the longitude and latitude. An edge set  $E$  is associated with the weight such as the distance or travel cost between two different vertices.

Definition 2. (Rider). A rider  $r_i = (r_i^s, r_i^e, t_r^e, t_r^L, t_{r-r}^L)$  is defined as a rider who intends to go to the destination  $r_i^e$  from  $r_i^s$  within an interval bounded by the earliest time  $t_r^e$  and the latest time  $t_r^L$ . (Driver). A driver  $d_i = (d_i^s, d_i^e, t_d^e, t_d^L, d_i^s, t_{d-r}^L, \lambda_{max})$  is defined as a driver who plans to go to the destination  $d_i^e$  from  $d_i^s$ , at the latest start time  $t_d^L$  with a acceptable detour distance ratio  $\lambda_{max}$ .  $D$  denotes the drivers set,  $R$  denotes the riders set.

Definition 3. (Sharing ratio for driver) considering the driver's minimum detour distance  $\lambda_{max}$ , we introduce the sharing ratio to measure the ride-sharing effectiveness.

$$\xi(d_i, r_j) = \frac{\delta(d_i, r_j)}{\delta(d_i, r_j) + \text{detour}(d_i, r_j)} \quad (1)$$

$\delta(d_i, r_j)$  stands for the shortest distance of rider  $r_j$ , the  $\text{detour}(d_i, r_j)$  refers to the cost of detour distance to pick up or put down the riders. We assume that if the sharing ratio is less than the ratio bound ( $\lambda_{max}$ ), the match is valid. Some key notations are summarized in Table 1 [7]:

Table 1. Some Key Notations

Notation	Definition
$r_i^s$	rider's start point
$r_i^e$	rider's destination point
$t_r^e$	rider's earliest start time
$t_r^L$	rider's latest start time
$t_{r-r}^L$	rider's latest reach time
$t_d^e$	driver's earliest start time
$t_d^L$	driver's latest start time
$d_i^s$	driver's start point
$d_i^e$	driver's destination point
$\lambda_{max}$	driver's maximum acceptable detour ratio
$\xi(d, r)$	detour distance for matched pair

#### B. Problem definition

The ride-sharing problem is defined as follows: For a set of drivers and a set of riders on a road network  $G(V, E)$ , when the riders send series requests, we aim to satisfy the requests

and acquire the maximum sharing matching ratio denoted by  $\sum_{\forall d_i \in D, r_j \in R} \text{detour}(d_i, r_j)$  under the temporal and distance factors [9]. In Figure 1, there are 3 drivers and 3 riders, we aim to match the drivers and riders with the minimum detour distance.

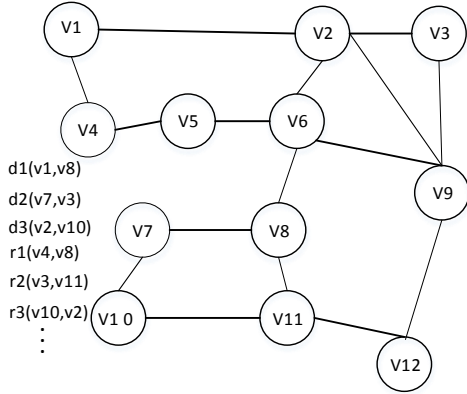


Fig. 1. Example of ride-sharing between drivers and riders

#### IV. A SIMULATED ANNEALING GENETIC ALGORITHM

Our goal is to minimize the detour distance for whole riders' requests, it is obvious a global optimization problem. A brute force method is to enumerate all possible driver-rider pairs. However, traditional approaches are rather time consuming and inefficiency. To tackle the challenges, we first divide the map into many regular squares to quickly find the approximate shortest distance, and then introduce a heuristic simulated annealing genetic algorithm (SAGA) to address the optimal driver-rider pairs matter.

##### A. Grid construction

To calculate the shortest path on road networks, Ma et al.[10] presented a spatial grid which did pre-calculation and stored the shortest path by dividing the map into small fixed square areas [18]. Their main ideas lay in the approximate path and distance. Similarly, we also utilize this basic ideas, select the history hottest visited places on behalf of the grid's location, and then construct the shortest distance matrix and the shortest path matrix. For any two arbitrary points, we first figure out which grids they geographically belong to, then we acquire the shortest distance by querying the corresponding grid matrix.

##### B. Formulation representation

The bi-objective function that minimizes the total distance and maximizes the number of the valid matched pairs is provided in Equation 2. The  $\alpha, \beta, \gamma$  define the relative weight factors. Let  $x_{i,j}=1$  if the driver-rider pair is matched from node  $i$  to node  $j$ . The formulation is,

$$\max(-\sum_{\forall d_i \in D, r_n \in R} \text{detour}(d_i, r_n) + \sum_{\forall x \in D, y \in R} x_{i,j}) \quad (2)$$

subject to:

$$x_{ij} \in \{0,1\} \quad (3)$$

$$\sum_{i \in D} d_i \sum_{j \in R} r_j = D + R \quad (4)$$

$$\sum x_{ij} \leq D_{\text{number}} \quad (5)$$

$$\sum x_{ij} \leq R_{\text{number}} \quad (6)$$

$$t_d^L \leq t_r^e \quad (7)$$

$$t_d^L \leq t_r^L \quad (8)$$

$$\xi(d_i, r_j) \leq \lambda_{\max} \quad (9)$$

where Constraint (3) enforces the matched pairs have only two choices: 0 or 1. Constraints (4) ensures that the number of drivers and riders leaving the starting is equal to the number of drivers and riders arriving at the destination. Constraints (5) and Constraints (6) ensures the total matched driver-rider are less than the total initial given drivers. Constraints (7) ensures driver's latest start time is earlier than rider's earliest start time. Constraints (8) ensures driver's latest start time is earlier than rider's latest start time. Constraints (9) ensures driver's detour distance is less than the driver's default acceptable detour distance.

##### C. Procedure of SAGA

The procedure of SAGA is described in Figure 2. The GASA process firstly starts by given some important parameters such as population size, initial temperature, number of generations, probability of crossover and mutation. An initial population of chromosome is also generated, every chromosome is a candidate solution for problem [17].

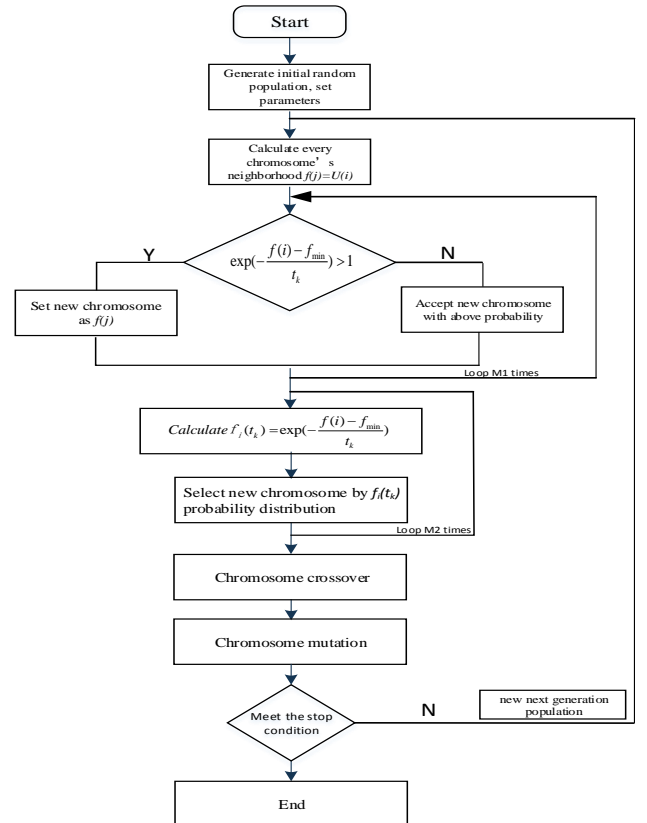


Fig. 2. SAGA procedure for ride-sharing

For every chromosome, we use the objective function (Equation 2) as a fitness function. The selection of fitness function directly affects the convergence speed of genetic algorithm. A random value  $U(i)$  as  $f(j)$  is selected from the neighborhood of  $f(i)$ , then probability function  $\exp(-\frac{f(j)-f(i)}{t_k})$  is calculated. If the value is greater than constant one, then accept the neighborhood  $f(j)$  as the new chromosome, else accept the new one with above probability. After above steps have been executed in  $M_1$  cycles, new population named NewPopOne where the size is equal to the initial population is generated. Later, the fitness of each chromosome represented by function  $\exp(-\frac{f(i)-f_{\min}}{t_k})$  is calculated, afterward, new chromosome with the probability distribution determined by above fitness function is generated, that process which is executed in  $M_2$  cycles finally results in the next new population named NewPopTwo. In addition, the crossover and mutation strategies are executed to create offspring until the next new population is generated.

The above mentioned steps are repeated until the results meet the expectation value or iteration operations are all completed.

#### D. Problem coding

We randomly assemble the driver-rider matched pairs successively in one chromosome, the length of the chromosome depends on the minimum of drivers or riders. The whole population is set as 50. Figure 3 presents the problem coding.

parent 1:

1	4	8	8	7	3	11	3	2	10	2	10
---	---	---	---	---	---	----	---	---	----	---	----

parent 2:

1	3	11	8	7	10	2	3	2	4	8	10
---	---	----	---	---	----	---	---	---	---	---	----

parent i:

.....											
-------	--	--	--	--	--	--	--	--	--	--	--

Fig. 3. Chromosome initial coding

#### E. Annealing Algorithm Fundamentals

In general, the key to integrate the two algorithms lies in how to select the next generation, the SAGA selects the next generation with the higher probability of getting close to the target in the range of neighborhood, in which the algorithmic process is a constant random walk from one state to another. We can use Markov process to describe the transfer probability, acceptance probability.

The challenge of Annealing Algorithm are as follows:

(1) The initial temperature  $t_0$ . When the initial temperature is high, it is more likely to search for the global optimal solution, but it takes a lot of computation time. On the other hand, the computation time can be decrease, but the global search performance may be affected.

(2) The annealing speed. The global search performance is closely related to the number of iterations at each temperature  $t$ . "Full" search at the same temperature is quite necessary, but it also takes time to calculate. The increase of the number of loop cycles will inevitably lead to an augment in computation overhead. We use the following function [18]:

$$t_{k+1} = t_k(1 + \beta t_k)^{-1} \quad (10)$$

$$\beta = \frac{t_0 - t_f}{M_1 t_0 t_f} \quad (11)$$

the  $t_0$  and the  $t_f$  are default value,  $M_1$  is the number of iterations. In theory, SA can solve most of the optimization problems, but in practice, due to the global optimum annealing speed is too slow to be accepted. In this article, we select the cooling strategy as follows:

$$f_i(t_k) = \exp(-\frac{f(i) - f_{\min}}{t_k}) \quad (12)$$

while the  $f_{\min}$  denotes the minimum of fitness value in population. According to the Metropolis criterion [21], the probability that particles tend to equilibrate at temperature  $t$  is  $\exp(-AE / (kT))$ , where  $E$  is the internal energy at temperature  $T$ ,  $k$  is Boltzmann's constant. Metropolis formulation are often expressed as follows:

$$p = \begin{cases} 1 & \text{if } E(x_{new}) < E(x_{old}) \\ \exp(-\frac{f(j) - f(i)}{t_k}) & \text{if } E(x_{new}) \geq E(x_{old}) \end{cases} \quad (13)$$

which means the algorithm is more likely to accept sort-of-bad jumps rather than complete refuse to accept it, the probability gradually decreases over time till solution becomes global stability.

#### F. Crossover

The function of the crossover is to ensure the stability of the population and evolvment towards the optimal solution. Crossover can facilitate avoiding premature convergence. Chromosome crossover doesn't mean descendants are definite better than their parents, but represent the next generations have a better development tendency than the previous generation. There are many kinds of crossover methods, multi-point crossover refers to exchanging the multiple crossover points in an individual chromosome. In this paper, we use the method of position-based multi-point crossover (PBC) [20]:

As in Figure 4, first, we select a random pair of chromosomes (parents) in population, the location may not be continuous, but the two parents chromosomes were selected at the same location; Second, a pro-offspring with the guarantee that the selected gene position is located at the same with the parents position is generated. Third, we find the position of the gene selected in the first step in another parent, and then put the remaining genes sequentially into the pro-offspring generated in the previous step 2.

parent 1:

1	4	8	8	7	3	11	3	2	10	2	10
---	---	---	---	---	---	----	---	---	----	---	----

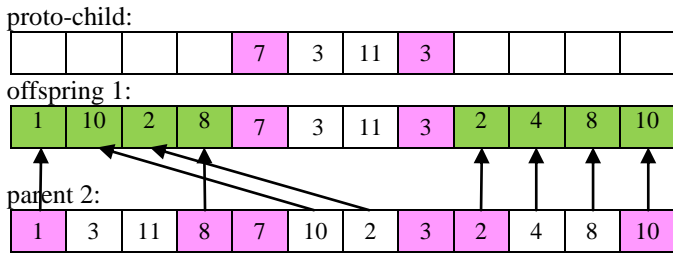


Fig. 4. Chromosome crossover

### G. mutation

The role of mutation is to ensure the vast diversity of the population with the operation that change the value of a particular gene from one generation of a population, that also can avoid the possible convergence of local convergence. In this work, the probability of mutation is set as 0.01~0.1.

## V. PERFORMANCE EVALUATION

In this section, we conduct experiments and evaluate the performance of our algorithm by using real trajectories dataset. We use the Beijing road data with about 300,000 vertices and 400,000 edges, and utilize two historical trajectories datasets, the Taxi, which contains about 100,000 trajectories of user orders generated by more than 5,000 public taxicabs in one month in Beijing, Ucar [7], which contains about 300,000 trajectories of user orders generated by more than 4,000 public taxicabs in two weeks in Beijing. The riders' trajectories include start and destination point, earliest start time, latest start time, latest reach time, the drivers' trajectories includes start and destination point, earliest start time, latest start time. We also simulate some drivers trajectories by using history hottest start and destination points. Experimental settings: All experiments are run on a machine equipped with 3.6 GHz Intel Core i7-4790 CPU, 16GB RAM, window7 OS and algorithms implemented in Python 27.

We evaluate the algorithms mainly from two aspects, the running time, the shared path ratio. We compare our algorithm with the XAR [11], TGA [15]. We conduct our experiments for different number of drivers from 3000 to 15000 with a fixed number of riders (i.e., 9000). By default,  $t_0 = 100$ ,  $t_f = 10$ , and  $M_I = 1000$  for SAGA algorithm. We make use of the grid matrix in Section 4 when compute the shortest distance.

### A. Efficiency

We assess the efficiency by fixing the number of drivers as 9000 while varying the number of riders from 3000 to 12000

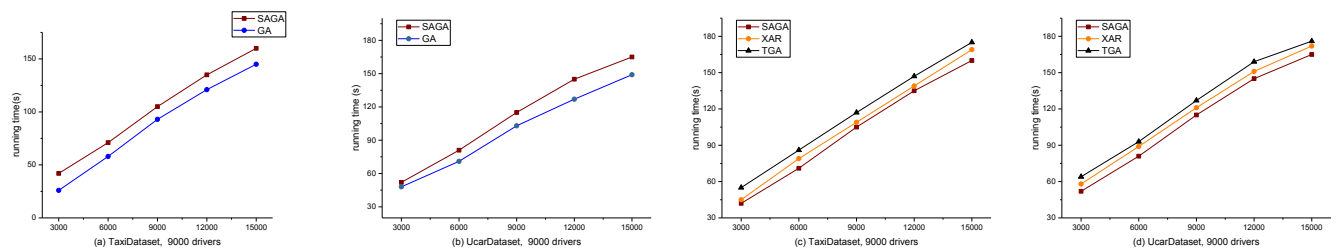


Fig. 5. Comparison of running time taken by SAGA, GA, XAR, TGA

presented by x-axis. In Figure 5a, 5b, we set the same number of iterations as 800 for SAGA and GA. Figure 5 shows the results corresponding to different algorithms. We can see that SAGA is greater than the pure genetic algorithm(GA). Meanwhile, since the SAGA takes advantage of a fast acceleration adaptation function, the running time is not too high. From Figure 5c and Figure 5d, our algorithm running time is superior to XAR and TGA.

### B. Effectiveness

Figure 6 shows the performance of average shared path ratio. The number of iteration is set as 800 for SAGA and GA. We have the following observations that the matched ratio of SAGA is higher than the GA, because the SAGA search in the range of chromosome neighborhood, thereby avoiding falling into local optimal. Figure 6c and Figure 6d show that our algorithm ratio is within 72%-75%, and is better than baseline algorithms within 65%-54%.

### C. The effects of the parameter : descent speed $M_I$

We evaluate how the annealing descent speed affects the running time and the matching ratio. We set parameter  $M_I$  as 500, 1000, 1500 for SAGA\_t1, SAGA\_t2, SAGA\_t3 respectively, the results are shown in Figure 7. The larger the  $M_I$  is, the faster the evolution change. We have the following observations in Figure 7, as  $M_I$  grows, the running time also increase, this is because it requires more loop iteration calculations. Meanwhile, from Figure 7b and Figure 7d, the approximate matched ratio enlarge as well, because it is possible to gain more candidate chromosomes from the neighborhood of old chromosomes.

## VI. CONCLUSION

In this work, we utilize real dataset to model a ride-sharing problem and present a simulated annealing genetic algorithm to address it. The proposed SAGA outperforms commonly baseline algorithms including XAR and TGA. We run a careful iterative turning process, and the extensive experiments on large car services datasets show the advantage of our algorithm.

There are three interesting directions for the future work, (a)we will incorporate traffic conditions to assemble the candidate riders, (b)consider the ride-sharing with social networks, (c)introduce artificial intelligence algorithm to solve the ride-sharing problem.

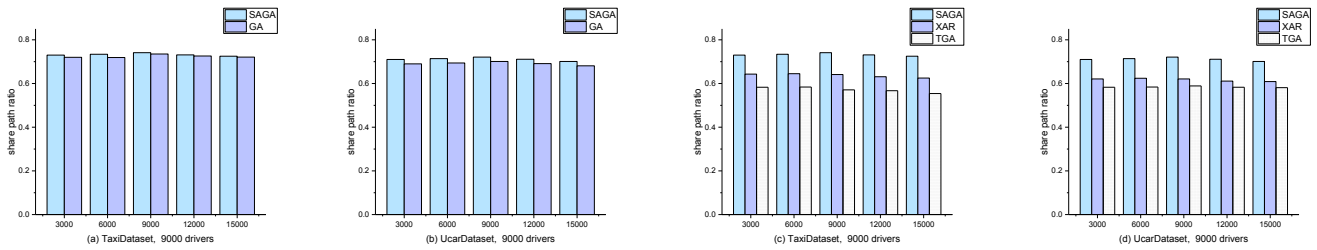


Fig. 6. Comparison of share path ratio taken by SAGA, GA, XAR, TGA

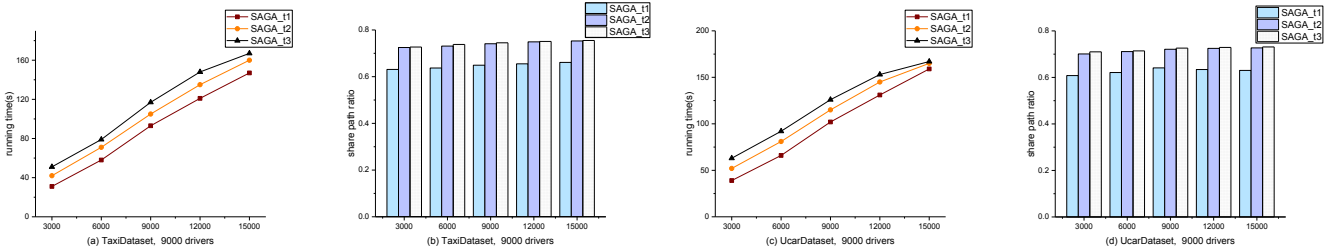


Fig. 7. Performance of running time and share path ratio by varying descent speed  $M_1$

### ACKNOWLEDGMENT

This work was supported by NSFC(91646202), the National High-tech R&D Program of China(SS2015AA020102), Research/Project 2017YB142 supported by Ministry of Education of The People's Republic of China, the 1000-Talent program, Tsinghua University Initiative Scientific Research Program.

### REFERENCE

- Bartolini E, Bodin L, Mingozzi A. The traveling salesman problem with pickup, delivery, and ride - time constraints[J]. *Networks*, 2016, 67(2): 95-110.
- Agatz N, Erera A, Savelsbergh M, et al. Optimization for dynamic ride-sharing: A review[J]. *European Journal of Operational Research*, 2012, 223(2): 295-303.
- Deb K, Pratap A, Agarwal S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II[J]. *IEEE transactions on evolutionary computation*, 2002, 6(2): 182-197.
- Z. Chen, H. T. Shen, X. Zhou. Monitoring path nearest neighbor in road networks. *SIGMOD*, pages 591-602, 2009.
- Goel P, Kulik L, Ramamohanarao K. Optimal pick up point selection for effective ride sharing[J]. *IEEE Transactions on Big Data*, 2017, 3(2): 154-168.
- Asghari M, Shahabi C. An On-line Truthful and Individually Rational Pricing Mechanism for Ride-sharing[J]. 2017.
- Ta N, Li G, Zhao T, et al. An Efficient Ride-Sharing Framework for Maximizing Shared Route[J]. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2017.
- Jia Y, Xu W, Liu X. An Optimization Framework For Online Ride-sharing Markets[C]//*Distributed Computing Systems (ICDCS)*, 2017: 826-835.
- Furuhata M, Dessouky M, Ordóñez F, et al. Ridesharing: The state-of-the-art and future directions[J]. *Transportation Research Part B: Methodological*, 2013, 57: 28-46.
- Ma S, Zheng Y, Wolfson O. T-share: A large-scale dynamic taxi ridesharing service[C]//*Data Engineering (ICDE)*, 29th International Conference on. IEEE, 2013: 410-421.

- Thangaraj R S, Mukherjee K, Raravi G, et al. Xhare-a-Ride: A Search Optimized Dynamic Ride Sharing System with Approximation Guarantee[C]//*Data Engineering (ICDE)*, 2017 IEEE 33rd International Conference on. IEEE, 2017: 1117-1128.
- Alarabi L, Cao B, Zhao L, et al. A demonstration of SHAREK: an efficient matching framework for ride sharing systems[C]//*Proceedings of the 24th ACM SIGSPATIAL*. ACM, 2016: 95.
- Qian S, Cao J, Mou F L, et al. SCRAM: a sharing considered route assignment mechanism for fair taxi route recommendations[C]//*Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015: 955-964.
- Huang Y, Bastani F, Jin R, et al. Large scale real-time ridesharing with service guarantee on road networks[J]. *Proceedings of the VLDB Endowment*, 2014, 7(14): 2017-2028.
- Herbawi W, Weber M. The ride matching problem with time windows in dynamic ridesharing: A model and a genetic algorithm[C]//*Evolutionary Computation (CEC)*, 2012 IEEE Congress on. IEEE, 2012: 1-8.
- Herbawi W M, Weber M. A genetic and insertion heuristic algorithm for solving the dynamic ridematching problem with time windows[C]//*Proceedings of the 14th annual conference on Genetic and evolutionary computation*. ACM, 2012: 385-392.
- Shen B, Zhao Y, Li G, et al. V-Tree: Efficient kNN Search on Moving Objects with Road-Network Constraints[C]//*Data Engineering (ICDE)*, 2017 IEEE 33rd International Conference on. IEEE, 2017: 609-620.
- Wenxun Xing, Jinxing Xie. *Modern optimization calculation method*[M]. Tsinghua University Press, 2005
- Pham D, Karaboga D. *Intelligent optimisation techniques: genetic algorithms, tabu search, simulated annealing and neural networks*[M]. Springer Science & Business Media, 2012.
- Razali N M, Geraghty J. Genetic algorithm performance with different selection strategies in solving TSP[C]//*Proceedings of the world congress on engineering*. Hong Kong: International Association of Engineers, 2011, 2: 1134-1139.
- Van Laarhoven P J M. *Simulated annealing*[M]//*Simulated annealing: Theory and applications*. Springer, Dordrecht, 1987.

# A Multiple-Level Assessment System for Smart City Street Cleanliness

Wenrui Li<sup>1</sup>, Bharat Bhushan<sup>2</sup>, Jerry Gao<sup>2,3\*</sup>

<sup>1</sup>School of Information Engineering, Nanjing Xiaozhuang University, Nanjing, China

<sup>2</sup>Department of Software Engineering, San Jose State University, San Jose, USA

<sup>3</sup>School of Information and Computer, Taiyuan University of Technology, Taiyuan, China

Email Address: {wenrui\_li@163.com; Bharat.Bhushan@sjsu.edu; jerry.gao@sjsu.edu}

**Abstract**—Advancement in mobile, cloud technologies and IoT has made the world even smaller and connected like never before. It has become a challenge and an opportunity for cities to leverage these growing technologies to solve real city administration problems. Cities are in the transformation state to become state of the art smart-city using these technologies. This paper is about the automation of street cleanliness assessment in near real-time. It answers the question of how can we assess the status of streets more efficiently and effectively. In order to address the problem, this paper proposes a multiple-level assessment service system on how the cleanliness status of streets is collected using mobile stations, connected via city network, analyzed in the cloud and presented to city administrators online or on mobile. The real applications show the usability and feasibility of our system. This also gives opportunities to city residents to participate and contribute towards making the city a better place.

**Keywords**- Smart City, Street Cleaning, Litter Detection, Machine Learning, Cloud Computing, Dashboards

## I INTRODUCTION

Street cleaning is an important city service, which involves a set of activities concerning the cleanliness of the street (usually defined as pavements and adjoining edges of roads and grassed and planted areas) [1]. Therefore, it involves street-sweeping (whether manual or machine), litter-picking, the uplift of fly-tipped refuse and the removal of graffiti and flyposting. When the street cleaning service is ineffective, the evidence is visible. And it could cause a significant impact on the quality of life and the attractiveness of its neighborhoods, towns and cities [2][3]. Moreover, people believe that there are the links between environmental problems and other forms of disorder and crime in cities [4]. On the other hand, good quality street cleaning service in a city provides and contributes the good environmental quality in its communities and neighborhoods, which can help urban development, make places attractive to tourists, investors and mobile workers [5]. Moreover, the effective street cleanliness could reduce the costs in cleaning underground water systems for cities.

In 2016, the city of *San Jose* has set up its goal for 2020 to provide a real-time city dashboard based on its city map to present the city street cleanliness to the public. The major objective is to provide more cost-effective and efficient street clean services and improve the quality of life for its neighborhoods by providing a more attractive

clean environment. Since 2017, the research center of Smart Technology, Computing, and Complex System in *San Jose* State University has been teamed up with the Environment Service Department (EDS) in the City of *San Jose* to develop an innovative smart digital street cleanliness assessment service system, known as *SmartClean*, based collected camera-based images using a cruise car on city streets. *SmartClean* is developed based on a well-defined street cleanliness assessment model with digital standards using machine learning technologies. In summary, the major contributions and distinct features of *SmartClean* service system for the city of *San Jose* are highlighted below..

- The system implemented the first and innovative hierarchical grid-based city street cleanliness assessment model [14] for digital processing, map-based regional analysis, street cleanliness pattern analysis and prediction.
- The system is the first one supporting automatic detection of diverse city street litter objects using machine learning techniques [13] based on collected camera images from the mobile stations as well as users' images. Different neural network models are used here. Based on our recent case studies in 2017, different types of litter object detection could achieve over 70% to 95% accuracy for major classes [13].
- After the system deployment, the system will be the first cloud-based comprehensive mobile enabled service system for city street cleanliness assessment and diverse services.

The rest of the paper is organized as follows. Existing work about smart city cleanliness is surveyed in Section II. Section III gives an overview and detail of our assessment model used in the paper. Section IV shows the architecture of our model. The implementation and case study are provided in Section V. Finally, Section VI concludes the paper.

## II RELATED WORK

Based on our recent online search and literature survey about the city street cleanliness, we found that many cities, industries, and IoT evangelists are still talking and looking for innovation solutions to address this challenge and need, even though there are some publications discussing the related issues and solutions. In addition to some case study reports [1][6][7], the closely related initiatives and solutions are summarized below.

\*Corresponding Author  
10.18293/SEKE2018-101

*Clean Street LA* [8] is an initiative challenge launched by the Landon City Mayor, and its objective is to use the ESRI GIS tool to map and plot the street cleanliness status block by block. Multiple layers and grids are created to reflect different parts of the city. Cleanliness information on the streets with a cleanliness score is visualized on a map. This information is used to decide the area that requires attention and cleaning services. However, the limitation of the system is that the monitoring is limited to Garbage bins and cannot be extended to monitor city streets.

*LondonCleanStreets* (UK) [9] is a crowdsourcing-based system which can keep the streets of London clean with cloud computing. Also, *LoveCleanStreets* is a crowdsourcing based online portal, which has a mobile app for the public to snap pictures and submit to local councils. This is widespread in UK and surrounding countries. It has an interactive map using Microsoft silver-light technology. Reports illegal dumping, potholes, and graffiti complain and shows clean-up time has improved by 87%.

*SmartBin* [10] is an intelligent monitoring solution which can enable waste management and recycling companies to optimize their collection operations and maximize the use of valuable resources. They do this by deploying SmartBin wireless ultrasonic sensors to a wide range of containers and using the data intelligence to drive operational efficiencies including optimized routes, asset tracking, and cost analysis. SmartBin sensors leverage the latest in IoT and cellular network technologies. However, the limitation is that the information is fed manually into the system. Also, the system is not real time and requires human intervention to update the status.

*Spot Garbage* [11] is a crowdsourcing approach based on the public inputs about the street clean status, where a crowdsourced platform is developed to use the pictures contributed by the public. The process of trash detection is automated. The limitation is that the litter in the images is not classified into object classes.

In [12], a mobile app is developed to evaluate the Street Cleanliness and Waste Collection Service. This app is based on a Plan of Indicators that can be used to evaluate the Street Cleanliness and Waste Collection Service of Santander municipality. Specific methodologies for calculating and evaluating 59 indicators have been developed to obtain information regarding the status of the different elements of the service. Pearson correlation coefficient results suggest that an inverse relationship between the Street Cleanliness Index values and the Frequency Street Cleanliness Services/population density ratio exists.

### III ASSESSMENT MODEL - LDAS

#### A. Overview

Fig.1 shows the various modules and the interconnection

between them. The proposed system has three layers.

- **Edge:** This layer is the layer where the data collection takes place in the form of images from streets. This data along with location coordinate is sent to cloud for processing.
- **Cloud:** This is the layer where the images are processed using analytical tools, created or fed the training model. Results from this layer fed to the end user database for visualization and reporting.
- **User:** This is the layer where reports are generated based on the Cloud processing. These results are visualized for city and community.

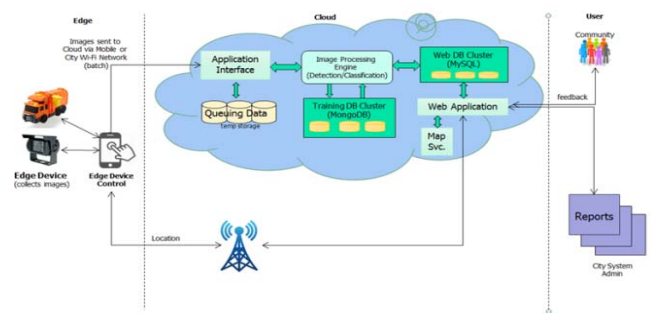


Fig. 1 the Assessment System Overview

The proposed model depends on various attributes which need to have well defined before and are assumed.

Assumptions are made in the following:

- Fixed image resolution and camera angle.
- Vehicle speed is approx.15 – 25 mph.
- Picture covers 150ft. of distance.
  - Pictures are collected every ~3 sec.
  - Set of four pictures are collected every spot in each direction.
  - Multiple pictures collected to increase the confidence level of the machine learning system.
- Start and End Locations are predefined.
- Stable Cloud connectivity requires continuous stable connection in order to achieve online image transmission or offline image transmission
- Data is unique and is collected on a weekly basis for the entire city. This can be customized.

#### B. Multi-level Assessment Model

Litter Detection Assessment System (LDAS) providing the cleanliness assessment is done in layers. Lower layers contribute to layers above. Top layer generalizes the results of layers below. On a high level this model is divided into four layers.

**Layer 1** is the first layer which defines the city area overall and sets the scope of assessment. This covers all



streets in the city and is the base layer. **Layer 2** is the second layer where a city is divided into areas from Layer1 based on the city plan. Each area has a code. It may not be same as zip code but would be a group of GPS locations and distance.

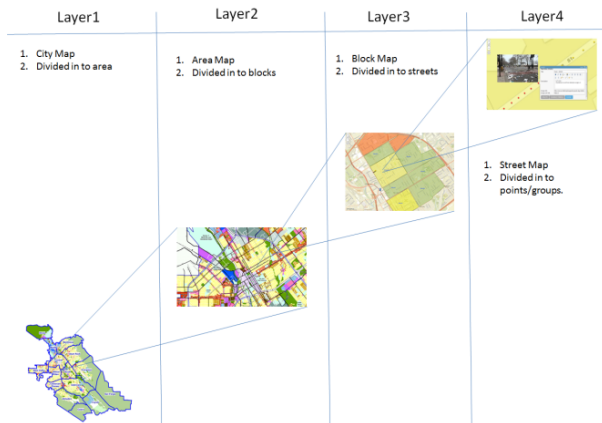


Fig. 2 Assessment Layers for San Jose.

**Layer3** is the third layer where each area is divided into blocks. Each block is uniquely identified by a combination of area and block name. **Layer4** is the top Layer which represents individual streets in the block with individual data collection points called grid points.

Fig.3 shows the hierarchical view of *San Jose* City. From the figure, we can see that city represents entire city divided into different areas. It is divided into no. of Areas (represented by a circle) E.g. Central *San Jose* or Alum Rock. Each Area has a number of blocks e.g. 20 in Central and 16 in Alum Rock. Color represents the cleanliness level of area overall. Area Value (AV) is indicated with an average of results from each block within the area.

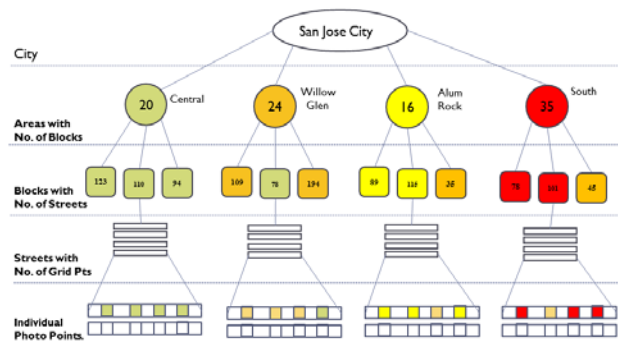


Figure 3: Multi-level Assessment Model

A grid is used to describe detail pictures of each point. The formal definition of a grid is described in the following:

Fig. 4 shows an example of the grid architecture. In the figure, all blocks are divided into several *Grid Point* and *INodes*. Then *INodes* are also divided into subsections

with *SNodes*. Each *grid point* has at most four picture points.

All blocks collectively represent an area. The different color-coding scheme is used to represent cleanliness level for each area. Table 1 describes the different street assessment levels. Grid Point is composed of all the picture points in 150ft Radius. All pictures within this range generate the collective value which indicates the cleanliness level of the point. Collection of grid points represents the Block. This includes the streets within the block based on the latitude (lat.) and longitude (long.) and distance in miles but can be customized. It is assumed that the distance between two grid points is 150 feet.

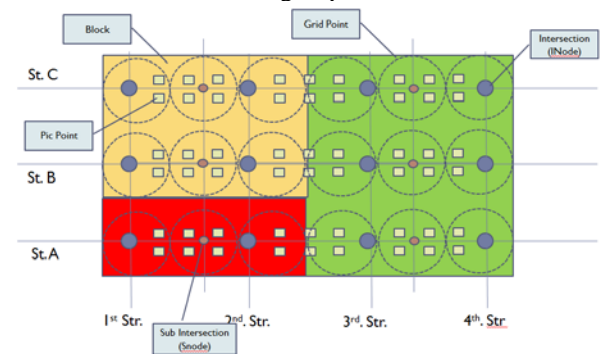


Fig. 4 Layer Point Plot.

Table 1. litter levels along with color code

Color	Level	Desc.
Green	A or 1	Not Littered
Yellow	B or 2	Slightly Littered
Orange	C or 3	Littered
Red	D or 4	Very Littered

Group Point is composed of all the picture points in 150ft Radius. All pictures within this range generate the collective value which indicates the cleanliness level of the point. Collection of grid points represents the Block. This includes the streets within the block based on the latitude (lat.) and longitude (long.) and distance in miles but can be customized. It is assumed that the distance between two grid points is 150 feet.

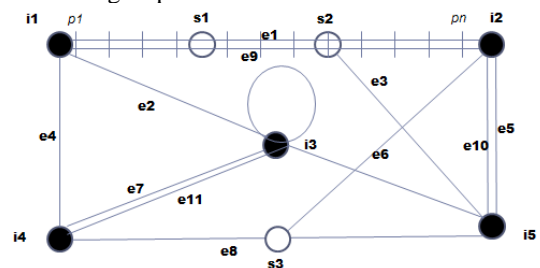


Fig. 5 Block Nodes and Edges.

Fig. 5 shows a block with intersections represented in solid dark circles as Nodes  $i_1, i_2, i_3 \dots i_N$ , white circles represent the intermediary road joins. Joining two nodes are Edges represented by  $e_1, e_2, e_3 \dots e_N$ . Each Edge

represents one-way traffic. Points, where pictures are taken, are represented as  $p_1, p_2, \dots, p_N$ . Directed Graph (G) is used to represent entire city as one big graph as each road has a direction.

A graph is defined as  $G = (N, E)$  where  $N$  is the set of nodes and  $E$  is the set of edges. Each edge  $e \in E$  would have a starting node and ending node represented as  $e = (i, e)$  where  $i$  is nodes and  $e$  is edges.

$$e = (i \text{ (source)}, I \text{ (term)}) \quad (1)$$

Each photo point (shown as Pic Point in Fig. 4) has a value from the ML system represented by pV (point value) between  $i_1$  and  $i_2$ . All values for each point are added and the average is generated for each photo point pV using:

$$pV_{(i_1-i_2)} = \sum_{p=1}^n pi / n \quad (2)$$

Once pV is calculated, the sum of  $n$  pV in a range from both directions are averaged to generate the average value of grid point (shown as a circle in Fig. 4) gV (grid Value) using:

$$gV = \sum_{g=1}^n gi / n \quad (3)$$

Block constitutes all gV in a distance  $x$  which is a custom value and is a collection of all the pic point and respective grid points as you can see in red, yellow and green. Each color represents block average (bV) representing the cleanliness level. This is calculated by the average value of all the grid points in a block using:

$$bV = \sum_{b=1}^n bi / n \quad (4)$$

Fig.4 is also an example of the visualization of representing the calculations above. Validation at each photo point:

1. Four pictures from each camera are clicked in quick succession.
2. All pictures are associated with the date and time and also with the geo location in the as latitude and longitude.
3. Each camera has a predefined angle and resolution which can be customized and covers defined range 20ft.
4. All pictures are sent to cloud for analysis and assessment value is returned.

From Fig. 5  $i1$  and  $i2$  are connected with two edges and similarly  $e5$  and  $e10$  are. These two edges represent two lanes of the same road. Every picture point on either side of the road would be part of same grind point and will be validating each other to confirm the cleanliness of the same point. This would base on the GPS coordinate of each photo point.

All blocks collectively represent the area. The different color coding scheme is used to represent the cleanliness

level. Tab. 1 describes the litter levels along with the color code.

#### IV ARCHITECTURE

Smart City Infrastructure has various components. Below shows the components required for the cleaning infrastructure. We are going to explore the use and application of machine learning, mobile, and cloud computing, Big-data, Databases, Web and UI technologies and solutions to address the city cleaning issues. Big data analytics implementation along with machine learning would create an autonomous system, which shall serve as a self-detection engine to analyze the data on the fly. Mobile system using the latest available wireless and Wifi networks would establish the secure network connectivity between the edge device, cloud, and the city infrastructure.

The system is divided into three major segments, which is shown in Fig.2. Below is the high-level view of each segment.

**Edge:** This is the edge layer where data collection takes place in the form of pictures and location coordinates from streets using a smart vehicle with a camera called Mobile Station. This data along with time stamps are sent to cloud for processing over wireless or Wifi networks. At this layer, mobile station controller and mobile apps are the interfaces to the users. A mobile app is created to simulate this layer.

**Cloud/Server:** This is the layer where the images are processed using analytical tools, created or feed to the training model. Results from this layer fed to the end user database for visualization and reporting. This layer runs various services like web application servicer to connect to multiple mobile stations and receive data, database services, and Web server services. In our practical system, Amazon Web Service (AWS) is used to simulate the cloud enabling the connectivity between the mobile stations and cloud services for city administrators.

**User:** This is the layer where users, city administrators and residents can interact with the system. A dashboard created for city admin to monitor and control the system as a single pane of glass. A map view can see the street cleanliness level. Analytics shows various statistics on different sections of the city, management, and feedback of the system. Various reports can be generated based on the user requests. These results are then visualized for city and community.

##### A. High-Level System Layers.

The approach is composed of different layers in order to have more flexibility and scalability. Each layer has a different component and functions; which are maintained independently to minimize the impact on the entire system. Lower layer contributes to the layer above. Upper layers depend on lower layers. High availability is set to be achieved by using a clustered approach in the cloud. At

a high level, there are two partitions of the system as shown in Fig. 7.

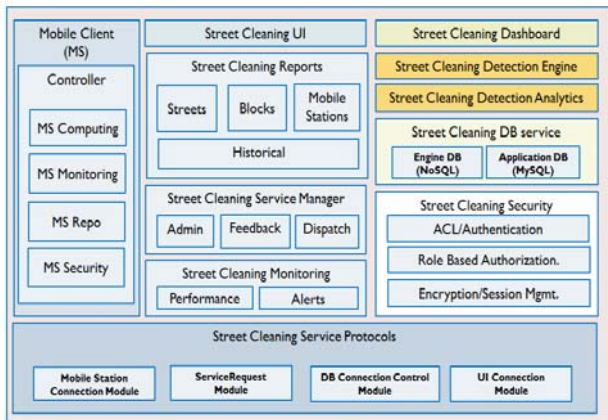


Fig.7 High-level Architecture

**Edge Service or Mobile Station** – This is the set of services run and managed on Mobile stations, which also serves as the client to cloud system. These are trucks or special vehicles, which connect to cloud service and updates continuously with latest updates. They have a local controller to manage edge service and have their own computer, repository, security and communication protocols for cloud connection. There is also a mobile app where the users can communicate with the cloud. This layer is heavily dependent on the network and needs to maintain a persistent connection with the cloud in order to have a real or near real-time communication.

**Cloud Service or Server Side** – This service includes various sub service and is composed of many modules or layers, which interact with clients and with each other and are running continuously. UI and Dashboards are setups for city administrators and other city officials. Part of it would be visible to public. Reporting and Analytics to reflect the assessment of the streets, blocks, zip codes, and city as a whole. It has management functionality for admin to manage the system. Various backend service like database and monitoring runs to monitor the system. It also has a detection engine, which is a machine learning based analytical engine running visual recognition algorithm to classify the images for cleanliness levels. These services designed to be highly available and visible to city admins and truck drivers and other users.

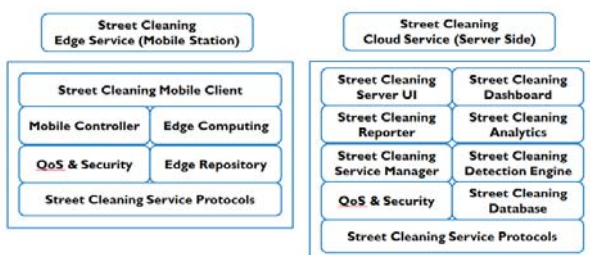


Figure 8: Detailed Systems Architecture.

## B. Detailed Architecture

Architectures with all modules are described below. Figure 8 shows all the components with every submodule we have.

### 1) Mobile Station Controller

This component controls the functionality of the mobile station. It is a specialized device mounted on every mobile truck and manages the services like connectivity to cloud, compute resources, local storage, monitoring, and security. It notifies the city admin for any alerts or issues with the mobile station. It enables the camera to take pictures, store them locally and transmit them using the wireless network to cloud.

### 2) Street Cleaning UI

Cloud user interface offers services to users who can view the current status of streets or blocks. City admin can view the status of mobile stations. Cleaning dispatch can be made using this interface and feedbacks received can be handled from the same view. System performance and alerts are also managed from UI by respective city admins. This UI is linked to the database and all events are logged. City admin can manage users like user registration, approvals, and access control. All system alerts and any feedback received can be acknowledged from this same view.

### 3) Street Cleaning Dashboard

Cleaning dashboard is the module where the status of the entire system is visible to city admin. This includes cleanliness status in grid view, on street basis. Mobile station status is monitored from the same interface. There is a submodule known as Street Cleaning Detection Engine with two subcomponents names Street Cleaning Analytics and Metrics, which are an external system. Our project when integrated with this external system would enable getting image processed for cleanliness level. This also includes database services for various engines and API end-points, which are being monitored from this dashboard along with system performance.

### 4) Street Cleaning Service Protocols

Service protocol refers to the inter-connectivity between various components of the system. This connectivity enables the data flow from mobile station to end users. It connects all mobile stations via wireless protocols including Mobile networks like 4G or LTE or city Wi-Fi. This layer also handles communication between various components of the cloud from application to database to detection engine.

Mobile Stations are connected to cloud services over the Internet, which rely on the wireless or Wi-Fi network. 4G or better connectivity is required for real-time communication. As soon as mobile station starts, it makes various network connectivity checks. Once connectivity is established, TCP connections are established between a mobile station and cloud services. Custom ports are used for secure communications.

Service Requests are handled on cloud services and use secure TCP protocol. It also uses authentication for each mobile station. Every mobile station is uniquely identified using a combination of use rid and mobile station id. It can be further enhanced at camera level.

DB connection control mechanism handles the connections to a database. In our case, most of the connections are made from cloud service for mobile and the application service. All DB connection is made over TCP and is authenticated. For security reason, we have kept DB services on a separate instance and it also decouples the app from database layer and gives the flexibility. UI connection modules are plugged into the web application service. Users are authenticated using a common separate database over TCP.

## V IMPLEMENTATION

An attempt is made to simulate the end-to-end functionality using a mobile app, cloud service and a UI. Machine Learning is an external module. Mobile stations (MS) are connected via mobile network preferred 4G/LTE. Testing is done by creating a mobile app. MS would be specialized devices with the specifications provided by a city.

### A. Use case:

Data is collected at mobile stations and then fed to a database in cloud for analysis and modeling. We created a mobile app to simulate same and collected actual data in San Jose city. This app at start performs checks like Network connectivity, Storage space on device and connectivity to cloud service. If any of the check fails it can report to administrator via email or SMS (text) to predefined contact. Data collected can be transmitted to cloud in real time or offline depending on network connectivity.

Certain assumptions were made to perform test. All images collected are transferred to cloud based database and then fed to machine learning for analysis. Data can be downloaded manually in case network connectivity is not available.

On the cloud side, data receiving application service run continuously which is used by all mobile stations. Images are stored on filesystem and metadata is stored in database for each image. We used AWS (Amazon Web Services)<sup>[11]</sup> cloud service to simulate the cloud connectivity and able to transmit data from mobile to cloud and update database. APIs are used to connect to machine learning module to get the result on each photo.

Application service connects to machine learning system and to the front-end database. NoSQL<sup>[12]</sup> and MySQL based databases are used to store and manage the data. NoSQL database would be used to have a detailed analysis and would be recording the fine details of each picture with assessment level. This would also be the base of all historic data and feed the trends and decision

making for future. MySQL would be a backend database for the dashboards.

User dashboards are created for city and residents. This also has an admin module which can be integrated with other city systems like weather, traffic or dispatch. Residents can also view the latest activities on the street cleaning and can contribute as a feedback or suggestions.

### B. Analysis – Point Level

Point level analysis is the core and the granular level of assessment. This is the key for entire system. Below are the attributes of the point analysis. At least four pictures are taken at every point. Two points distance is customizable by admin and will be defined as part of the configuration. From the experiences, we have taken 20Ft. as a distance based on the speed limits. Pictures will be sent to the cloud along with meta-data i.e. date, time, location, image path. Log information will be stored on device periodically. Each point will be represented in a different color to indicate the level cleanliness level. Figure 9 shows an example point level analysis, where a figure is shown in four pictures, left, front, right and back.



Figure 9: Picture Point.

### C. Analysis – Street Level

From each point on a street between start (S) and end (E) points, all numbers would be averaged to generate an overall assessment of the street. The assessment would be done for every street. Results are stored in DB with image reference, date time and level. Each street is a part of one block. Grid based analysis and part of the block. Figure 10 is an example of street level analysis.



Figure 10: Street Level

#### D. Analysis – Block Level

It is an aggregate of all the points on the block. Block Assessment Value is an aggregate value of all streets. The assessment would be based on every street in the block and the aggregate value. Results are stored in DB with image reference, date time and level. Block can have any number of streets. Everything is based on each data points. Figure 11 is an example of block level analysis.

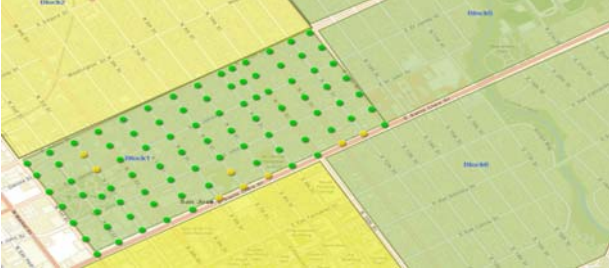


Figure 11: Block Level Analysis

#### E. Analysis – Area level

Based on the block level analysis, area level analysis can also be performed. Figure 12 is an example of area level analysis. Area is a combination of many blocks with different colors.

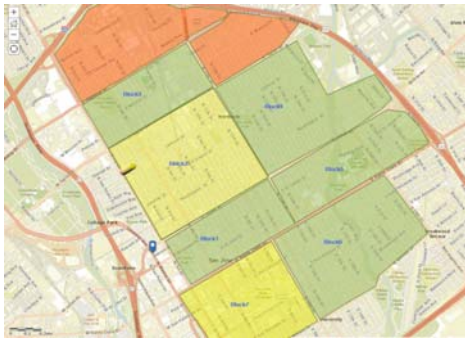


Figure 12: Area Level Analysis

### VI CONCLUSION

In order for a city to become a smart city, infrastructure needs to be upgraded to next level. A clean city is the essential component and reflects the image of the city for residents and visitors. This system can also be well integrated with other city systems like traffic, weather etc. Residents can also contribute and help city using this system by providing feedback and by reporting any issues identified by them. This system can be implemented by any city and in fact, this can act as one big system as a grid of cities. Security and access can be controlled by individual city administrators.

### VII ACKNOWLEDGMENT

The authors thank the strong support from SJSU

Excellence Research Center on Smart Technology, Computing, and Complex Systems, and Jesse Huang and Rob Xiaoming Ding at Futurewei Technologies, Inc. and Jo Zientek, Anna Szabo, and Ed Ramirez at Environmental Service Department in the City of San Jose, especially. This work was also supported in part by the Research Fund of Nanjing Xiao Zhuang University under Grant 2016NXY16, and the Key Laboratory of Trusted Cloud Computing and Big Data Analysis under Grant 15BDA02.

### REFERENCES

- [1] Annette Hastings, Nick Bailey, Glen Bramley, Rob Croudace, and David Watkins, "Street cleanliness in deprived and better-off neighborhoods - A clean sweep?", Technical Report, November 2009, URL: <https://www.jrf.org.uk/sites/default/files/jrf/migrated/files/neighborhood-street-cleanliness-full.pdf>
- [2] A. Parkes, A. Kearns, and R. Atkinson, (2002), "What makes people dissatisfied with their neighborhoods?", *Urban Studies*, Vol. 39, No. 13, pp. 2413–38.
- [3] Silverman, E., Lupton, R. and Fenton, A. (2006), "Attracting and Retaining Families in New Urban. Mixed Income Communities", York: Joseph Rowntree Foundation.
- [4] Home Office (2006) Respect Action Plan. London: Home Office.
- [5] Hastings, A., Flint, J., McKenzie, C. and Mills, C. (2005) *Cleaning Up Neighborhoods: Environmental Problems and Service Provision in Deprived Areas*. Bristol: The Policy Press.
- [6] Edinburgh Council, UK, "Street cleanliness", URL: [www.edinburgh.gov.uk/download/downloads/id/3273/soe\\_street\\_cleanliness.pdf](http://www.edinburgh.gov.uk/download/downloads/id/3273/soe_street_cleanliness.pdf).
- [7] Hastings, A., Bailey, N., Bramley, G., Croudace, R. and Watkins, D., 2009. *Street cleanliness in deprived and better-off neighbourhoods*, URL: <https://www.jrf.org.uk/report/street-cleanliness-deprived-and-better-neighbourhoods-clean-sweep>.
- [8] Clean Street LA, "Clean Streets LA Challenge", <http://cleanstreetsla.com/clean-streets-challenge/>
- [9] A. E. F. Seghrouchni, F. Ishikawa, Hérault Laurent, and H. Tokuda, *Enablers for smart cities*. London: ISTE, 2016.
- [10] Sharma, Narayan, Nirman Singha, and Tanmoy Dutta. "Smart Bin Implementation for Smart Cities." *International Journal of Scientific & Engineering Research* 6.9 (2015): 787-791.
- [11] Spot Garbage - Detects Garbage using Artificial Intelligence, <https://www.youtube.com/watch?v=cTkCsZ5C8zs>
- [12] López, I., Gutiérrez, V., Collantes, F., Gil, D., Revilla, R., & Gil, J. L. (2017). Developing an indicators plan and software for evaluating Street Cleanliness and Waste Collection Services. *Journal of Urban Management*.
- [13] Chandni Balchandani, Rakshith Koravadi Hatwar, Parteek Makkar, Yanki Shah, Pooja Yelure, Magdalini Eirinaki: A Deep Learning Framework for Smart Street Cleaning, *IEEE BigDataService 2017*: 112-117.
- [14] Wenrui Li, Bharat Bhushan, Jerry Gao, "A Multiple-Level Assessment Model and System for City Street Cleanliness", Technical Report, 2017, which has been submitted for publication in 2018...

# Method and System for Detecting Anomalous User Behaviors: An Ensemble Approach

Xiangyu Xi<sup>\*†</sup>, Tong Zhang<sup>\*†</sup>, Guoliang Zhao<sup>§</sup>, Dongdong Du<sup>†‡</sup>, Qing Gao<sup>‡</sup>, Wen Zhao<sup>†</sup> and Shikun Zhang<sup>†</sup>

<sup>\*</sup>School of Software and Microelectronics, Peking University

<sup>†</sup>National Engineering Research Center for Software Engineering, Peking University

<sup>‡</sup>School of Electronics Engineering and Computer Science, Peking University

<sup>§</sup>CASIC-CQC Software Testing and Assessment Technology(Beijing) Corporation

email:{xixy,zhangtong17,dudong,gaoqing,zhaowen,zhangsk}@pku.edu.cn

**Abstract**—Malicious user behavior that does not trigger access violation or data leak alert is difficult to detect. Using the stolen login credentials, the intruder doing espionage will first try to stay undetected, silently collect data that he is authorized to access from the company network. This paper presents an overview of User Behavior Analytics Platform built to collect logs, extract features and detect anomalous users which may contain potential insider threats. Besides, a multi-algorithms ensemble, combining OCSVM, RNN and Isolation Forest, is introduced. The experiment showed that the system with an ensemble of unsupervised anomaly detection algorithms can detect abnormal user behavior patterns. The experiment results indicate that OCSVM and RNN suffer from anomalies in the training set, and *iForest* gives more false positives and false negatives, while the ensemble of three algorithms has great performance and achieves recall 96.55% and accuracy 91.24% on average.

**Index Terms**—anomaly detection, insider threat, user behavior, unsupervised learning, ensemble

## I. INTRODUCTION

Insider threat has emerged in enterprise security and received increasing attention over last several years. A survey [1] by Haystack shows 56% of respondents feel that insider attacks have become more frequent. Privileged IT users such as administrators with access to sensitive information, pose the biggest insider threat. IT assets such as databases, file servers and mobile devices are top assets at risk.

Insider threat is defined as any activity by military, government, or private company employees whose actions or inactions, by intent or negligence, result (or could result) in the loss of critical information or valued assets [2]. Two types of insider threats are distinguished: malicious insider threats and unintentional insider threats [3]. The first threat is a current or former employee, contractor, or business partner who has or had authorized access to an organizations network, system, or data and intentionally exceeded or misused that access in a manner that negatively affected the confidentiality, integrity, or availability of the organizations information or information systems. The attempted attack by a Fannie Mae employee after being dismissed is a typical example of an insider threat likely motivated by revenge [4]. The second form is from insiders without malicious intent [5] such as human mistakes, errors.

This work was partially supported by National Key Research and Development Program of China (No. 2017YFB0802900).

DOI reference number: 10.18293/SEKE2018-036

A key problem discussed frequently is to detect compromised user accounts and insiders within the company, which does not induce enormous data flow and/or any access violation [6]. For example, an attacker may steal user credentials using social engineering and access sensitive information or copy it to untrusted storage. In this scenario, security systems such as firewalls, IDS [7] or Security Information and Event Management(SIEM), and Data Leak Prevention System(DLP) [8] cannot detect effectively. Relying on analysts to investigate attacks is costly and time-consuming, as they have to deal with millions of logs.

User Behavior Analytics, which has been used in online social media analysis [9] and improving web search ranking [10], is emerging in security area. User behavior analytics is a cyber security process about detection of insider threats, targeted attacks, and financial fraud. They look at patterns of human behavior, and then apply algorithms and statistical analysis to detect meaningful anomalies from those patterns [11]. UBA collects various types of data such as organization structure, user roles and job responsibilities, user activity trace and geographical location. The analysis algorithms consider factors including contextual information, continuous activities, duration of sessions, and peer group activity to compare anomaly behavior. UBA determines the baseline of normal behavior of individual user or peer group according to history data. The deviation of ongoing user activities compared with past normal behavior is significant if the user acts abnormally [12].

This paper introduces a User Behavior Analytics Platform built to detect potential insider threats. Specifically, the platform can 1) collect and preprocess logs from systems and applications; 2) extract each user's activity records from logs; 3) aggregate activity records and generate feature vector for each user; 4) detect anomalous user access. Besides, an ensemble by multiple unsupervised anomaly detection algorithms is proposed and shows great performance in detecting user's anomalous access and operation within enterprise.

This paper is organized as follows. Section 2 introduces related work. User behavior analytics architecture and platform which contains four components is presented in Section 3. Section 4 introduces experiment scenario, data characteristics and feature selection. In Section 5, anomaly detection algo-

rithms for user behavior analytics are demonstrated. Section 6 gives dataset, experiment and results. Besides, discussion and comparison of algorithms are demonstrated. Finally, section 7 concludes the paper and provides future work.

## II. RELATED WORK

Anomaly detection is an important problem that has been researched within diverse research areas and application domains including information security [13]. Research of applying anomaly detection is popular in intrusion detection [7], fraud detection [14] [15], medical and public health anomaly detection and industrial damage detection. Many anomaly detection techniques have been specifically developed for certain application domains, while others are more generic.

Anomaly detection techniques being applied to user behavior analytics is increasingly popular. Veeramachaneni K et al. [16] put forward  $AI^2$  that combines analyst intelligence with an ensemble of three outlier detection methods to detect account takeover, new account fraud and service abuse.

Madhu Shashanka et al. [17] presented the User and Entity Behavior Analytics(UEBA) module of the Niara Security Analytics Platform which uses a SVD-based algorithm to detect anomalies in user accessing server within an enterprise. Both users historical baseline and peer baseline are applied with same algorithm.

Sapegin A et al. [18] proposed a poisson-based two-step algorithm to identify anomaly user access to workstation within Windows domain. However, the dataset is from simulation scenario and of limited features. The algorithms are not persuasive enough and of limited extensibility.

Wei Ma et al. [19] defined a user behavior pattern and proposed a knowledge-driven user pattern discovery approach which can extract users behavior patterns from audit logs from distributed medical imaging systems. The work is emphasized on extracting user behavior patterns and there is a long way to go before administrator directly use it.

Li at al. [20] proposed a kind of security audit technology based on one-class support vector machine detect the abnormal behavior of database operations.

## III. USER BEHAVIOR ANALYTICS ARCHITECTURE AND PLATFORM

In this section, an architecture of user behavior analytics is presented. Based on that, the implementation of our UBA platform is described.

Relying on analysts to investigate attacks is costly and time-consuming, as they have to deal with millions of logs and alerts. Our UBA platform collects logs about user-related events and user session activity in real-time or near real-time, and compares each and every action to the corresponding baseline of users to spot anomalies in their behavior. Based on detection results, a risk label or score that reveals human risk will be assigned to every user, which is helpful and meaningful for security analysts especially when analysts investigate or monitor employees for suspicious behaviors or attacks. Fig.1 provides the architecture composed of four components. Each of them is described in the following.

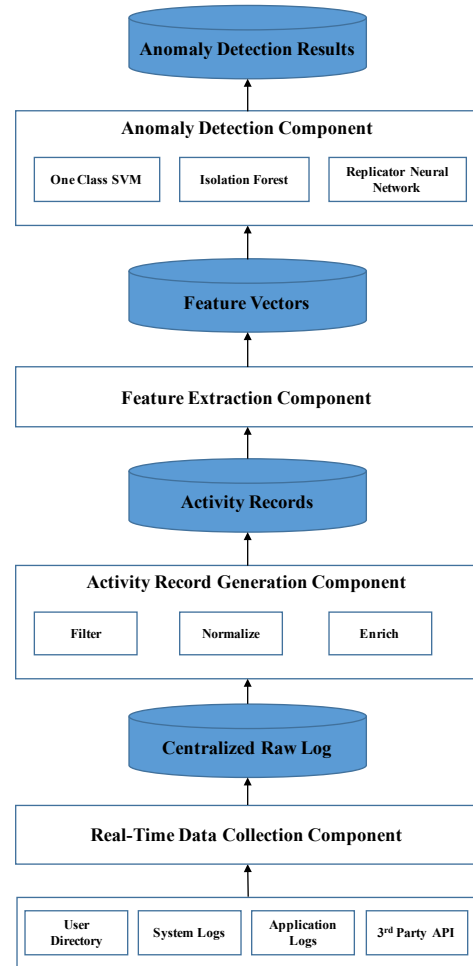


Fig. 1. Architecture of User Behavior Analytics Platform

### A. Data Collection Component

Data collection component stores raw logs generated by systems and applications for further extraction and analysis. The collected raw logs are stored in ElasticSearch [21], which is a distributed, JSON-based search and analytics engine designed for horizontal scalability, maximum reliability, and easy management. Logs of users accessing ftp server within enterprise and operations such as downloading, uploading, deleting files or directories are collected. The user information can be gathered from activity directory of the enterprise.

Data source our platform can process includes:

- 1) system logs,
- 2) application logs such as web access logs and DLP logs,
- 3) user directory logs, etc.

### B. Activity Record Generation Component

Centralized raw logs are of respective unique formats from which feature cannot be extracted directly. For example, apache server logs and windows security logs consist of different items. Due to the lack of normalized formats, activity record generation component

- 1) builds a general schema for activity records,

- 2) generates regular expressions as filters for each type of log to extract useful information,
- 3) fills the schema with extracted information.

Then the activity records with user information are generated, which will be processed in the following Feature Extraction Component.

### C. Feature Extraction Component

After generating normalized activity records with user information attached, we compute user behavioral features over an interval of time such as 24 hours. For performance consideration, the strategy from [16] is applied. Each hour we retrieve activity records within last hour and compute the features labeled with last hour. In midnight, we only need to retrieve the 23 feature sets and activity records within last hour rather than activity records within last 24 hours.

### D. Anomaly Detection Component

With features extracted for each user, anomaly detection component detects anomalous users on a daily basis. The component is designed loosely coupled, flexible and independent from other components. The details of algorithms are demonstrated in V.

## IV. DATA CHARACTERISTICS

In this section, a scenario within a typical software company is introduced. The behavior of employees accessing ftp files and data within work groups are monitored and audit logs are generated and collected by UBA platform presented before. Then dataset and feature selection is introduced.

### A. Experiment Scenario

Consider a file server within an enterprise, authorized employees can access the server for files and data with different authorization, which normally is configured by the administrator. For example, one can read, write, upload, download or delete files or directories. As documents and data are important information, accessing and operations need to be monitored for possible actions from compromised accounts or rogue users. UBA platform monitors the access patterns and operation patterns of each user while accessing the server and files.

### B. Dataset

The ftp server logs are collected by the data collection component presented before. In total 8 kinds of logs are collected and each corresponds to 1 or more types of events. For example, Download Log only represents Download Event and the log carries information including timestamp, user name, SUCCESS/FAIL flag and client IP. An UPLOAD LOG may represent an upload file/directory event, create file event or remote copy event and cannot be distinguished by content as different events share same format. Table I shows the representation map between logs and events.

We collected operation logs within a software company for 3 months and selected four employees with explicit different behavior patterns. Activity records generated were checked

TABLE I  
LOGS AND CORRESPONDING EVENTS

LOG	EVENT
CONNECT LOG	CONNECT Event
LOGIN LOG	Login Event
DOWNLOAD LOG	Download Event
UPLOAD LOG	Upload Event Create File Event Remote Copy Event
DELETE LOG	Delete File Event Delete Directory Event
MKDIR LOG	Make Directory Event
RMDIR LOG	Remove Directory Event
RENAME LOG	Rename Event Remotely Move Files Event

and all can be considered as normal behaviors so we simulate several abnormal operations for each user as testing dataset. Based on investigation in the enterprise, abnormal behaviors mainly include four categories shown in Table II.

TABLE II  
CATEGORIES OF ABNORMAL BEHAVIOR

anomaly ID	Description
anomaly 1	multiple login attempts and failures
anomaly 2	anomalous downloads operations
anomaly 3	anomalous delete operations
anomaly 4	operations at non-working hours

### C. Feature Selection

With activity records generated by activity record generation component, feature extraction component produces a feature vector for each user daily, which characterize the pattern of users' access to ftp server and operations. The features are shown in Table III. The features for a user daily is denoted

TABLE III  
THE LIST OF FEATURES. 21 FEATURES ARE EXTRACTED AND USED IN TOTAL.

Feature ID	Description
1	number of total connections of the day
2	timestamp of first login attempt of the day
3	timestamp of last login attempt of the day
4	number of login success of the day
5	number of login fail of the day
6	total download bytes
7	number of download success of the day
8	number of download fail of the day
9	largest download bytes of the day
10	total upload bytes
11	number of upload success of the day
12	number of upload fail of the day
13	largest upload bytes of the day
14	number of delete success of the day
15	number of delete fail of the day
16	number of mkdir success of the day
17	number of mkdir fail of the day
18	number of rmdir success of the day
19	number of rmdir fail of the day
20	number of rename success of the day
21	number of rename fail of the day

by 21-dimension vector  $x = (x_1, x_2, \dots, x_{21})$ .



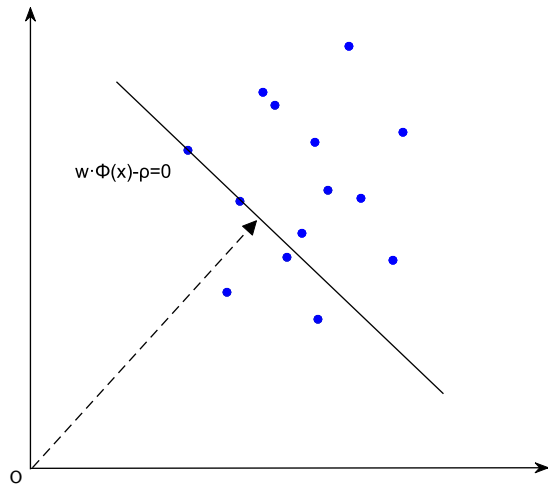


Fig. 2. diagram of OCSVM hyperplane

## V. ALGORITHM

UBA platform practically is fed with data without label, which motivates us to use unsupervised anomaly detection techniques. In practice, it's unknown whether training set contains abnormal data points and the proportion, different algorithms are of better performance under different conditions. For example, when training set contains normal instances only, Replicator Neural Network and OCSVM work better, but Isolation Forest might suffer a small reduction. As a result, an ensemble of three unsupervised anomaly detection algorithms is used to improve the robustness and performance.

### A. One Class SVM

OCSVM, proposed by Scholkopf [22], has been applied to anomaly detection. As Fig.2 shows, the OCSVM algorithm maps input data into a high dimensional feature space via a kernel and iteratively finds the maximal margin hyperplane which best separates the training data from the origin.

$$\begin{aligned} \min_{w, \zeta_i, \rho} \quad & \frac{1}{2} \|w\|^2 + \frac{1}{\nu n} \sum_{i=1}^n \zeta_i - \rho \\ \text{s.t.} \quad & (w^T \phi(x_i)) > \rho - \zeta_i, i = 1, \dots, n \\ & \zeta_i > 0, i = 1, \dots, n \end{aligned} \quad (1)$$

The decision function presented is  $f(x) = \text{sgn}(w^T \phi(x) - \rho)$ . After solving the dual problem below:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \sum_{ij} \alpha_i \alpha_j k(x_i, x_j) \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq \frac{1}{\nu l}, i = 1, \dots, n \\ & \sum_{i=1}^n \alpha_i = 1 \end{aligned} \quad (2)$$

The decision function is given by :

$$f(x) = \text{sgn}\left(\sum_{i=1}^n (\alpha_i K(x_i, x) - \rho)\right). \quad (3)$$

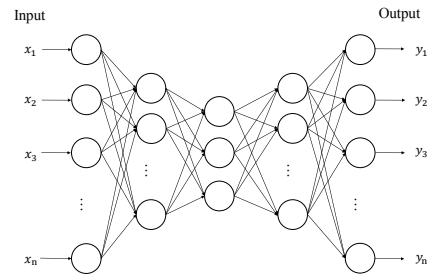


Fig. 3. Replicator Neural Network with three hidden layers

### B. Replicator Neural Network

Replicator Neural Network is an artificial feed-forward multi-layer neural network with an output layer having the same number of nodes as the input layer. The purpose of Replicator Neural Network is to produce the output data which as is similar as the input data. Fig.4 presents the structure of the fully connected RNN with three hidden layers.

Replicator Neural Network is effective in anomaly detection as an unsupervised machine learning algorithm because anomalies are few and there exist some common patterns in normal data. By the trained RNN, the common patterns representing bulk of the data can be well reproduced, while anomalies will have a much higher reconstruction error. The reconstruction error for a  $d$ -dimensional instance  $x = (x_1, x_2, \dots, x_d)$  is computed as follow:

$$e = \sum_{i=1}^d (x_i - y_i)^2 \quad (4)$$

in which  $d$  is the dimension of input vector  $x$  and  $y = (y_1, y_2, \dots, y_d)$  is the reconstructed output.

### C. Isolation Forest

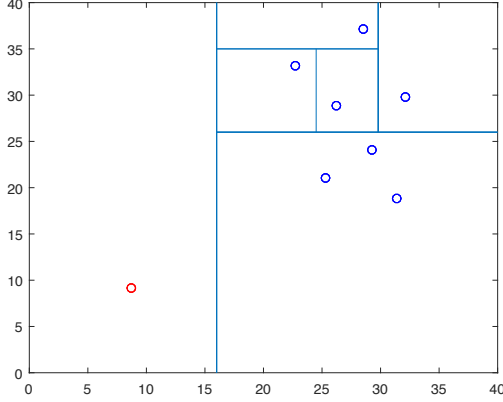
Since anomalies are few and different and therefore they are more susceptible to isolation. Based on the concept of isolation, Isolation Forest [23] builds a set of  $iTrees$  for a given data set, then anomalies are those instances which have short average path lengths on the  $iTrees$ . For example, in Fig.4, the red outlier (8.7, 9.2) is isolated at first split, while normal points marked as blue need more than 3 splits in the isolation tree.

To be specific, for a given dataset,  $iTrees$  are constructed by recursively partitioning the given training set until instances are isolated or a specific tree height is reached. There are only two variables in this method:

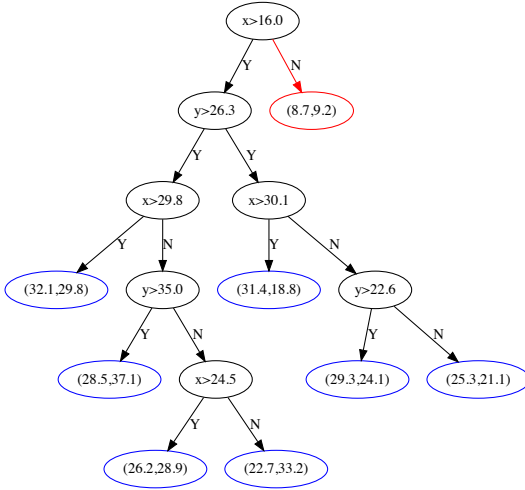
- 1) the number of trees to build  $t$ , and
- 2) the sub-sampling size  $\psi$

Path length  $h(x)$  of a point  $x$  is measured by the number of edges  $x$  traverses an  $iTree$  from the root node until the traversal is terminated at an external node. The anomaly score  $s$  of an instance  $x$  is defined as:

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}}, \quad (5)$$



(a) Training Data



(b) Generated Isolation Tree

Fig. 4. Isolation Tree Generated with data set

where  $E(h(x))$  is the average of  $h(x)$  from the trained collection of isolation trees.

#### D. Ensemble and Strictly Filtering

We combine the predictions of three algorithms introduced before and apply the strictly filtering strategy to predict whether a user is anomalous or not. OCSVM directly produces label  $y \in \{0, 1\}$ . Output of RNN is reconstruction error  $err \in \mathbb{R}$  while  $iForest$  generates anomaly score  $s(x, n) = 2^{-\frac{E(h(x))}{c(n)}} \in (0, 1)$ . Labels are given by comparing output with corresponding threshold. Output 0 represents abnormal while 1 represents normal.

$$f_{OCSVM}(x; X) = \begin{cases} 0, \\ 1 \end{cases} \quad (6)$$

$$f_{iForest}(x; X) = \begin{cases} 0, & s(x, n) > \varepsilon_1 \\ 1, & s(x, n) \leq \varepsilon_1 \end{cases} \quad (7)$$

$$f_{RNN}(x; X) = \begin{cases} 0, & err(x) > \varepsilon_2 \\ 1, & err(x) \leq \varepsilon_2 \end{cases} \quad (8)$$

where “; X” indicates the model is trained with X as training set. As recall is an important measure metric in security, we apply strictly filtering strategy and regard data point as abnormal as long as any of the three algorithms outputs 0, shown by formula 9 and 10.

$$f(x; X) = f_{OCSVM}(x; X) + f_{iForest}(x; X) + f_{RNN}(x; X) \quad (9)$$

$$s(x; X) = \begin{cases} 0, & f(x; X) < 3 \\ 1, & f(x; X) = 3 \end{cases} \quad (10)$$

Given the  $k_{th}$  user’s historical behavior  $X_k = [x_k^1, x_k^2, \dots, x_k^i, \dots, x_k^m]$ , in which  $i \in \{1, 2, \dots, m\}$  denotes the index of days and  $x_k^i = (x_k^{i,1}, x_k^{i,2}, \dots, x_k^{i,21})^T$  is the feature vector of  $k_{th}$  user in  $i_{th}$  day.

For  $k_{th}$  user with feature vector to be detected and denoted by  $\hat{x}_k = (\hat{x}_k^1, \hat{x}_k^2, \dots, \hat{x}_k^{21})^T$ , the prediction is given by:

$$score_k = s(\hat{x}_k, X_k) \quad (11)$$

## VI. EXPERIMENT AND RESULTS

### A. Experiment Setup

Data preserved on ftp server is not enough so we perform a simulation after fitting the data collected with a polynomial distribution. Besides, a small proportion 2.06% and 4.03% of anomalous user behaviors is mixed into training set to find out performance when training sets mixed with different purities. Hence three training sets are used as Table IV shows and the data sets are composed of five categories.

In OCSVM, we exploit LIBSVM [24] and RBF kernel  $K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$ ,  $\gamma > 0$ , where  $\gamma = 0.01$  and  $\nu = 0.05$  is selected as parameters.

In  $iTree$ , number of trees  $t = 200$  and the sub-sampling size  $\psi = 200$ . The threshold  $\varepsilon_1 = 0.45$ .

A RNN with 3 hidden layers is applied and the number of neurons in each layer is [20, 8, 4, 8, 20]. Activation function  $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$  is selected. The threshold selected is  $\varepsilon_2 = 1.20$ . Based on stochastic gradient descent, the training contains 60,000 train epochs and batch-size is set 20.

### B. Results and Discussion

The detection rate of single algorithm in different category of testing data is shown in Table V. Assuming abnormal data points is our focus and marked as positive, the overall accuracy, precision and recall is shown in Table VI.

With anomaly-free training set, OCSVM has best performance with recall=100% and accuracy=96.72%. All of the anomalies can be detected. With more anomaly points in the training set, performance of OCSVM gets worse. With 4.03% anomaly points, the recall is 75.86% and accuracy is 81.39%.

TABLE IV  
COMPOSITION OF TRAINING SETS AND TESTING SET. TRAINING SETS WITH 0%, 2.06% AND 4.03% OF ANOMALIES MIXED INTO ARE USED.

dataset	normal	anomaly 1	anomaly 2	anomaly 3	anomaly 4	total	anomalies proportion
training set 1	1000	0	0	0	0	1000	0.00%
training set 2	1000	6	6	3	6	1021	2.06%
training set 3	1000	12	12	6	12	1042	4.03%
testing set	100	45	47	32	50	274	63.50%

TABLE V  
DETECTION RATE OF OCSVM, RNN AND *iForest* WITH DIFFERENT TRAINING SETS

category	training set 1(0.00%)			training set 2(2.06%)			training set 3(4.03%)		
	OCSVM	RNN	<i>iForest</i>	OCSVM	RNN	<i>iForest</i>	OCSVM	RNN	<i>iForest</i>
normal	91.00%	92.00%	51.00%	91.00%	88.00%	87.00%	91.00%	92.00%	87.00%
anomaly 1	100.00%	100.00%	100.00%	93.33%	100.00%	91.11%	82.22%	62.22%	77.78%
anomaly 2	100.00%	100.00%	100.00%	79.59%	97.87%	68.09%	89.36%	55.32%	65.96%
anomaly 3	100.00%	100.00%	100.00%	100.00%	100.00%	68.75%	100.00%	81.25%	100.00%
anomaly 4	100.00%	92.00%	78.00%	54.00%	90.00%	38.00%	42.00%	92.00%	36.00%

TABLE VI  
ACCURACY, PRECISION AND RECALL OF OCSVM, RNN AND *iForest* WITH DIFFERENT TRAINING SETS

category	training set 1(0.00%)			training set 2(2.06%)			training set 3(4.03%)		
	accuracy	precision	recall	accuracy	precision	recall	accuracy	precision	recall
OCSVM	96.72%	95.08%	100.00%	83.58%	93.88%	79.31%	81.39%	94.29%	75.86%
RNN	95.62%	95.51%	97.70%	93.43%	93.33%	96.55%	79.56%	94.03%	72.41%
<i>iForest</i>	78.10%	76.89%	93.68%	73.58%	89.76%	65.52%	74.09%	89.92%	66.67%
ensemble	<b>91.60%</b>	<b>88.32%</b>	<b>100.00%</b>	<b>90.88%</b>	<b>89.84%</b>	<b>96.55%</b>	<b>91.24%</b>	<b>93.10%</b>	<b>93.10%</b>

RNN has the similar performance and trend to OCSVM. Fig.5 shows the mean reconstruction error  $e = \sum_{i=1}^d (x_i - y_i)^2$  of 5 categories of test data during RNN training with 2.06% anomalies in training set. The training process converged and abnormal data has higher reconstruction error which makes separating possible. The mean reconstruction error of normal data is 0.539, much lower than abnormal data in testing dataset (4.947, 3.887, 8.627, 2.409). However, the time cost of training is much higher than other algorithms.

Isolation Forest has the worst performance of the three algorithms as Table V and Table VI shows. With training set 2, the anomaly score of test data from Isolation Forest is presented in Fig.6. The normal data has a lower anomaly score 0.390 than anomaly data (0.490,0.487,0.475,0.442). However, scores of some data are pretty close as Fig.6 shows, especially the data of operations at non-working hour. In each category of anomaly data we simulated, data is anomaly in only few dimensions. At training stage, attribute and split point is randomly selected. Statistically it's hard to split data with anomalous attributes before the tree goes deep. However, if the training set has more complicated and real anomalies, *iForest* can be of better performance. Besides, the threshold  $\varepsilon_1$  can be adjusted flexibly, which is a valuable characteristic. In addition, *iForest* didn't suffer an obvious reduction with more anomalies mixed in the training set.

Table VI shows that the ensemble and strictly filtering strategy improve robustness and performance especially recall.

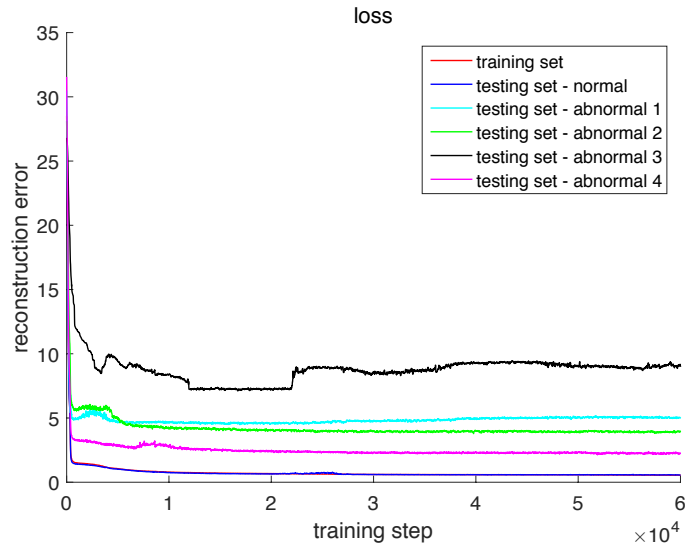


Fig. 5. Mean reconstruction error during training with Replicator Neural Network.

When anomalies in training set increase, algorithms alone are less reliable. With 2.06% anomalies in training set, the ensemble gives recall=96.55% and accuracy=90.88%. With 4.03% anomalies in the training set, RNN has recall=72.41% and accuracy=79.56%, while the ensemble has great performance recall=93.10% and accuracy=91.24%. It can be a good and

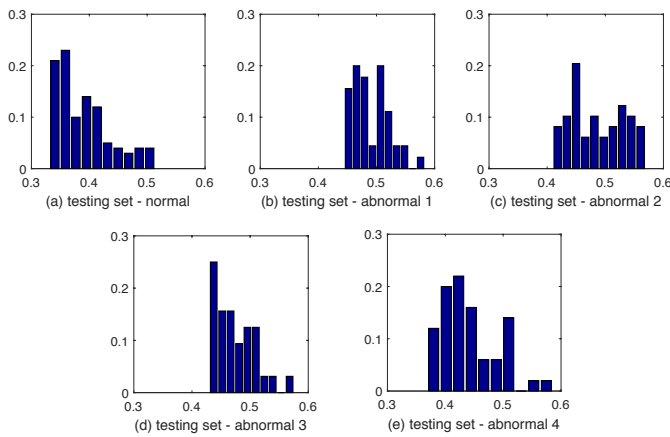


Fig. 6. Anomaly Score of Isolation Forest for test data of different categories

optional strategy especially when security analysts focus on recall.

## VII. CONCLUSION AND FUTURE WORK

This paper presents an overview of UBA architecture and platform for detecting anomalous user behaviors within enterprise. The platform, composed of four components working independently, is suitable for running on distributed platforms. The anomaly detection component contains an ensemble of OCSVM, RNN and Isolation Forest. Strictly filtering strategy is applied and can improve the performance and robustness no matter whether there exist anomalies in the training set.

The sequences of events contain valuable information about users and we will focus on anomaly detection of sequence data. Besides, the peer group analysis, which may play an important role in practice, can be introduced into the UBA platform in the future.

## REFERENCES

- [1] New haystack technology survey shows most organizations ill-prepared for insider threats. <https://haystack.com/blog/2017/03/29/new-haystack-technology-survey-shows-most-organizations-ill-prepared-for-insider-threats/> (accessed December, 2017).
- [2] A. P. Moore, K. A. Kennedy, and T. J. Dover, "Introduction to the special issue on insider threat modeling and simulation," *Computational and Mathematical Organization Theory*, vol. 22, no. 3, pp. 261–272, 2016.
- [3] D. M. Cappelli, A. P. Moore, and R. F. Trzeciak, *The CERT guide to insider threats: how to prevent, detect, and respond to information technology crimes (Theft, Sabotage, Fraud)*. Addison-Wesley, 2012.
- [4] U. A. Office. Fannie mae corporate intruder sentenced to over three years in prison for attempting to wipe out fannie mae financial data. <https://archives.fbi.gov/archives/baltimore/press-releases/2010/ba121710.html/> (accessed December, 2017).
- [5] F. I. P. Bureau, "Unintentional insider threats: A foundational study," 2013.
- [6] M. Uma and G. Padmavathi, "A survey on various cyber attacks and their classification." *IJ Network Security*, vol. 15, no. 5, pp. 390–396, 2013.
- [7] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection." *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.
- [8] A. Shabtai, Y. Elovici, and L. Rokach, *A survey of data leakage detection and prevention solutions*. Springer Science & Business Media, 2012.
- [9] Y. Amichai-Hamburger and G. Vinitzky, "Social network use and personality," *Computers in human behavior*, vol. 26, no. 6, pp. 1289–1295, 2010.
- [10] E. Agichtein, E. Brill, and S. Dumais, "Improving web search ranking by incorporating user behavior information," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2006, pp. 19–26.
- [11] T. Bussa, A. Litan, and T. Phillips, "Market guide for user and entity behavior analytics," URL: <https://www.gartner.com/doc/3538217/market-guide-user-entity-behavior> ( 29.07. 2017), 2016.
- [12] W. Ma, "User behavior pattern based security provisioning for distributed systems." Ph.D. dissertation, 2016.
- [13] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.
- [14] D. J. Weston, D. J. Hand, N. M. Adams, C. Whitrow, and P. Juszczak, "Plastic card fraud detection using peer group analysis," *Advances in Data Analysis and Classification*, vol. 2, no. 1, pp. 45–62, 2008.
- [15] M. Ahmed, A. N. Mahmood, and M. R. Islam, "A survey of anomaly detection techniques in financial domain," *Future Generation Computer Systems*, vol. 55, pp. 278–288, 2016.
- [16] K. Veeramachaneni, I. Arnaldo, V. Korrapati, C. Bassias, and K. Li, "Ai<sup>2</sup>: training a big data machine to defend," in *Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS), 2016 IEEE 2nd International Conference on*. IEEE, 2016, pp. 49–54.
- [17] M. Shashanka, M.-Y. Shen, and J. Wang, "User and entity behavior analytics for enterprise security," in *Big Data (Big Data), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1867–1874.
- [18] A. Sapegin, A. Amirkhanyan, M. Gawron, F. Cheng, and C. Meinel, "Poisson-based anomaly detection for identifying malicious user behaviour," in *International Conference on Mobile, Secure and Programmable Networking*. Springer, 2015, pp. 134–150.
- [19] W. Ma, K. Sartipi, and D. Bender, "Knowledge-driven user behavior pattern discovery for system security enhancement," *International Journal of Software Engineering and Knowledge Engineering*, vol. 26, no. 03, pp. 379–404, 2016.
- [20] Y. Li, T. Zhang, Y. Y. Ma, and C. Zhou, "Anomaly detection of user behavior for database security audit based on ocsvm," in *Information Science and Control Engineering (ICISCE), 2016 3rd International Conference on*. IEEE, 2016, pp. 214–219.
- [21] elasticsearch.io. <https://www.elastic.co/products/elasticsearch> (accessed December, 2017).
- [22] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection," in *Advances in neural information processing systems*, 2000, pp. 582–588.
- [23] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE, 2008, pp. 413–422.
- [24] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.

# Modeling and Verification of IEEE 802.11i Security Protocol for Internet of Things

Yuteng Lu and Meng Sun

LMAM & Department of Informatics, School of Mathematical Sciences,  
Peking University, Beijing, China  
{luyuteng,sunm}@pku.edu.cn

**Abstract**—IEEE 802.11i is the IEEE standard that provides enhanced MAC security and has been widely used in wireless networks and Internet of Things. It improves IEEE 802.11(1999) by providing a Robust Security Network (RSN) with two new protocols: the 4-way handshake and the Group-key handshake. These protocols utilize the authentication services and port access control described in IEEE 802.1X to establish and change the appropriate cryptographic keys. In this paper, we carry out a formal modeling and verification approach based on timed automata for IEEE 802.11i protocol, using the UPPAAL model checker, to check correctness of the changes in IEEE 802.11i protocol and provide better security.

**Keywords:** IEEE802.11i Protocol, Model Checking, 4-Way Handshake, Group-Way Handshake, UPPAAL

## I. INTRODUCTION

The Internet-of-Things (IoT) [5], [17] is an evolving paradigm that offers a family of sophisticated computing services and physical instruments which cooperate with each other over the Internet. It has gained increased attention in the past decade due to rapid development in computing and storage technologies and easy access to the Internet, and brings promising opportunities and challenges. IoT has played a key role in the next generation of information, network, and communication systems. With the rapid development of IoT, security of communication, being of paramount importance, has attracted more and more attention. Since the intruder can attack a network remotely, the industry has been trying to introduce various security protocols to improve the security of the wireless network and IoT.

IEEE 802.11 is a set of standards defined by IEEE for wireless network communication, such as 802.11e for QoS enhancement of 802.11, 802.11k for radio resource management, and 802.11n for high throughput enhancement, and so on. In the actual usage, IEEE802.11 exposed a lot of security issues, so IEEE also developed a set of IEEE802.11 amendment to make up for its fragile security encryption, which is the IEEE802.11i protocol being studied in this paper.

Industrial practice has shown that in the design of a complex protocol, more time and effort are usually spent on verification of the correctness of the protocol, rather than in the formulation of the protocol itself. Formal verification techniques, especially model checking [3], aim to establish correctness with mathematical rigor and offer a large potential to obtain

an early integration of verification in the design process, to make verification activities more effective, and to reduce the verification time. For these reasons, model checking has been recognized as an important method to guarantee the correctness of protocols formally and avoid further odious problems caused by errors in early stage.

A large body of literature for analysis and verification of IEEE 802.11i protocol already exist. For example, the High-level Petri Net (HPN) model was adopted in [10] to specify the protocol framework of the 4-way handshake protocol, and model checking techniques are used to carry out the security verification on the HPN models. In [11] a logic based approach is taken to verify several properties of some typical methods in extensible authentication protocol, which are major solutions in IEEE 802.11i implementation. The behavior tree models for IEEE 802.11i RSN were developed and verified by using the SAL model checker in [15]. The IEEE 802.11i amendment is analyzed and a number of potential threats are identified in [18]. The 4-way handshake authentication WPA-PSK protocol in IEEE802.11i was verified in [13] using the CasperFDR model checker. On the other hand, the 4-way handshake phase in IEEE 802.11i Standard has been analyzed in [1] using theorem prover Isabelle to identify a new Denial-of-Service (DoS) attack. A key refreshing technique to reduce 4-way handshake latency in 802.11i based networks was proposed in [14] which provides per frame key freshness and generates a new refreshed secret key for encryption of each frame. However, most of these works only focus on part of IEEE 802.11i protocol and ignore the other parts.

In this paper, we investigate the usage of UPPAAL [16] to analyze and verify different protocols in the IEEE 802.11i standard. UPPAAL is a toolbox for verification of real-time systems that can be modeled as networks of timed automata (TA) [2] extended with integer variables, structured data types, and channel synchronization. UPPAAL can be used to automatically check whether a given property is satisfied by a system. The query language of UPPAAL used to specify the properties to be checked is a subset of CTL. It uses a client-server architecture, splitting the tool into a graphical user interface and a model checking engine. The UPPAAL model checker is based on the theory of timed automata and its modeling language offers additional features such as bounded integer variables and urgency. In the past decades, UPPAAL has been applied successfully in various industrial case studies

ranging from communication protocols to multimedia applications, such as Bounded Retransmission Protocol [4], Bang & Olufsen audio / video protocol [6] and Philips audio protocol [9].

In an actual network communication, after the user login the network, whether its landing time is more than the scheduled time and in each time period whether it is still online are the problems that the authentication side must concern about. So timing parameters play an essential role in the IEEE 802.11i protocol, especially for communications in wireless networks and IoT. In addition, our work also examines whether the protocol will be deadlocked, which is a vital security attribute in communication. For wireless network and IoT, it is also an important issue that the interactive port can eventually be opened normally.

The rest of this paper is organized as follows: The IEEE 802.11i standard is briefly described in Section II. Section III presents the analysis of the IEEE 802.1X protocol which is used in the authentication process of IEEE 802.11i. Then the verification of 4-Way Handshake for key management and distribution and the Group Key Handshake in UPPAAL is provided in Section IV. Section V concludes the paper and discusses possible future work.

## II. THE IEEE 802.11I PROTOCOL

IEEE802.11i is an IEEE standard designed to provide enhanced MAC security in wireless networks, which enhances the IEEE 802.11 standard in terms of security by providing a Robust Security Network (RSN) [7]. The protocol consists of several parts, including a 802.1X authentication phase using TLS over EAP, the 4-way handshake protocol to establish a fresh session key, and an optional Group Key Handshake protocol for group communications. This series of protocols together define a RSN.

The 4-way handshake protocol and the Group Key Handshake protocol utilize the authentication services and port access control described in IEEE 802.1X [8]. Authentication services involve three parts: the applicant, the certifier, and the authentication server. The applicant is a client device that needs to be connected to a LAN / WAN, and can also refer to software that runs on the client and provides credentials to the certifier. The verifier behaves like a guard of a protected network. Applicants (such as client devices) do not allow authenticated access to the protected side of the network until the identity of the applicant is verified and authorized. And the authentication server is typically a host running RADIUS and EAP-enabled protocols. The EAP data is first encapsulated in an EAPOL frame and transmitted between the supplicant and the authenticator, and then encapsulated in RADIUS or Diameter, transmitted between the verifier and the authentication server.

The initial authentication process is carried out either using a pre-shared key (PSK), or following an EAP exchange through 802.1X. If a 802.1X EAP exchange was carried out, the PMK is derived from the EAP parameters provided by the authentication server. The supplicant and the authenticator

use this handshake to confirm the existence of the PMK, verify the selection of the cipher suite, and derive a fresh Pairwise Transient Key(PTK), based on the shared PMK, the nonces and MAC addresses. In case of multicast application, the authenticator will generate a fresh Group Temporary Key(GTK) and may distribute a GTK to supplicants. During the Group Key Handshake, the same PMK may be used repeatedly for multiple times.

## III. VERIFYING THE IEEE802.1X PROTOCOL

In this section we first introduce the working principle of the IEEE 802.1X protocol and verify some important properties. The details of the message forwarding in the protocol and its specific implementation process are described in Figure 1.

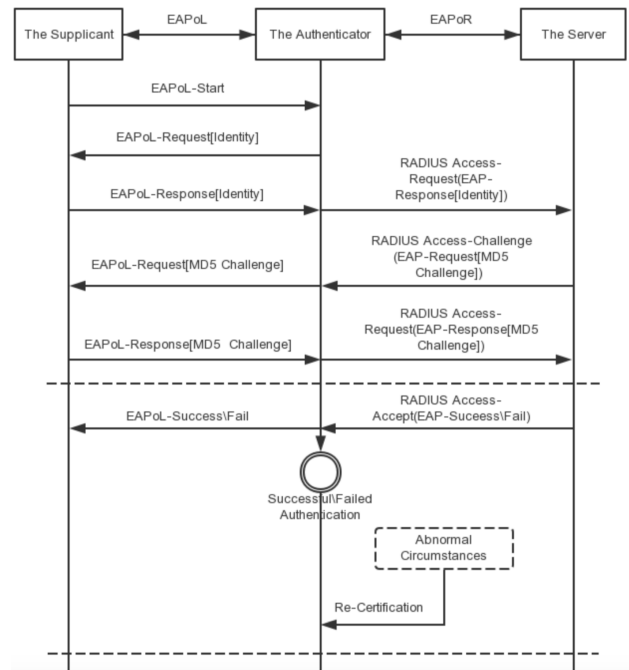


Figure 1. Message Forwarding in the IEEE 802.1X Protocol

When a user requests to access the network, the supplicant receives the user's request login information and sends the start message "EAPoL-Start" to the authenticator to trigger the authentication process. Once receiving the "EAPoL-Start" packet from the supplicant, the authenticator sends a request to the supplicant asking for the user identity. After receiving the "EAPoL-Request[Identity]" packet from the authenticator, the supplicant sends the user identity to the authenticator. Then, the authenticator passes the user identity through the RADIUS Access-Request "EAPoL-Response[Identity]" to the server. After receiving the packet, the server sends an Access-Challenge packet "EAPoL-Request[MD5 Challenge]" to the authenticator to request a password, and then the authenticator sends the message "EAPoL-Request[MD5 Challenge]" to the supplicant. After the supplicant receives the packet, it sends the password to the authenticator through the message "EAPoL-Response[MD5 Challenge]".

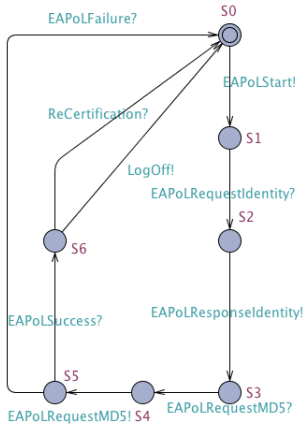


Figure 2. UPPAAL Model for Supplicant in IEEE 802.1x Protocol

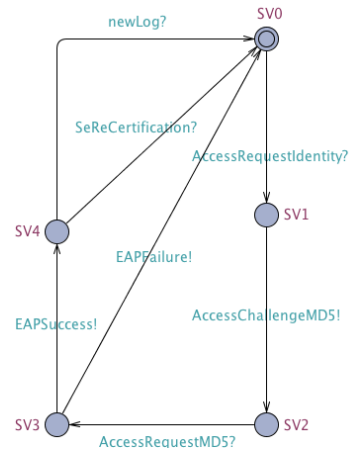


Figure 4. UPPAAL Model for Server in IEEE 802.1x Protocol

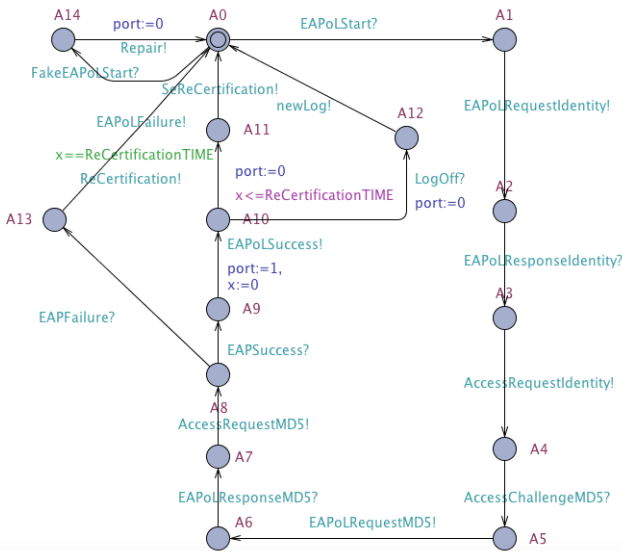


Figure 3. UPPAAL Model for Authenticator in IEEE 802.1x Protocol

After receiving the packet, the authenticator sends the message to the server for verification. If the authentication succeeds, the server sends the authenticator success message “EAP-Success” to the device. The authenticator sends the message “EAPoL-Success” to the supplicant to notify the user that the authentication is successful. Otherwise, the server sends an authenticator failure packet, and the authenticator informs the supplicant of the failure of the authentication message so that the user knows that the authentication fails. The user can communicate normally once the authentication succeeds. However, although the authentication is successful, the legitimate users may be disconnected due to abnormal circumstances, there may be the situation that illegal users replace legitimate users, so the re-certification mechanism was introduced. That is, after successful authentication for the first time, the server authenticates the user to determine whether the user is online and whether it is legitimate over a period of time.

The UPPAAL templates of the automata for the supplicant, the authenticator and the server in the IEEE 802.1x model are provided in Figure 2, 3 and 4. They run exactly as the protocol orders them. The main work of the IEEE802.1x protocol is to carry out the exchange of information, for which we have defined a number of channels. For example, the supplicant model moves from state  $S_0$  to state  $S_1$ , and sends the authentication start message “EAPoLStart”, synchronizing with the authenticator by output action EAPoLStart!, for which we define the channel EAPoLStart. As a receiver terminal apparatus, the authenticator moves from state  $A_0$  to state  $A_1$ , and synchronize with the supplicant by the corresponding co-action EAPoLStart?. The supplicant, the authenticator and the server interact with each other and their concurrent composition leads to the protocol.

After the supplicant receives the “EAPoL-Request [MD5 Challenge]” packet, it sends the password to the authenticator through the corresponding “EAPoL-Response [MD5 Challenge]” and the authenticator sends the message to the server for verification. Once the authentication is successful, we need to consider re-certification issues. Our model introduces the time variable  $x$  to record the elapsed time after the first authentication succeeds and the ReCertificationTIME records re-authentication interval. After the authentication is successful, the port is opened and the recording time starts, correspondingly,  $x$  is initialized to 0. The state transition of the authenticator sending the re-certification request is guarded by “ $x == \text{ReCertificationTIME}$ ”, so once the property “ $x$  will eventually be equal to ReCertificationTIME” is satisfied, it is guaranteed that the re-certification process can occur normally.

Since the IEEE 802.1x protocol is used to authenticate the identity of the user, the first property being required is that once the authentication is successful, the network port will open normally. Correspondingly, once the authentication fails, the network port will not open. In addition, the protocol should also ensure that the port can always be closed or opened, so as to avoid the situation that the port is always closed or

open. Since we also consider the issue of re-authentication, we should also ensure that the port can be opened or closed according to the situation after each re-authentication.

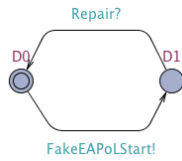


Figure 5. UPPAAL Model for DosAttack in IEEE 802.1x Protocol

Authentication phase of 802.11i suffers against DoS attacks mainly due to lack of authentication of packets [12]. In Figure 5 we define the DosAttack model to simulate a denial of service and define a new channel, FakeEAPoStart. After a denial of service attack, if there is no manual recovery, a deadlock will occur and the authenticator model will stay in the state  $A_{14}$ . Once the Repair channel is added to the model, the deadlock behavior will disappear.

A family of properties can be reformulated as CTL formula. For example:

- 1)  $E \langle \rangle \text{port} == 1$ , on behalf of the port can be opened.
- 2)  $E \langle \rangle \text{port} == 0$ , on behalf of the port can be closed.
- 3)  $A []$  not deadlock, on behalf of the IEEE 802.1x protocol will not be deadlocked.
- 4)  $\text{authenticator.A10} \rightarrow \text{port} == 0$ , representing that the port will be closed once the re-authentication process begins.
- 5)  $\text{authenticator.A8} \rightarrow \text{port} == 0$ , representing that once the authentication failed the port will not be opened.
- 6)  $E \langle \rangle x == \text{ReCertificationTIME}$ , representing that the re-certification process will occur.
- 7)  $\text{authenticator.A0} \rightarrow \text{port} == 0$ , representing that the port was closed initially.

The verification results given by UPPAAL show that all these properties have been proved to be satisfied. Thus, the port can be normally opened or closed, and there is no deadlock in authenticating the identity of the user. After the 802.1X authentication, a shared secret key is generated, called the Pairwise Master Key (PMK). The PSK is derived from a password that is put through the cryptographic hash function. In a pre-shared-key network, the PSK is actually the PMK. If a 802.1X EAP exchange is carried out and no deadlock can be guaranteed, the PMK is derived from the EAP parameters provided by the authentication server.

#### IV. 4-WAY HANDSHAKE AND GROUP KEY HANDSHAKE PROTOCOLS

In this section, we first consider security properties of the 4-way handshake protocol. During the handshake, the authenticator and the supplicant generate fresh nonces, then derive a fresh PTK based on the shared PMK, the nonces, and their MAC addresses, so that the authenticator and the supplicant can independently prove to each other that they know the

PMK, without ever disclosing the key. They authenticate the key material generated using keyed hashes. After this stage, the IEEE 802.1x ports are unblocked for data packets.

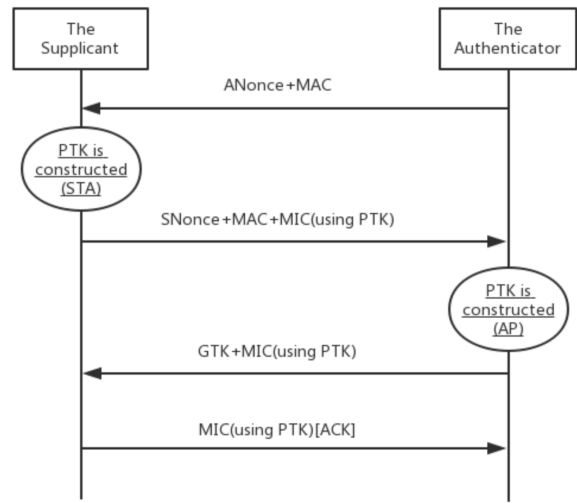


Figure 6. The Authentication Process in the 4-Way Handshake Protocol

Figure 6 describes the authentication process in the 4-way handshake protocol. In the form of first message the authenticator (AP) sends the random number “ANonce” and MAC address of itself. In response, the supplicant generates another random number “SNonce”, and sends it to the AP with the MAC address and message integrity code (MIC) using PTK. Then, a third message is sent by the AP after generating GTK and verifying MIC based on the PTK derived. The supplicant verifies MIC of this message and sends a MIC and install PTK at supplicant. After receiving the MIC message, AP also installs PTK and the 4-way handshake communication is completed. This is the normal way of handshake behavior. Once the supplicant does not receive the first message “ANonce+MAC” within the expected time interval, it will try the authentication again. On the other hand, the authenticator will timeout and retry the message if it does not receive the expected message “SNonce+MAC+MIC” within the configured time intervals.

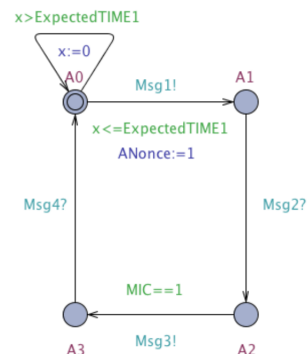


Figure 7. UPPAAL Model for Authenticator in the 4-Way Handshake Protocol



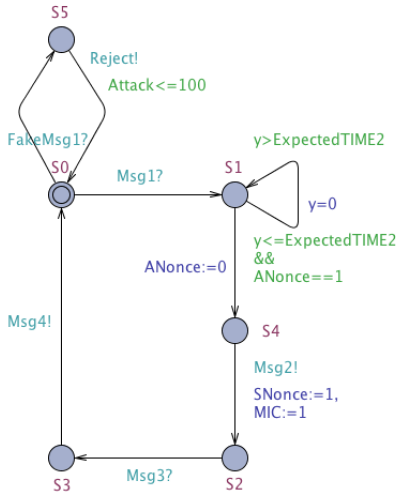


Figure 8. UPPAAL Model for Supplicant in the 4-Way Handshake Protocol

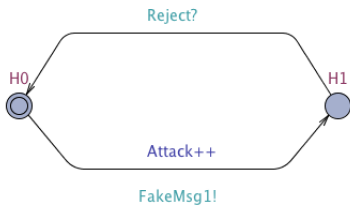


Figure 9. UPPAAL Model for Hacker in the 4-Way Handshake Protocol

The UPPAAL templates of authenticator and supplicant in the 4-way handshake protocol are described in Figure 7 and 8. The authenticator and the supplicant synchronize through the channel. For example,  $Msg1!$  represents authenticator sends the first message “ANonce+MAC” to supplicant, synchronizing with  $Msg1?$ , representing supplicant has received the message.

In Uppaal, we build a Hacker model and consider the number of attacks. Hacker multiple attacks will exhaust the memory and complete a denial of service attack. Here we have a  $FakeMsg1$  channel. Every time when the hacker sends  $FakeMsg1$  message, the Supplicant will be attacked. We use  $Attack$  to record the number of hacker attacks. Combined with the experience of the industry, we can notice that the agreement will collapse when the number of attacks exceeds a certain number limitation. In our model, once we set the number of attacks more than 100 times, Supplicant will not be able to reject the false information normally, thus it cannot be restored to the initial state.

A family of properties related to the 4-way handshake protocol have been verified in UPPAAL. For example, we have checked whether the handshake behavior will be deadlocked, and can eventually generate PTK. The corresponding properties are reformulated in CTL as follows:

- 1)  $A[]$  not deadlock, representing that the 4-way handshake behavior will not be deadlocked.

- 2)  $E<>MIC==1$ , meaning that MIC eventually can be verified so that the last message can be sent successfully and PTK will be installed by the supplicant.

The verification results in UPPAAL show that the above properties are satisfied.

We have not consider the attacker in this model. In the transition from state  $S_0$  to state  $S_1$ , ANonce is updated to 1, followed by the state transition as a guard so that once ANonce has not been updated normally, the model blocks in state  $S_1$ . Attacker is in accordance with this principle on the handshake attack. This Dos attack arises from the vulnerability of the message  $Msg1$ . Actually, the 4-way handshake protocol is vulnerable to Dos attack during handshake. So we use the guard  $y <= ExpectedTIME1 \&\& ANonce==1$  to make sure that we get the correct ANonce.

The authenticator may distribute a Group Temporary Key (GTK) to supplicants in multicast applications. PTKs are used to encrypt unicast data between a supplication station and an authenticator, and GTKs are used to encrypt multicast data between a supplicant station and an authenticator.

Group-key handshake contains 2-way handshake. The authenticator sends message  $GrpMsg1$  containing the new GTK to each supplicant in the network. The GTK is encrypted and assigned to the supplicant, and protects the data from tampering, by using a MIC. Then, the supplicant confirms receipt of the new GTK and sends reply to the authenticator. MICs are used to provide authentication and message integrity. This 2-way handshake lends simplicity and much less overhead to multicast key generation and distribution to supplicants.

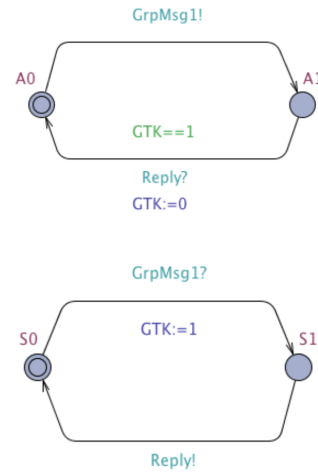


Figure 10. UPPAAL Model for Group Key Handshake Protocol

Figure 10 shows the models in the group-key handshake protocol. Since group-key handshake is 2-way handshake, we only need to define two states respectively. The authenticator sends  $GrpMsg1$  to the supplicant. Once the supplicant synchronizes with the authenticator successfully (represented by  $GrpMsg1?$  in Figure 10), the GTK used to protect the data from tampering is updated (represented by the assignment  $GTK:= 1$  in Figure 10). Once the GTK has been updated to 1,

it means that the GTK has been successfully encrypted. Thus, what we should ensure in our model now is that the supplicant sends reply to the authenticator after confirming receipt of the new GTK. The edge transforming from state  $A_1$  to state  $A_0$  is guarded by the condition  $GTK==1$  so that only when GTK is successfully received, the supplicant can send a reply.

Through the verification of our model, we can prove that the property  $A[] \text{ not deadlock}$  is satisfied, which means that the model is not deadlock. It is further illustrated that once the GTK has been successfully encrypted and sent to the supplicant, group-key handshake will be able to keep proceeding.

## V. CONCLUSION AND FUTURE WORK

This paper analyzes the IEEE802.11i protocol for mutual authentication, group communications and key establishment. And we have considered the case of re-certification in the protocol, which means we need adding timing issues to the extended protocol. Using UPPAAL allows us to simulate, debug and verify the IEEE802.11i protocol in a real time setting. UPPAAL can be used to generate the protocol's simulation path and the study of time allows us to ensure that the re-authentication process will occur. In addition, we introduced a validation analysis of deadlock. Ensuring that the deadlock does not occur in the protocol is the primary requirement for regulatory security.

Designing secure and efficient key management protocol in 802.11i standard is a significant issue. In this paper, we model the 4-way handshake protocol with four synchronization channels, simulating the interaction of the protocol and consider the timeout issues. Based on the formal model, we perform an integrated formal verification of the protocol using UPPAAL. The verification results show that the 4-way handshake protocol will not be deadlock if the Dos attack can be prevented. Thus, in our further study we will consider an enhanced 4-way handshake to repair vulnerability to attack.

Basic analysis for the 4-way handshake protocol in this paper is based on the idealized handshake protocol shown in Figure 6. In our model we have not considered subsequent verification, such as verifying Message Integrity Code based on the PTK. The PTK is generated by concatenating the attributes PMK, ANonce, STA SNonce, MAC address of authenticator and supplicant. In fact, the attacker precisely makes the handshake deadlock by providing the wrong ANonce to derive a wrong PTK. In the future, we will also consider more situations, such as not successfully verifying MIC, in our model.

Furthermore, we have just considered whether the protocol is safe and proceeds normally and simplified the key passing process in this work. Assuming that once the key has been generated, the sub-protocol can run normally. This means that we have not considered emergencies that can cause transformation of the key failed. We will add the impact of the environment into the formal model in the future work as well.

## ACKNOWLEDGEMENT

The work was partially supported by the National Natural Science Foundation of China under grant no. 61772038, 61532019, 61202069 and 61272160.

## REFERENCES

- [1] A. Alabdulatif, X. Ma, and L. Nolle. Analysing and attacking the 4-way handshake of IEEE 802.11i standard. In *Proceedings of 8th International Conference for Internet Technology and Secured Transactions*, pages 382–387. IEEE, 2013.
- [2] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
- [3] C. Baier and J.-P. Katoen. *Principles of Model Checking*. The MIT Press, 2008.
- [4] P. R. D'Argenio, J. Katoen, T. C. Ruys, and J. Tretmans. The bounded retransmission protocol must be on time! In *Proceedings of TACAS '97*, volume 1217 of *LNC3*, pages 416–431. Springer, 1997.
- [5] S. Dustdar, S. Nastic, and O. Scekic. *Smart Cities - The Internet of Things, People and Systems*. Springer, 2017.
- [6] K. Havelund, A. Skou, K. G. Larsen, and K. Lund. Formal modelling and analysis of an audio / video protocol: An industrial case study using uppaal. In *Proceedings of the 18th IEEE Real-Time Systems Symposium*, pages 2–13. IEEE Computer Society, 1997.
- [7] IEEE Std 802-2014. *IEEE standard for local and metropolitan area networks: Overview and architecture*, 2014.
- [8] IEEE Std 802.1X-2010. *IEEE standard for local and metropolitan area networks—port-based network access control*, 2010.
- [9] K. G. Larsen, P. Pettersson, and W. Yi. Diagnostic model-checking for real-time systems. In *Proceedings of the 4th DIMACS Workshop on Verification and Control of Hybrid Systems*, volume 1066 of *LNC3*, pages 575–586. Springer, 1996.
- [10] J. Liu, X. Ye, J. Zhang, and J. Li. Security verification of 802.11i 4-way handshake protocol. In *Proceedings of ICC 2008*, pages 1642–1647. IEEE, 2008.
- [11] X. Liu and A. O. Fapojuwo. Formal evaluation of major authentication methods for IEEE 802.11i WLAN standard. In *Proceedings of VTC Fall 2006*, pages 1–5. IEEE, 2006.
- [12] S. pyo Hong and J. Lee. Supporting secure authentication and privacy in wireless computing. In *2006 International Conference on Hybrid Information Technology (ICHIT)*, pages 594–599. IEEE Computer Society, 2006.
- [13] K. V. K. Raju, V. Vallikumari, and K. Raju. Modeling and analysis of IEEE802.11i wpa-psk authentication protocol. In *Proceedings of 3rd International Conference on Electronics Computer Technology*, pages 72–76. IEEE, 2011.
- [14] R. Singh and T. P. Sharma. A key refreshing technique to reduce 4-way handshake latency in 802.11i based networks. In *Proceedings of 4th International Conference on Computer and Communication Technology*, pages 157–162. IEEE, 2013.
- [15] E. Sithirasanan, S. Zafar, and V. Muthukumarasamy. Formal verification of the IEEE 802.11i WLAN security protocol. In *Proceedings of (ASWEC 2006)*, pages 181–190. IEEE Computer Society, 2006.
- [16] UPPAAL. <http://www.uppaal.org/>.
- [17] R. H. Weber and R. Weber. *Internet of Things - Legal Perspectives*. Springer, 2010.
- [18] X. Xing, E. M. Shakshuki, D. G. Benoit, and T. R. Sheltami. Security analysis and authentication improvement for IEEE 802.11i specification. In *Proceedings of GLOBECOM 2008*, pages 1887–1891. IEEE, 2008.

# SeqBAC: A Sequence-Based Access Control Model

Diogo Domingues Regateiro<sup>1</sup>, Óscar Mortágua Pereira<sup>2</sup>, Rui L. Aguiar<sup>3</sup>

Instituto de Telecomunicações  
DETI, University of Aveiro  
Aveiro, Portugal  
{diogoregateiro<sup>1</sup>, omp<sup>2</sup>, ruilaa<sup>3</sup>}@ua.pt

**Abstract**—Access control, when used in the context of database applications, is aimed to supervise the requests made by legitimate users to access sensitive data. These requests represent actions that a user can perform on a database and they typically read or write data. While this supervision can be formalized at a higher level, e.g. using an access control model such as RBAC, in the end, the data access is done through each authorized action. Therefore, the current access control models enforce their policies on an action by action basis, being unable to support relations of order between them. In many database applications, access to data is not done randomly, but by following very specific sequences of actions which are not supervised. This paper argues that a better security policy can be achieved by supervising these sequences. Thus, previous research is leveraged to propose a formalized model, capable of enforcing access control over the sequences of actions that can complement existing access control models.

*Keywords*—information security, access control, sequence enforcement, database security, SeqBAC.

## I. INTRODUCTION

Access control is a mechanism that limits the activity of legitimate users in a system. There are many strategies to enforce access control in a system, of which we emphasize the main four: discretionary access control (DAC), mandatory access control (MAC), attribute-based access control (ABAC) and role-based access control (RBAC). However, these access control models are not one-size fits all solutions, and so many other access control models exist [1][2][3][4][5]. Nevertheless, RBAC has risen as the dominating access control model used, especially for relational database applications. In this model, a user can only perform some action if he has been given permission to enact the role that governs said action. When it comes to data, actions are usually single read or write operations.

However, actions are not always independent of one another, some are used to collect values that are passed on to subsequent actions or to achieve some higher-level use case. A basic example of this would be a doctor prescribing some drug to a patient while allergies must be accounted for. First, the information about the patient would be selected, then any information about potential allergies for that user queried. This information is then used to filter the drugs that can be prescribed and added to the patient's profile. The referenced access control models do not support this kind of relation between actions to be encoded in the policies. A possible solution is for this dependency logic to be put into the application layer. Unfortunately, if the application is incorrectly implemented or

exploitable, it may be possible to bypass this logic. If this is the case, it would be possible to perform authorized actions in unforeseen ways, such as prescribing a drug without allergies being checked. Another solution is to use stored procedures defined on the DBMS. However, they are not always supported and are not easily manageable since it is difficult to know which actions are being used and in which order.

A solution to prevent this dependency logic between actions from being bypassed is to design and validate the sequences in which actions are being executed. This approach benefits from the fact that designing sequences of actions can be done early in a project lifecycle, helping with implementation later. Moreover, a model based on these sequences could validate the sequences automatically and in real-time. In contrast, manually written code (e.g. application logic, stored procedures, etc.) is prone to implementation mistakes. These mistakes can compromise the correctness of the system, such as forgetting a crucial validation check, which can go unnoticed for extended periods of time. Additionally, by imposing an order of execution, it is possible to provide values for parameters directly from previous actions. This would lead to a more secure solution because it becomes possible to know the origin of the values passed as parameters instead of being only provided by the user.

Thus, we present a sequence-based access control model (SeqBAC) that aims to be able to encode the relations between actions that a user is authorized to perform and guarantee that they are executed in the sequence that they were meant to be. Additionally, the work presented in [6] could be used to implement a tool that could generate the code necessary to follow the defined sequences automatically and access the data. A previous iteration of this model [7] has seen a proof of concept, showing that such a concept is possible to execute and implement at a very basic level. This paper formalizes the results obtained from that proof of concept, expanding the concepts to include sequence branching and subsequence calls.

This paper is divided as follows: section II provides some of the state of the art, section III describes the model in terms of the desired policy, section IV introduces the necessary definitions to formalize the model, section V provides the model formalization and section VI provides our conclusions about the work.

## II. RELATED WORK

There are many access control models in the literature. From the previously mentioned DAC, MAC, RBAC, and ABAC, to the Bell-LaPadula [8] model for government and military

This work is funded by National Funds through FCT - Fundação para a Ciência e a Tecnologia under the project UID/EEA/50008/2013 and SFRH/BD/109911/2015.

DOI reference number: 10.18293/SEKE2018-099

applications, and many others [1][2][3][4]. Other access control models based on finite state machines also exist [9][10], but they usually operate at a higher level, controlling authentication and other access control related tasks. The model proposed in this paper aims to control the lower level database operations.

The Extensible Access Control Markup Language (XACML) [11][12] has been proposed as a standard which includes a language for defining access control policies based on ABAC, an architecture for enforcing them and a processing model which describes how requests are evaluated. However, defining policies to control the sequence of actions a user may perform on a system is not within the original scope of ABAC. XACML could be extended to support the set of rules required to implement SeqBAC policies, but it would just be one implementation of the model herein described. Other access control languages exist, such as the Enterprise Privacy Authorization Language (EPAL) [12][13] which has been proposed to protect the customer's data privacy within a company, and it is another possible language where our model may be implemented on.

Barker [5] states that existing access control models all use the same basic concepts, which are then applied in a restrictive manner. Thus, Barker proposes a meta-model for access control based on these basic concepts and shows some examples of how other access control models are supported by this model. This approach was studied but ultimately seemed inadequate for the access control model being presented, as actions have several order relations between them, i.e. actions can exist in several sequences, and this was not supported by Barker's meta-model. Staab and Muller [14] also introduced the MITRA framework, which is another meta-model for information flow where trust and reputation architectures are in place.

Non-deterministic access control models could also be considered for implementation of sequences of actions, especially when branches are considered and users can follow any of them. Several of these models exist, of which we emphasize: probabilistic models to determine risk [15][16][17], cognitive-based systems [18] and fuzzy theory-based models [19][20]. Ultimately, the SeqBAC model presented here is meant to be deterministic, which would allow security experts to conduct auditing and to keep a level of assurance that no unexpected access decision is made.

The model in this paper builds upon a CRUD expression driven access control model [7][21] and generalizes it so that it no longer depends on the RBAC model to authorize to execute the actions. In [21], an architecture was proposed to enforce access control based on RBAC, and driven by the CRUD expressions that are naturally part of the domain of the applications using the architecture. In [7], an extension to the RBAC model was proposed to support basic sequences of CRUD expressions to complement the role-based approach. We allow sequences of CRUD expressions to be ruled by roles, and users can follow these sequences if they are allowed to play the role. However, this extension was limited to simple chains of CRUD expressions and it lacked a proper formalized model.

While to the best of our knowledge no attempts have been made to create a model that can enforce sequences of actions to access the data, the process of altering an existing model to re-

purpose it for other scenarios is widely used in the literature. Many examples of this practice exist [2] where the RBAC model is extended with geographic information for the purposes of using a user's location to allow or deny access to data. The formalized model in this paper addresses this gap and generalizes the work in [7] to sequences of actions.

### III. BASE POLICY

In this section, the SeqBAC model being proposed will be described in terms of the simplest base scenario that it aims to support.

A simple policy could contain just an ordered set of actions and parameter tuples, where an authorized user could execute the first action, then the second, etc. However, scenarios such as the drug prescription, described in section I, could require a doctor to go back in the order of execution to add a different drug to a prescription after a previous one had been found to cause allergic reactions on a patient. Thus, it is necessary to define which actions a user can take at some point in a use case execution.

Since the policy is meant to restrict the order in which actions can be performed on a database by a legitimate user in the terms described above, such a policy should contain the following:

- A set of actions  $A$  and their input parameters  $P$ .
- A set  $E$  of directed transition relations between actions.
- The set of users  $U$  allowed to execute the policy.

The set of users can be defined explicitly or implicitly through some condition the users must satisfy, such as be playing some role, possess a set of attributes, etc.

This type of policy is more descriptive than other policies, such as RBAC policies, due to the set of transition relations between the actions. There is an initial action, which each user executes first, and then the users can execute other actions by following the transition relations between them. Given the fact that actions are defined with transitions between them, a policy in SeqBAC defines a sequence of actions, hereby known as an action flowchart.

Hence, the SeqBAC model supervises a set of actions that can be executed over the set of available data and will limit the executions of these actions to certain sequences. This list of actions is not required to be complete *a priori* for the model to be used: new actions may be added at any given time by the system's administrator or other authorized users. Furthermore, the sequences of actions may branch, allowing the users multiple actions to choose from to allow flexibility. Sequences of actions may also be reused in other sequences when their purpose is needed in several situations, and this will be pursued in more detail in section V.C.

The importance of these policies is that, in many cases, actions over data are not executed randomly. Instead, actions are executed following some notion of order that is related some use case. However, these sequences are not generally encoded in the access control mechanisms, which enforces access control on an action by action manner. This can lead to unintended results when malicious users can execute actions by impersonating a

legitimate user.

To exemplify one such policy, consider the following set of four actions {A, B, C, D} in the context of an online shop. Action A queries the database for client information to authenticate it, action B queries for the checkout cart of the client, action C allows the client to review its payment options and action D allows it to place an order. In this scenario, we will require action A to be done always first, following it by action B to show the current checkout cart to the client once the authentication succeeds. Then, once the client decides to place the order, the client may want to review and update their payment options before doing so, making action C optional and finalizing with action D.

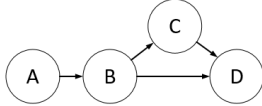


Figure 1. Scenario representation of the relations between actions.

This scenario is represented in Figure 1, where each action is represented by its letter and an arrow connects one action to the next one that naturally follows as per description. If a malicious user breaches the application and tries to obtain payment details, it is unable to do so. The action to review the payment options is in the middle of a sequence of actions, and to reach it the malicious user would have to execute actions A and B. Since action A is used for authentication, unless the malicious user possesses the authentication credentials or breaches the DBMS itself, it will be impossible to access the payment options of the users.

#### IV. MODEL CONCEPTS

Formally, the SeqBAC model is a set of action flowcharts associated with each user. It follows closely the existing concepts that govern flowcharts, and as such we will use flowchart notation to formalize the model.

First, consider the set of defined actions  $A$  defined in formula 1 and a parameter  $P$  defined in formula 2, where actions  $a_1$  to  $a_N$  are the actions defined in the system,  $name$  is a string of characters and  $datatype$  represents the datatype of the associated parameter in the database, if relevant.

$$A = \{a_1, a_2, \dots, a_N\} \quad (1)$$

$$P = (name, datatype) \quad (2)$$

With these concepts, we can define the set of action nodes  $V$  in our model's flowcharts, where each node is a pair of an action and a set of parameters as shown in formula 3. An action allows an authorized user to access or modify some subset of the data.

$$V = \{(a, \{P\})\}, a \in A \quad (3)$$

Having defined the action nodes of the flowchart we now need to define the transitions between them. The set of valid transitions  $E$  are defined as a set of ordered 2-element from  $V$ , which forms the unidirectional transitions between elements of  $V$ . Formally, the set of transitions  $E$  is defined in formula 4.

$$E = \{(u, v) : u, v \in V\} \quad (4)$$

Formula 5 defines the flowchart  $G$ , used to model a high-level use case. Each flowchart is an ordered pair of a set of action nodes  $V$  and a set of transitions  $E$  that connects two action nodes.

$$G = (V, E) \quad (5)$$

Additionally, we define the functions  $ActionSet(G)$  to return the set of action nodes of the flowchart  $G$  and  $TransitionSet(G)$  to return the set of transitions.

Finally, formula 6 defines the set of flowcharts  $SOF$  that makes up the model. Each user  $U$  is then given a subset of  $SOF_U$  which they are authorized to execute, as shown in formula 7.

$$SOF = \{G_1, G_2, \dots, G_M\} \quad (6)$$

$$SOF_U \subseteq SOF \quad (7)$$

#### V. MODEL DEFINITION

In order to enforce SeqBAC, a way to define how a user can be tracked along a flowchart  $G \in SOF$ , is needed. Furthermore, the possible move operations that a user can perform at a given node, as well as what type of information can flow from one node to the next is required.

From the information provided in the previous section, a definition of the SeqBAC model can now be created.

**Definition 1: SeqBAC.** The SeqBAC model has the following components:

- $A$ ,  $P$ , and  $U$  denote actions, parameters, and users, respectively;
- $V$  denotes an action node, which is a tuple of an action and a set of parameters necessary to execute the action;
- $E \subseteq V \times V$  is a relation between action nodes, describing the authorized transitions;
- $G = (V, E)$  and  $SOF$  denote an action flowchart, with the set of actions and the authorized transitions between them and the set of all defined action flowcharts, respectively;
- $auth : U \rightarrow 2^G$  is a function that determines if a user is allowed to access a certain action flowchart:  $auth(u_i) \subseteq SOF, u_i \in U$ .

Given that enforcing access control over sequences of actions is the primary concern, it is required to locate the position of a user within a sequence at any time. To achieve this, the User Access Pointer is defined.

**Definition 2: User Access Pointer.** Given a SOF, the user access pointer (UAP) is a pair of elements  $(G, v)$  that uniquely identifies a flowchart  $G \in SOF$  and the current node  $v \in V$  the user is allowed to use.

This UAP allows a system to keep track of which flowchart the user is using and on which node within it. We will now define how the UAP can be updated in order to move the user within a flowchart, which uses an operation called *Stepping* and it involves moving along a transition of the flowchart.

**Definition 3: Stepping.** Consider the flowchart  $G$  and its UAP

on step  $n$  of the flowchart traversal, denoted  $UAP_n$ . Stepping is the process in which  $UAP_{n+1}$  is generated by referencing a new node such that:

$$\forall_x \forall_y \left( \left( UAP_n = (G, x) \wedge UAP_{n+1} = (G, y) \right) \Rightarrow \right. \\ \left. (x, y) \in \text{TransitionSet}(G) \right) \quad (8)$$

Definition 3 constraints moving from node to node, and therefore updating the UAP, along the transitions between them as previously informally described. When a UAP first references the initial node of the flowchart, we consider it to be in step 1 ( $UAP_1$ ) and the step counter increments with each stepping. A UAP on step 0 ( $UAP_0$ ) is not referencing any node and is used when no flowchart is currently being traversed by the user.

There are several different situations in which *Stepping* may be used, and they differ on the in-degree and the out-degree of the nodes involved, i.e. the number of transitions in and out of a node respectively, as well as the direction of the transitions between them. The transition between nodes will be detailed first without considering the information that can flow between them.

### A. Stepping

We will now describe several scenarios in which *Stepping* operations may occur and how they are handled within the context of our model. The most trivial *Stepping* operation occurs when we have two nodes  $A$  and  $B$ , where node  $A$  has an out-degree of 1 and node  $B$  an in-degree of 1, as shown in Figure 2. In this case, if the UAP is at node  $A$ , then by Definition 3 it follows that it can only move to node  $B$ .

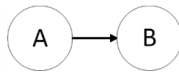


Figure 2. Stepping's trivial case.

We will now analyze the different type of *Stepping* operations that can occur when the in-degrees and out-degrees differ. When the out-degree of node  $A$  is bigger than 1, then we have more than one node that can satisfy Definition 3 and we say that it causes *Splitting* in the sequence. In this case, the user can decide which node to go to. This case is analogous to the piece of pseudo-code on Figure 3, where the action  $A$  is always executed, and then either action  $B$  or  $C$  are executed depending on some condition criteria.

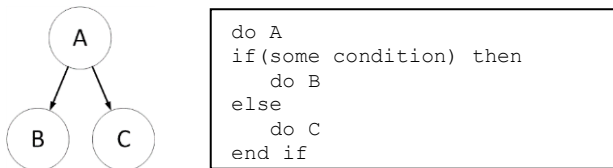


Figure 3. Splitting example and associated code.

We can now consider the opposite situation, where the in-degree of a node is bigger than 1. This is shown in Figure 4 and it represents a situation where two or more branches of a sequence join at the node. For the user, it is exactly the same situation as in a trivial *Stepping* operation, but he would have been able to reach that node through some other sequence of nodes in the flowchart. In the example, node  $C$  is simultaneously reachable from both nodes  $A$  and  $B$ . This case is analogous to the

piece of pseudo-code on Figure 4, where either action  $A$  or  $B$  are executed depending on some condition criteria. Then, action  $C$  is executed independently of the condition criteria.

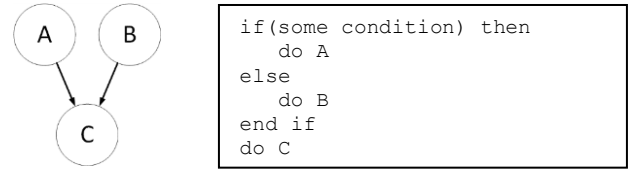


Figure 4. Merging example and associated code.

So far, we've only discussed situations where *Stepping* moves the user forward in a flowchart. However, if a transition exists in both directions between two nodes, then it is possible for the user to step between those two nodes as many times as it wants given that the only restriction to *Stepping* is the Definition 3. We call this situation a *Cycle* in the sequence, as demonstrated in Figure 5, and it could potentially be created with many in-between nodes. While this can make sense in some situations, it should be possible to restrict the number of times the user can go back to the same node. Additional restrictions will be discussed in section V.B.2). This case is analogous to the piece of pseudo-code on Figure 5, where the actions  $A$ ,  $B$ , etc. can be executed in a cycle. The cycle ends when the execution transitions out of the cycle, normally on the last node.

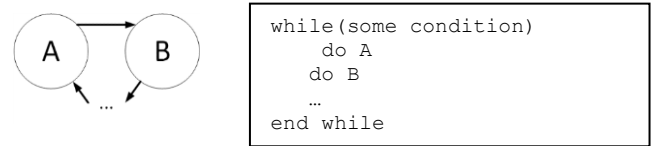


Figure 5. Cycle example and associated code.

One particular case of a *Cycle* is when one is created using only one node that can transition to itself. We call this case a *Loop* and it allows a user to reuse the same node several times, possibly with information obtained from past accesses. Bulk operations can benefit from this type of transition.

### B. Information Flow

One important fact to consider is that most data access is not static, i.e. parameters can be used to select or modify data. In fact, some of the data used as input parameters might need to have its source validated to ensure that the user does not access data that he is not authorized to access. To achieve this type of data validation it is proposed that nodes can use data from previous nodes as parameters. We refer to this concept as the information flow and we will define it in this section.

#### 1) User Context

To handle the passing of information between nodes, we need to create a user context that contains all the data a user accessed and that a node can use to parameterize the data access. To define this context, we will first define the *Accessed* predicate that indicates whether a node  $v$  was accessed in the past or not.

**Definition 4: Accessed( $G, v$ ).** A node  $v \in \text{ActionSet}(G)$  for a given flowchart  $G$  is said to have been accessed when a user's UAP possessed a reference to node  $v$  on at least one step leading up to the current step  $N$ :

$$\forall v \in ActionSet(G) \exists n \leq N (Accessed(G, v) \Rightarrow UAP_n = (G, v)) \quad (9)$$

**Definition 5: User Context.** Given a user  $U$  and each graph  $G$  from its set of flowcharts  $SOF_U$ , the User Context of user  $U$  ( $UC_U$ ) is a pair of elements containing the current UAP on step  $N$  ( $UAP_N$ ) and a set of previously accessed nodes by user  $U$ , which can be referenced by the predicate  $AccessedSet(UC_U)$ .

$$UC_U = (UAP_N, \{v : v \in ActionSet(G) \wedge Accessed(G, v)\}) \quad (10)$$

This  $UC_U$  is updated each time a *Stepping* operation occurs and it can be used by a node to obtain values to parametrize the data access. This means that the user does not provide the parameters himself, ensuring that the data is valid and that the user can request it. However, the UC must be made secure to prevent a user from breaching it in any way. Additionally, when a user stop traversing a flowchart, the  $UC_U$  should be emptied so the previously accessed data cannot be used out of context. Formally, the reset operation of the  $UC_U$  empties the set of accessed nodes and puts the current UAP back to step 0:

$$Reset(UC_U) : UC_U = (UAP_0, \emptyset) \quad (11)$$

There are several ways to implement this method of passing a result of a previous action to another as a parameter. However, one way to do this could involve the usage of a protected server that caches the request results sent to each client, and then it would automatically pass the necessary information to each action when the client requests it to be executed.

## 2) Information Flow Restriction

One important aspect to ensure that data is not accessed out of context is the ability to prevent an action from using some specific data obtained through a previous action as parameter inputs. Thus, a method to remove unneeded data from the UC and free resources on the system is desirable.

To address this, we introduce the ability to revoke access to the data obtained from a previously accessed node. The revocation is done automatically using a list of nodes targeted for revocation, which we call the revocation list. This list can exist in the transitions between nodes or on each node. We will analyze both options.

**Definition 6: Revocation List.** Given a flowchart  $G \in SOF$ , a revocation list  $R$  is a set of previously accessed nodes in  $ActionSet(G)$  that must prevent further access to their data.

$$R = \{v \in ActionSet(G)\} \quad (12)$$

Since the enumerated nodes in the list prevent any further access to their data, it cannot be used as parameter values for consequent access attempts. We must now update the user context definition to contemplate the revocation list, i.e. the user context for some user  $U$  and a flowchart  $G$  must now contain, at a given step  $N$ , the list of previously accessed nodes that have not appeared on any revocation list, as shown in formula 13.

$$UC_U = (UAP_N, \{v : v \in ActionSet(G) \wedge Accessed(G, v) \setminus R\}) \quad (13)$$

When the revocation list is encoded in the transitions between nodes, the set of transitions will then be defined by an ordered triplet of two nodes and a revocation list:

$$E = \{(u, v, R) : u, v \in V\} \quad (14)$$

When a *Stepping* operation is carried out, the revocation list

associated with it is processed and the nodes in the list can be removed from the list of nodes in the user context.

In the other solution, i.e. placing the revocation list at each node, the revocation list is associated with each node in the flowchart  $G$  instead:

$$V = \{(a, \{P\}, R)\}, a \in A \quad (15)$$

Whenever a *Stepping* operation is carried out and some node  $v$  is referenced by the UAP, following the restriction imposed in Definition 3, the user context must be updated with the revocation list by subtracting the list from the user context's list of accessed nodes. Both approaches are valid and which one is used depends solely on the ease of implementation.

## C. Subsequence Calls

We will now describe the process by which sequences of actions may be reused on other sequences, also known as a sub-sequence call. Sub-sequence calls are operations that move the UAP to the root of another flowchart  $G$  to perform some action that is required by multiple other flowcharts, making it a common action. Figure 6 shows this idea, where the node  $A'$  on the flowchart to the right is a call to the entire sub-sequence to the left. Two main types of sub-sequence calls are considered in this work: dependent and independent.

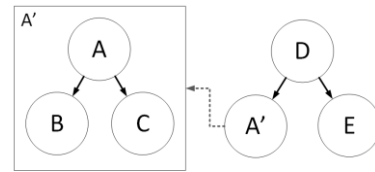


Figure 6. Sub-sequence calls example.

Dependent calls are distinguished from independent calls by their need to use the current UC. As the name implies, dependent calls require the UC to be passed to the sub-sequence, allowing access to the data obtained from the initial flowchart.

Independent calls, however, do not require the data from the initial flowchart to perform the data access. This ensures that data is not misused and passed only on a need-to-know basis. Just like the *Stepping* operation, the sub-sequence call allows the user to move to another node except the node exists on a different flowchart. When the UAP reaches the end of the sub-sequence  $G'$ , the UC from it is copied to the initial flowchart  $G$ , allowing the node the user moves into in the original flowchart  $G$  to access the data obtained, if any. This doesn't violate the definition of the User Context (Definition 5) since the set of accessed nodes do not have to belong to a single flowchart.

More formally, when a call to a sub-sequence finishes and returns a  $UC'$ , the initial  $UC_n$  is updated to  $UC_{n+1}$  as follows:

$$UC_{n+1} = (UAP_n, AccessedSet(UC_n) \vee AccessedSet(UC')) \quad (16)$$

Note that the original UAP remains the same as it was before the sub-sequence call was made, and it is only updated when the user execution returns to the original flowchart. It is also important to note that the dependent calls and the  $UC'$  returned by the sub-sequence call can have revocation lists associated with them as described in section V.B.2). However, since the new  $UC'$  is copied from the original  $UC$ , the revocation list of a

dependent call only affects  $UC'$ , leaving  $UC$  untouched. The  $UC$  can be restricted using a revocation list after the sub-sequence call terminates and the user moves to the next node via *Stepping*.

Additionally, when the  $UC$  is copied for the execution of a sub-sequence, the node of the original sequence is not passed along with it. This means that when a sub-sequence finishes execution, the system still needs to know where to point the user to continue the sequence execution. Therefore, every time a subsequence call is performed, the current  $UC$  should be saved.

A natural solution to save the  $UC$ s in is a stack, in which the last item to be stored is the first to be removed. This way, when a sub-sequence is called, the  $UC$  used is put into the stack and copied to the sub-sequence. When a sub-sequence terminates, the last  $UC$  is removed from the stack, merged with the current one, and the previous  $UC$  recovered as shown in formula 16.

#### D. Implementation

While this paper is concerned primarily with defining the SeqBAC model, it can also be beneficial to consider how such a model can be implemented. The unique way that actions interact with each other through transition relations makes it clear that a graph is the type of structure seems to be the most appropriate for storing SeqBAC policies. Furthermore, graph databases, such as Neo4j, allow to define properties on each node and edge. This feature could be used to store things such as subsequence calls by referencing another graph, the revocation list for accessed data pruning purposes and other data.

However, storing the actions and the relations between them is not enough. If the access control system only checks if a user is authorized to execute some action, then either the user knows the sequences or can experience many authorization errors. This can be expected to happen since there can be many ways to fulfill a use case and a defined sequence may only account for a specific way to do it. A solution to this implementation issue is the development of a tool that can parse the defined sequences and then generate the necessary code that follows them automatically. A previous work, presented in [6], shows how this could be used to integrate a similar idea into existing DBMS solutions. This way, when an application using SeqBAC is being developed, the developers do not need to master the policies defined as action flowcharts.

## VI. CONCLUSION

In this paper, the SeqBAC model was introduced and formalized. The model was designed to enforce access control policies over sequences of actions, allowing users to execute them in controlled sequences, and extends a previous work.

This paper also considers how this model could be implemented, addressing the issue of developers having to master the defined sequences of actions with the proposal of using a tool to parse the defined sequences and generating the code to use them automatically. While this model requires some work to define the sequences of actions when compared to other models that allow unrestricted access to data, it helps to ensure that the use cases are implemented correctly faster.

Regarding future work, an actual implementation of the example discussed in section V.D is thought to be next natural

step. Additionally, tools to define policies, validate source-code and generate mechanisms based on the defined policies are also being considered. These would allow developers to know easily what operations are available at any point during the execution of a sequence, preventing the need for them to master the policies, and to ensure the overall correctness of their code.

## REFERENCES

- [1] E. Bertino, P. A. Bonatti, and E. Ferrari, "Trbac," *Proc. fifth ACM Work. Role-based access Control - RBAC '00*, no. May 2016, pp. 21–30, 2000.
- [2] M. L. Damiani, E. Bertino, B. Catania, and P. Perlasca, "Geo-Rbac," *ACM Trans. Inf. Syst. Secur.*, vol. 10, no. 1, p. 2–es, 2007.
- [3] E. Tarameshloo and P. W. L. Fong, "Access control models for geo-social computing systems," in *Proceedings of the 19th ACM symposium on Access control models and technologies - SACMAT '14*, 2014, pp. 115–126.
- [4] I. Ray and M. Toahchoodee, "A Spatio-temporal Role-Based Access Control Model," *Data Appl. Secur. XXI*, vol. 4602, pp. 211–226, 2007.
- [5] S. Barker, "The next 700 access control models or a unifying meta-model?," *Proc. 14th ACM Symp. Access Control Model. Technol.*, pp. 187–196, 2009.
- [6] Ó. M. Pereira, D. D. Regateiro, and R. L. Aguiar, "Secure, Dynamic and Distributed Access Control Stack for Database Applications," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 25, no. 09n10, pp. 1703–1708, Nov. 2015.
- [7] Ó. M. Pereira, D. D. Regateiro, and R. L. Aguiar, "Extending RBAC Model to Control Sequences of CRUD Expressions," *SEKE'14 - Intl. Conf. Softw. Eng. Knowl. Eng.*, 2014.
- [8] D. Bell and L. LaPadula, "Secure Computer Systems: A Mathematical Model. Volume II.," vol. II, no. May 1973, 1973.
- [9] Bin Duan and Bing Liu, "Design of security state machine of access control for control object based on IEC 61850," in *2006 IEEE Power Engineering Society General Meeting*, 2006, p. 3 pp.
- [10] M. Thirumaran, P. Dhavachelvan, D. Aishwarya, and R. Shanmugapriya, "Finite State Machine based Access Control Mechanism for Web Service Work Flow Management," *IERI Procedia*, vol. 4, pp. 391–397, 2013.
- [11] B. Parducci and H. Lockhart, "eXtensible Access Control Markup Language (XACML) Version 3.0," *OASIS Standard*, 2013.
- [12] G. Yee, *Privacy Protection for E-Services*. Idea Group Inc (IGI), 2006.
- [13] J.-W. Byun, *Toward Privacy-preserving Database Management Systems - Access Control and Data Anonymization*. ProQuest, 2007.
- [14] E. Staab and G. Muller, "MITRA: A Meta-Model for Information Flow in Trust and Reputation Architectures," *arXiv Prepr. arXiv1207.0405*, p. 19, Jul. 2012.
- [15] R. McGraw, "Risk-Adaptable Access Control ( RADAC )," in *Privilege Manag. Work. NIST-National Inst. Stand. Technol. Technol. Lab.*, 2009.
- [16] D. R. dos Santos, R. Marinho, G. R. Schmitt, C. M. Westphall, and C. B. Westphall, "A framework and risk assessment approaches for risk-based access control in the cloud," *J. Netw. Comput. Appl.*, vol. 74, pp. 86–97, Oct. 2016.
- [17] S. Kandala, R. Sandhu, and V. Bhamidipati, "An Attribute Based Framework for Risk-Adaptive Access Control Models," in *2011 Sixth International Conference on Availability, Reliability and Security*, 2011, pp. 236–241.
- [18] IBM, "Cognitive Security White Paper," 2016. [Online]. Available: <http://cognitivesecuritywhitepaper.mybluemix.net/>. [Accessed: 11-Jan-2017].
- [19] C. Martínez-García, G. Navarro-Arribas, and J. Borrell, "Fuzzy Role-Based Access Control," *Inf. Process. Lett.*, vol. 111, no. 10, pp. 483–487, 2011.
- [20] J. Kacprzyk, S. Zadrozny, and G. De Tré, "Fuzziness in database management systems: Half a century of developments and future prospects," *Fuzzy Sets Syst.*, vol. 281, pp. 300–307, Dec. 2015.
- [21] Óscar Mortágua Pereira, D. D. Regateiro, and R. L. Aguiar, "Role-Based Access Control Mechanisms," ... (*ISCC*), *2014 IEEE ...*, vol. 2, no. 1, pp. 1–7, Jun. 2014.



# A Self-Adaptation Framework of Microservice Systems

Shuai Zhang, Xinjun Mao, Peini Liu and Fu Hou

College of Computer

National University of Defense Technology

Hunan, China 410073

{zhangshuai16a, xjmao, liupeini16, houfu}@nudt.edu.cn

**Abstract**—Microservice has been more and more applied to build software systems in industry field and research in academic field. And software systems are increasingly expected to dynamically self-adapt to accommodate resource variability, changing user needs, and system faults. Compared with traditional software systems, microservice systems have some characteristics, such as the highly self-contained components and the dynamic running instance, which pose challenges to traditional self-adaptation methods. Therefore, it needs to propose corresponding techniques and methods to cope with the characteristics of microservice systems. This paper analyzes the special self-adaptive requirements of microservice systems, proposes a microservice reference model, which describes basic elements and their relationships of microservice systems. Then we present a microservice system self-adaptation framework *MSSAF* to support the self-adaptation of microservice systems. We illustrate the feasibility and effectiveness of our approach in the context of a microservice system case.

**Keywords**—self-adaptation; microservice; reference model; framework

## I. INTRODUCTION

Microservices recently demonstrated to be an effective architectural paradigm to cope with software complexity and scalability [1]. Although microservice has received more and more pay attention, there is no generally accepted definition for microservice. A widely-recognized concept of microservice is proposed by Martin Fowler and James Lewis [2]. The success of the paradigm has been demonstrated in some domains, including mission-critical systems [3]. However, while microservice systems run in open and dynamic environment, there are still many uncertainties, e.g. changing user requirements and unpredictable system errors. To deal with the uncertainties, microservice systems need dynamically self-adapt to state changes of external environment and itself.

Microservice systems have some characteristics, e.g. platform-dependent and one microservice may have multiple instances. Since traditional software systems don't have these characteristics, the traditional self-adaptation methods have no specific solutions.

To realize the self-adaptation of microservice systems more efficient, this paper proposes a microservice reference model

and a microservice system self-adaptation framework *MSSAF*. The microservice reference model describes the basic elements and their relationships of microservice systems. Microservice system self-adaptation framework *MSSAF* integrates some necessary tools to provide support for the self-adaptation of microservice systems. Based on these methods and tools, this paper has achieved better cost-effectiveness in self-adaptation of microservice systems.

The rest of this paper is structured as follows. Section II discusses some related works. Section III describes a microservice reference model. Section IV gives a detailed description of microservice system self-adaptation framework *MSSAF*. In section V, our works are illustrated by an intelligence system which is a microservice system. Finally, we summary this paper and discuss further works.

## II. RELATED WORK

Nowadays microservice has become more and more popular in software engineering field. There are some summaries of existing studies on microservice find that most of the studies focused on microservice application, method and architecture [4]. Those studies mainly focus on maintainability, extensibility, complexity, flexibility and so on, while there are few studies related to the self-adaptivity of microservice. Nicola Dragoni et al. describe the origin, current and future of microservice in detail [1]. Franco Callegati et al. describe a service-oriented architecture that exploits the microservices orchestration paradigm to enable the creation of new services [5]. Massimo Villari et al. present an Orchestration Broker for each involved Fog computing node[6]. Despite these works consider the characteristics of microservice, there is no self-adaptation in their works. Sara Hassan and Rami Bahsoon refer to microservice and self-adaptation in their work [7]. But they consider the self-adaptation microservice from a design viewpoint without concrete implementation.

The aim of self-adaptation is to let the system monitor itself and based on its goals reconfigure or adjust itself to satisfy the changing conditions, or if necessary degrade gracefully [8]. In recent years, many researchers have proposed many methods to study the self-adaptation from different aspects. Shang-Wen Cheng and David Garlan propose a self-adaptation framework Rainbow which is based on software architecture [9]. Denaro

Giovanni et al. proposes an approach to design self-adaptive service-oriented architectures [10]. Marcello Thiry and Roger Anderson Schmidt present relevant approaches of self-adaptive systems driven by runtime models [11]. Besides, Luca Florio and Nitto focus on how to add autonomic capabilities to microservices without changing the way they are implemented but exploiting their containers [12]. The paper considers that microservice is a kind of distributed component, but they mainly discuss about autonomic capabilities which are different from self-adaptive capabilities.

In the above researches, there are few works related to self-adaptation in the study of microservice. To cope with the characteristics of microservice, this paper analyzes the characteristics and proposes corresponding solution to realize the self-adaptation of microservice systems.

### III. MICROSERVICE REFERENCE MODEL

Microservice manages growing complexity by functionally decomposing large systems into a set of independent services. Despite microservice is an architectural pattern emerging out of Service-Oriented Architecture (SOA), it shows some important distinctive characteristics that are different from SOA:

**Platform-dependent:** There may be a lot of microservices in microservice software systems, communications are complexity and configurations are rather difficult and can also be prone to error. To make up these limitation, corresponding platforms must be developed to manage MSS.

**Multi-instance:** In microservice systems, instance is a running entity of a microservice, and a microservice can have multiple instances. At runtime, there are a lot of microservice instances that interact to realize the application logic of microservice systems.

The characteristics mentioned above propose some special needs for the self-adaptation of microservice systems. Since traditional self-adaptation methods don't have special solutions for these characteristics, the best effect cannot be achieved when realizing the self-adaptation of microservice systems. Therefore, it is necessary to propose some methods to deal with the characteristics. Based on descriptions of microservice architecture in some microservice literatures [13][14][15], we propose a microservice reference model (show in Fig. 1). The model consists of three layers, including system layer, service layer and instance layer.

**System layer** consists of three parts, microservice system realize the business logic, management platform provides management capabilities for microservice system, and limitations impose constraints on microservice system. **Service layer** consists of some microservices and corresponding protocols, microservice communicates with each other through protocols to compose microservice systems. **Instance layer** includes containers and microservice instances. Microservice instances run in containers. A set of microservice instances compose a microservice and they communicate with other microservice instances to realize concrete functions. The reason for such layering is as follow. System layer realizes business logic, mainly focuses on high-level goals and doesn't involve concrete implementation. Service layer focuses on how

microservices cooperate to provide services for the system layer. Instance layer mainly related to the specific implementation. These three layers provide different observation dimension for microservice systems, and clearly express the characteristics which have mentioned earlier.

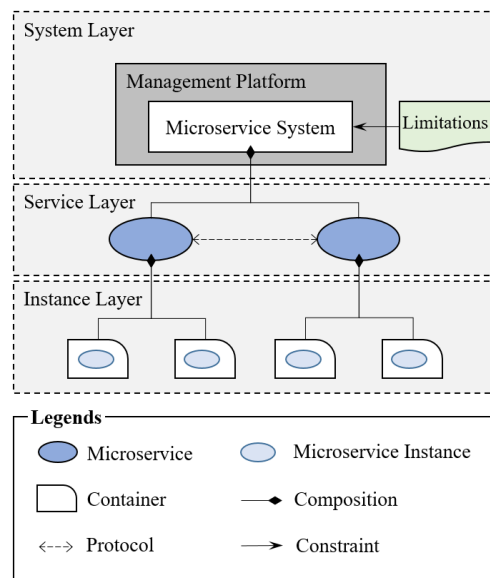


Fig. 1 Microservice reference model

### IV. MSSAF: SELF-ADAPTATION MICROSERVICE SYSTEM FRAMEWORK

The customizable self-adaptation framework has many advantages. For example, it reduces the cost of development by providing a substantial base of reusable infrastructure greatly. And it allows engineers to tailor the framework to different systems with relatively small increments of effort by providing separate customization methods.

To make microservice systems self-adaptation, we refer to the MAPE loop proposed by IBM [16]. For each part of the MAPE loop, we provide a corresponding tool, meanwhile we also provide some other necessary tools. Based on the microservice reference model, this paper proposes a Microservice System Self-Adaptation Framework (MSSAF) (as shown in Fig. 2). To automate system self-adaptation, we provide a self-adaptation strategy description language to represent self-adaptation scenario.

MSSAF consists of three parts, including translation tools, self-adaptation engine and microservice systems. Translation tools is used to translate the self-adaptation strategies, the results of translation are used to support self-adaptation engine to complete the self-adaptation logic. Self-adaptation engine communicates with other two parts to implement self-adaptation. Management platform and microservice systems correspond to the system layer of microservice reference model.

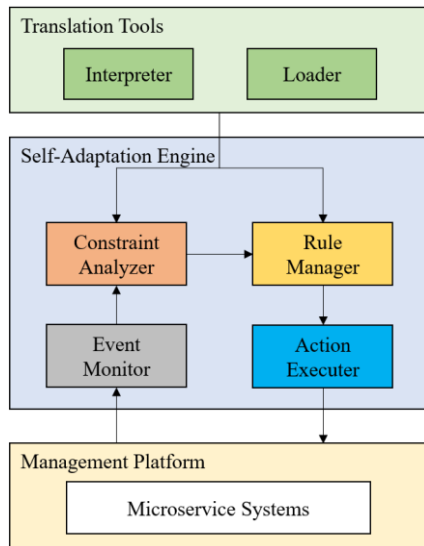


Fig. 2 Self-adaptation microservice system framework

The reason why MSSAF has no the service layer and instance layer of microservice reference model is that we mainly focus on the self-adaptation process of microservice systems in MSSAF. The actual adjustment objects are the microservice and microservice instance which are the parts of microservice systems.

**Self-adaptation engine** consists of several components that provide the monitoring, detection, decision, and action capabilities of self-adaptation. *Event Monitor* obtains the monitoring data of microservice systems from the management platform. *Constraint Analyzer* uses informations which come from Event Monitor to judge whether there is a constraint violation. There are some alternative constraints, and the thresholds are specified in self-adaptation strategies by users. *Rule Manager* selects corresponding rules after triggering by Constraint Analyzer. A self-adaptation rule consists of two parts, including condition and action. The condition includes a signal which is sent by Constraint Analyzer for indicating if the rule is triggered, and it may include some essential constraints which are specified at design time. The actions of self-adaptation rules are the abstract operations of different objects. *Action Executer* performs the actual operations for microservice systems according to the actions of self-adaptation rules which are managed by Rule Manager.

**Translation tools** include an Interpreter and a Loader. Strategies cannot be directly executed and needs to be interpreted before running. Since we provide some common self-adaptation strategies, it needs a tool for static translation. And when there is a need for adding new self-adaptation strategies at runtime, it needs to provide a tool for dynamic loading. Our translation tools can be customized for different program languages and platforms.

## V. ILLUSTRATION OF MSSAF WITH INTELLIGENCE SYSTEM

Since there are few related studies as similar as our works, it cannot compare our works with others. This paper adopts an

intelligence system to illustrate the feasibility and availability of MSSAF.

The intelligence system is used to collect and manage intelligence from different sources and provides the functions of search, analysis, distribution for these intelligences. Besides, the intelligence system also provides a frontend service and a user service for access and management. When there are many users accessing the system in a short time, it will need expand to reduce CPU load. Correspondingly, when user access is too small, it will need shrink to be cost saving. To meet these requirements, we use MSSAF to design and realize the self-adaptation of the intelligence system.

To illustrate the feasibility of intelligence system self-adaptation by using MSSAF, we use the CPU usage rate of microservice instance as the quality attribute that self-adaptation concerns. We monitor the CPU usage of a microservice instance for a period of time, and take the average as CPU usage rate of this instance, namely "MSICPUUsage".

**Self-adaptation scenario:** If there is a swift growth of MSICPUUsage in a short period of time, and there is a threshold, such as 40%. Once the MSICPUUsage exceeds this threshold, the self-adaptation rule is triggered. When the constraints in the self-adaptation rule are satisfied, this self-adaptation rule will be executed. Finally, the specific modification is mapped to the platform by a series of self-adaptation actions.

**Experiment:** The experimental object is analysis microservice (ams), the experimental setup is that the initial instance number is 2, the refresh interval is 120s, the CPU usage rate threshold is set to 40%, and the number of microservice instance is limited to no more than 5. The translation of strategy StrategyMSICPUOverload is shown in Fig. 3.

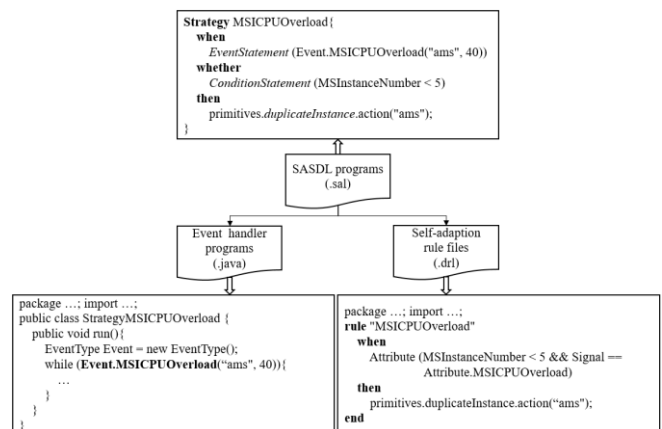


Fig. 3 The translation of self-adaptation strategy StrategyMSICPUOverload

In the experiment, we use the Locust performance testing tool to simulate the stress test, it sets as 500 users access 1000 times per second, and it stops after 10 minutes. Fig. 4 indicates the changes of the number and CPU usage rate of microservice instance during the experimental process.

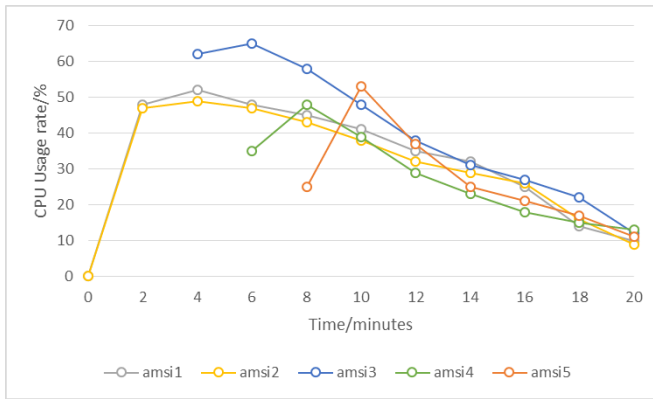


Fig. 4 The diagram of experiment process

As shown in Fig. 4, analysis microservice have two instances amsi1 and amsi2 at beginning. There are large numbers of user accesses in 2 minutes, the CPU usage rate of microservice instance increases rapidly, then the overloading strategy is triggered. The increased amsi3 handles large numbers of accesses in 4 minutes, CPU usage rate is too high, and the overloading strategy will be triggered. The increased amsi4 begins to handle user requests in 6 minutes, the first 3 instances' CPU usage rates still exceed the threshold 40%, and the overloading strategy continues to be triggered. The instance amsi5 was added in 8 minutes, while other instances' CPU usage rates exceed the threshold, the strategy won't be triggered now because the self-adaptation rule set the number of microservice instance to be no more than 5. The stress test has stopped after 10 minutes, the CPU usage rates of the five instances tend to get closer because of the load balancing.

Through the experiment above, we illustrate that MSSAF can provide some preliminary self-adaptation capabilities for microservice systems. From the experimental results, the self-adaptation implementation of MSSAF can utilize the characteristics of microservice systems very well and conduct elastic expansion on more fine-grained. In short, MSSAF has the potential to satisfy the self-adaptation and customization requirements of microservice systems.

## VI. CONCLUSION

To deal with the problems that microservice systems need to be self-adaptation and traditional self-adaptation methods cannot cope the characteristics of microservice systems, we analyze the characteristics of microservice systems and propose a microservice reference model. To implement the self-adaptation of microservice systems and provide reuse infrastructures, we propose a self-adaptation microservice system framework *MSSAF*, which can implement the self-adaptation of microservice systems. Through an example of an intelligence system, we illustrate that *MSSAF* can provide self-adaptation capabilities for microservice systems.

In future work, we would like to improve our approach by enhancing these tools, such as adding strategy conflict detection for Constraint Analyzer, adding strategy selection

algorithm for Rule Manager, etc. These may be our future research directions. In addition, each component of self-adaptation engine can be published as a microservice, then it will be possible to realize the self-adaptation of self-adaptation logic via multiple MAPE loops to expand the self-adaptation capabilities.

## ACKNOWLEDGMENT

This research is supported by research grants from Natural Science Foundation of China under Grant No. 61532004 and 61379051.

## REFERENCES

- [1] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina, "Microservices: yesterday, today, and tomorrow," *Present and Ulterior Software Engineering*, Springer, Cham, 2017.
- [2] J. Lewis and M. Fowler, *Microservices*, <http://martinfowler.com/articles/microservices.html>.
- [3] N. Dragoni, I. Lanese, S. T. Larsen, M. Mazzara, R. Mustafin, and L. Safina, "Microservices: How To Make Your Application Scale," *arXiv preprint arXiv:1702.07149*, 2017.
- [4] C. Pahl, and P. Jamshidi, "Microservices: A Systematic Mapping Study," in *International Conference on Cloud Computing & Services Science*, 2016, pp. 137-146.
- [5] F. Callegati, G. Delnevo, A. Melis, S. Mirri, M. Prandini, and P. Salomoni, "I want to ride my bicycle: A microservice-based use case for a MaaS architecture," in *IEEE Symposium on Computers and Communications*, 2017, pp. 18-22.
- [6] M. Villari, A. Celesti, G. Tricomi, A. Galletta, and M. Fazio, "Deployment orchestration of microservices with geographical constraints for Edge computing," in *IEEE Symposium on Computers and Communications*, 2017, pp. 633-638.
- [7] S. Hassan, and R. Bahsoon, "Microservices and Their Design Trade-Offs: A Self-Adaptive Roadmap," in *IEEE International Conference on Services Computing*, 2016, pp. 813-818.
- [8] D. Weyns, "Software Engineering of Self-Adaptive Systems: An Organised Tour and Future Challenges," *Handbook of Software Engineering*, Springer, 2017.
- [9] S.-W. Cheng, "Rainbow: Cost-Effective Software Architecture-Based Self-Adaptation," *Dissertations & Theses - Gradworks*, 2008.
- [10] G. Denaro, D. Tosi, and D. Schilling, "Towards self-adaptive service-oriented architectures," in *Workshop on Testing*, 2006, pp. 10-16.
- [11] M. Thiry, and R. A. Schmidt, "Self-adaptive Systems Driven by Runtime Models," in *The International Conference on Software Engineering and Knowledge Engineering (SEKE)*, 2017, pp. 248-253.
- [12] L. Florio, and E. D. Nitto, "Gru: An Approach to Introduce Decentralized Autonomic Behavior in Microservices Architectures," in *IEEE International Conference on Autonomic Computing*, 2016, pp. 357-362.
- [13] D. Namiot, and M. Sneps-Sneppé, "On Micro-services Architecture," *International Journal of Open Information Technologies*, vol. 2, no. 9, pp. 24-27, 2015.
- [14] J. Thönes, "Microservices," *IEEE Software*, vol. 32, no. 1, pp. 113-116, 2015.
- [15] M. Mazzara, K. Khanda, R. Mustafin, V. Rivera, L. Safina, and A. Sillitti, "Microservices Science and Engineering," in *International Conference in Software Engineering for Defence Applications*, Springer, Cham, 2016, pp. 11-20.
- [16] G. Agarwal, R. Shah, J. Walrand, H. A. Alzoubi, S. S. Lee, M. Rabinovich, O. Spatscheck, V. D. M. Jacobus, I. Avramopoulos, and M. Suchara, "An Architectural Blueprint for Autonomic Computing," *IBM White Paper*, 2006, pp. 31.

# A Framework to Support the Development of Self-adaptive Service-oriented Mobile Applications

William Filisbino Passini<sup>1</sup> and Frank José Affonso<sup>2</sup>

Department of Statistics, Applied Mathematics and Computation

São Paulo State University – UNESP

PO Box 178, 13506-900, Rio Claro, SP, Brazil

<sup>1</sup>william.passini@gmail.com, <sup>2</sup>frank@rc.unesp.br

**Abstract**—Today’s society is increasingly dependent on the use of mobile devices, which have changed over these last 10 years the way people perform their daily tasks. This can certainly be one of the factors that has boosted the demand for development of high-quality Mobile Applications (MobApps). In short, to improve the efficiency of the development life cycle, these applications often use third-party components (e.g., software components, web services, and other mobile applications). Service-oriented MobApps have been a feasible alternative to overcome hardware limitations of these devices to increase the processing and storage capacity. In another perspective, it is also noted a change in the behavior of users of MobApps and their needs, which require applications capable of modifying their structure and/or behavior at runtime. Thus, this paper presents a framework to support the development of Self-adaptive Services-oriented MobApps (Self-MobApps), which enable adaptation of services at runtime. To show the feasibility of our framework, a case study for a smart restaurant was conducted. The results of this study enable us to create a positive perspective on the contribution of our framework to the research communities involved.

**Index Terms**—Framework; Mobile Applications; Service Computing.

## I. INTRODUCTION

The complexity of software systems and their computational environments has increased in the last years. Nowadays, our society has become increasingly dependent of such systems, which must be able to work in 24/7 mode (i.e., 24 hours per day, seven days per week). Thus, it can be noted that most human daily tasks are managed by applications embedded (i.e., Mobile Applications – MobApps) into mobile devices (e.g., smartphones or tablets), which allow on-line access to information regardless of the users’ location [1], [2], [3]. Regarding the development, such applications can have, at the same time, some items: (i) ad-hoc components and applications developed by third-party; (ii) on-line services; and (iii) platform-dependent components to access device-specific hardware (e.g., camera, GPS – Global Positioning System, microphone, among others) [4].

Mobile devices have some physical limitations (e.g., processing and storage) compared to personal computers. For these reasons, research has been boosted to minimize the impact of such limitations and, at the same time, to facilitate access to information providing mobility to their users. Based on the presented context, the integration of MobApps into SOA-based (Service-Oriented Architecture) systems have been a feasible alternative to overcome these limitations. In short, SOA provides an architectural model that enables services to be published by service providers, discovered and consumed

by the stakeholders (i.e., client applications, other services, among others) by means of platform-independent communication process based on set of XML-based (eXtensible Markup Language) standards [1], [5], [6].

SOA-based systems have played an important role in the development of distributed applications over the Internet. In this context, web services can be considered elements of first class to support the development of applications based on services in heterogeneous environments. Moreover, such applications, regardless of type (e.g., distributed, mobile, or web), must be prepared to deal with the changes at runtime, which can be performed to meet the user’s new needs (e.g., new requirements) or autonomously react to modifications in their execution environment (e.g., services unavailability). Therefore, services that enable adaptation at runtime can be classified as self-adaptive service [4], [7].

Based on presented context, a framework to support the development of Self-adaptive Services-oriented MobApps (Self-MobApps) is proposed in this paper. In short, this framework enables services to be monitored by a supervisor system and adapted at runtime. Such system was designed by our research group in previous work [8] and enables to classify and analyze sensory data to autonomously detect and mitigate faults at runtime (e.g., service unavailability, failures, or high response time). Moreover, this framework aims to support the development of Self-MobApps by means of a dynamic approach for service deployment. In other words, unavailable services can be replaced by a similar one in a transparent way without the perception of their stakeholders (i.e., client applications). For reasons of scope, our framework addresses only services based on JAX-WS (Java API for XML Web Services) [9].

The paper is organized as follows: Section II presents the background and related work; Section III provides a description of our framework; Section IV presents a case study to show the applicability of our framework; and Section V summarizes our findings, conclusions, and perspectives for further research.

## II. BACKGROUND AND RELATED WORK

This section presents the background (i.e., concepts and definitions on self-star software and self-adaptive services) and related work that contributed to the development of our framework.

**Self-star software.** Self-adaptive Software (SaS) has specific characteristics compared to a traditional one because this type of software enables structural, behavioral, or contextual changes at runtime. Among these changes, some of them deal with management of complexity, robustness in handling unexpected conditions (e.g., failure), changing priorities and policies governing the goals, and changing conditions (e.g., execution environment). According to Salehie & Tahvildari [10], “SaS is expected to fulfill its requirements at runtime in response to changes. To achieve this goal, software should have certain characteristics, known as self-\* properties (...). These properties provide some degree of variability, and consequently, help to overcome deviations from expected goals (e.g., reliability)”. To manage the changes at runtime, feedback-loop proposed by IBM [11] has been a good alternative, since all decisions are taken based on a plan established in the data collected from execution environment.

**Self-adaptive service.** Li et al. [12] proposed a self-healing framework for QoS-aware (Quality of Service) web services composition. Self-healing is a self-property that provides special ability to software systems, which can perceive that they are not operating correctly and, without human intervention, make the necessary adjustments to restore them to normal operation. To do so, this framework uses Case-Based Reasoning (CBR) for using previous experiences to understand and solve new problems by means of service similarity. A knowledge-based approach for Service Composition based on self-healing was developed by Angarita et al. [5]. SC is an application composed of a service set that interacts with each other and is invoked on the Web. This type of service can be classified in two categories: (i) static, which represents the aggregation of services taken place at design time; and (ii) dynamic, which enables determining and replacing services at runtime [13].

As related work, Sefid-Dashti & Habibi [14] proposed a mobile SOA Reference Architecture (RA) based on 26 mobile SOA patterns and introduced a new domain specific layer. This RA enables reaction to changes in infrastructure in order to streamline a service, which can be realized by several mobile SOA patterns. A SOA-based platform-specific framework for context-aware MobApps was developed by Daniele et al. [15]. This framework was based on a RA composed of components typically used by MobApps by means of automated design approach. Moreover, the design of context-aware MobApps is platform-independent and can be realized with different specific target implementations. Cherif et al. [6] proposed a Reference Model for specifying Self-adaptive Service-based Applications (ReMoSSA). This model was based on FORMS model [16] and the automate element proposed by IBM [11]. Moreover, costs and efforts of maintenance can be minimized, since this model enables to inspect if the dynamic monitoring and the dynamic adaptation are being considered in the design phase. A declarative approach called SelfMotion (Self-Adaptive Mobile Application) was designed by Cugola et al. [4]. In short, applications based on SelfMotion can be performed by a middleware that enables to create at runtime the best sequence

of abstract actions (i.e., service orchestration) to achieve the goals, mapping them to the concrete actions to execute in accordance with the specified QoS Policy. Such sequences are elaborated by automatic planning techniques, which enable the service changes without perception of their stakeholders. Finally, Nasridinov & Byun [17] proposed a framework named WS-DIRECT (Web Service-Discoverability, REcoverability, Classifiability and Trustworthiness). This framework provides a set of mechanisms to deal with service adaptations at runtime, such as: (i) semantic discovery; (ii) self-healing, i.e., monitoring, diagnosis and repair; (iii) classification of QoS; and (iv) ontology-based security. Li et al. [12] designed a framework QoS-aware web service composition based on self-healing property and CBR. This framework enables to propose solutions for detected problems, to make assumptions and predictions based on previous experiences, and to adapt to changes of the environment.

### III. FRAMEWORK FOR SELF-MOBAPPS

According to Erl [18] and OASIS [19], SOA is composed of three elements: (i) web service provider, which is responsible for providing the services that will be executed; (ii) web service repository, which is responsible for describing, publishing, and finding services; and (iii) web service client, which represents the consumers of such services. Regarding the second element, it represents an XML-based standard called UDDI (Universal Description, Discovery, and Integration). In short, it enables the registry of all web service’s metadata, including a pointer to the WSDL (Web Service Description Language) description of a service, beyond a set of WSDL port type definitions for manipulating and searching such registry. Figure 1 shows the general representation of SOA.

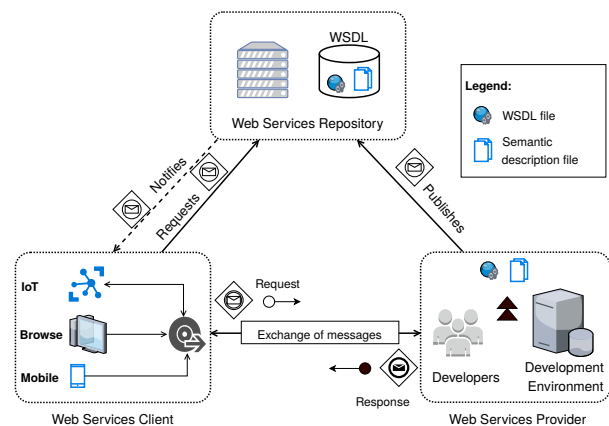


Figure 1. Service-oriented Architecture (Adapted from [19])

Based on these concepts, this section presents a framework to support the development of Self-MobApps. This framework enables services to be monitored at runtime and replaced in case of problems (e.g., service unavailability, failures, or high response time). With regard to the replacement of services, two possibilities are allowed by our framework: design time and runtime. The first uses a list of preferred services defined by

the developers in the design phase of a primary service. The second uses an automatic mechanism of search to find a similar service in the web service repository. Finally, our framework addresses only services based on JAX-WS and simplifies creating and deploying web services and web services clients. Figure 2 shows the general representation of our framework, which is composed of a core for adaptation (dotted line) and four additional modules: development, search, action plan, and deploy. Next, a brief description of this architecture is addressed.

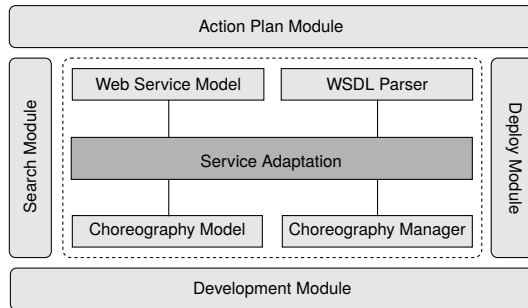


Figure 2. Framework for Self-MobApps

**Development module.** This module provides a set of guidelines for Self-MobApps development. Web services are designed by the developers and registered into environment execution (i.e., Web Service Repository – Figure 1). These services are composed of WSDL and semantic description files. The first one provides to our framework technical description about operations embedded into a service that will be used by our “WSDL Parser” for parameter matching. The second one contains textual description of a service that will be used by “Search Module” when a new service is requested. Moreover, it is noteworthy that when a composition of services (i.e., primary and preferred services) is developed, the software engineer can associate a list of alternative services to each preferred service of this composition. This list must be used when the primary service presents execution problems (e.g., unavailability, poor performance, among others) and a preferential service can assume its role.

**Search module.** This module aims to assist in the search of services in the repository when an adaptation activity is invoked (Figure 1). To do so, three search methods are provided: (i) semantic, which can be defined as a search query by means of contextual meaning for services. This type of search provides more meaningful results by finding the most relevant service in the repository; (ii) technical, which can be specified only with service information in template format, which is converted as input parameters to search in the service repository for matching operation; and (iii) quality, which can be defined as the description or measurement of the overall performance of a service. According to [20], unresolved QoS issues may cause serious problems in relation to execution (e.g., unacceptable levels of performance degradation). Thus, our framework addresses seven quality attributes [20], [21]: availability, accessibility, integrity, performance, reliability,

regulatory, and security. These attributes represent minimal quality of a service. However, other attributes can be found in the literature.

**Action plan module.** This module aims at assisting in the adaptation activity of services providing means to control dynamic behavior, individual reasons, and execution state of each service in relation to the execution environment. To do so, a framework for decision-making in SaS developed by Affonso et al. [8] in previous work was used. In short, such framework is composed of two modules: classification and recommendation. The main purpose of the first module is to present a classification for a data set collected via sensors from execution environment. The second module aims to present a solution set ranked by statistical measures for a problem reported by the classification module.

**Deploy module.** This module aims to support the deployment process for Self-MobApps. In other words, a service can be deployed, undeployed, and redeployed. To do so, we have used jUDDI implementation [22], which is an open source Java implementation of OASIS’s UDDI specification for Web Services. When a service is inserted into repository (Figure 1), its WSDL file is parsed into a structure that aims to provide information about services and the operation of such services. Then, for each operation, its parameters and types are retrieved. In addition, technical service information (e.g., namespace and URL (Uniform Resource Locator) service) must also be obtained.

**Core of adaptation.** This structure can be considered the “heart” of our framework, since enables managing service adaptation at runtime. Basically, this core is organized in five modules: (i) Web Service Model, (ii) WSDL Parser, (iii) Choreography Model, (iv) Choreography Manager, and (v) Service Adaptation. Figure 3 shows the UML model for core of service adaptation. Next, a brief description of each module is reported.

**Web Service Model.** This module contains the model elaborated for the representation of a web service (`parserManager.model` package – Figure 3). From this point onwards, this model may be also referred to as WSMModel (Web Service Model). The `WebService` class is composed of five attributes that describes a service. Each web method of a service has none or many parameters (`Parameter` class).

**WSDL Parser.** This module (`parserManager.parser` package – Figure 3) contains a set of classes responsible for transforming a WSDL document (i.e., file or URL) into WSMModel (`parserManager.model` package). Thus, from such document, relevant information of a service is extracted by means of DOM (Document Object Model) API. The main purpose of this operation is to read an XML file and parse its content into a tree structure, where each node is composed of XML document tags.

**Choreography Model.** This module has the model elaborated for the representation of a choreography that must be executed in the service (`wsManager.model` package – Figure 3). The `WebService` class represents the list of web methods that can be executed by a service and the respective parameters

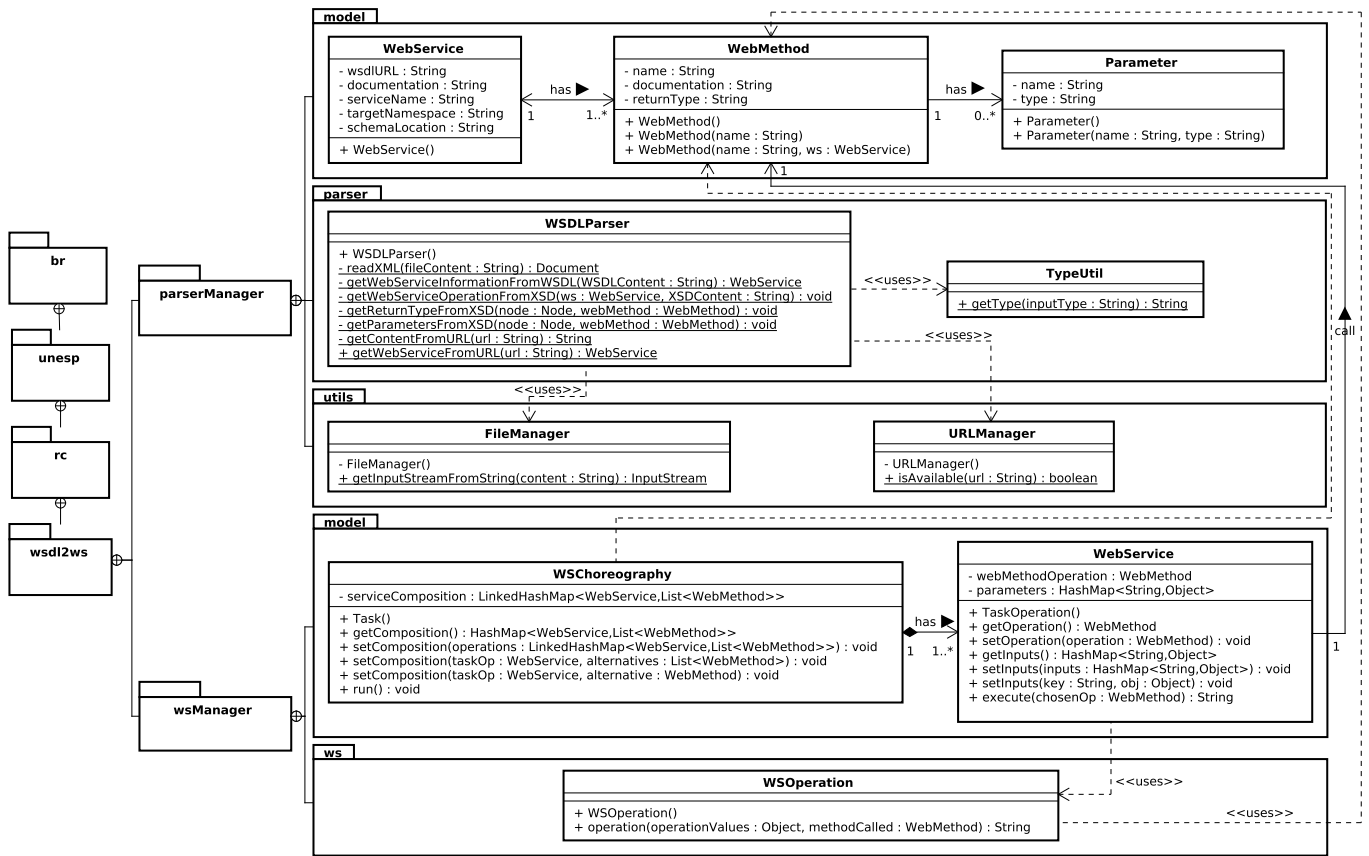


Figure 3. UML model for core of service adaptation

that must be provided to each method at runtime.

**Choreography Manager.** This module has only the `WSOperation` class (`wsManager.ws` package – Figure 3), which is responsible for acting as a client application in the communication process via web service. In short, this class has only the `operation` method that enables executing a web method of a service via parameters.

**Service Adaptation.** This module can be considered as RA4Self-MobApps “orchestrator”, since it performs calls and coordinates all activities of the other modules (i.e., Core of adaptation). In short, this module can be defined as a supervisor system of web services, monitoring their requisitions in the execution environment. To do so, this module implements a well-defined process to adapt a web service at runtime. Finally, this process must be performed automatically by software engineering tools, intending to reduce implementation complexity and minimize uncertainties generation.

#### IV. CASE STUDY

To evaluate the applicability, strengths, and weaknesses of our framework this section presents a case study we have conducted. As subject application of our empirical analysis, we have selected an application addressed to the management of a smart restaurant called App2Rest. This restaurant provides a table set for its customers, which are equipped with a device set that aims to automate the restaurant service from orders to

payments. Next, a brief description of our subject application and the empirical strategies adopted for conducting this case study is presented.

**Subject Application.** The App2Rest was organized in two layers: (i) Devices, which represent the means of access to the clients (i.e., front-end side) of this application, which can use mobile devices equipped with the Android operating system (i.e., smartphones and tablets) and personal computers via the Web; and (ii) Web Server, which represents an application composed of a service set (i.e., back-end side). Regarding the exchange of information between these two layers, a subsystem was developed to serialize application models via JSON (Java Script Object Notation) and facilitate the exchange of information between these layers (i.e., client and server). For instance, when a menu is requested by the client side, a query is performed on the server side and a model composed of several classes (i.e., data) is instantiated. Next, such model is serialized to a String (i.e., JSON format) and transferred to the client side. The inverse process is also considered.

**Empirical research strategy.** Figure 4 illustrates the generic structure of our application, which is organized in a client-server architecture composed of two layers. Regarding the operation of this application, all requests of the “Devices” layer (i.e., client side) are sent to the “Web Server” layer, whose purpose is to intermediate the communication (i.e., message exchange) between customer or restaurant devices



and requested web services of such application. The web services contained in the App2Rest can be classified in two levels of complexity: (i) simple service, which represents the encapsulation of a functionality to be made available to its customers (i.e., “Devices” layer); and (ii) service composed of services, which represents the encapsulation of more than one functionality of the restaurant application to be made available to its customers. This service type executes a call set to other services by means of a choreography (i.e., action sequences and conditions) for its functionality to be fulfilled. For instance, the authentication of a customer in the restaurant application is a simple service. Customers request orders (i.e., items for an order) can be characterized as a composite service.

Regardless of the complexity level, the App2Rest has a “Service Monitor” component in each service to monitor its execution status (Figure 4). In a first analysis, this monitor will determine if the web service that is being requested is available to be accessed by a customer application (i.e., “Devices” layer). In a more refined analysis, the quality of such services can also be assessed. In both analyzes, one service can be replaced by another equivalent at runtime without client’s perception. In this sense, to present the details of this last phase, the `WS_01` service will be considered, which enables customers to make orders for the restaurant. This service is designed based on three services (`WS_A`, `WS_B`, and `WS_C`) by means of a service composition. The first service (`WS_A`) aims to authenticate the customer in restaurant application via “E-MAIL” or “GOOGLE SIGN IN”. Once authenticated, the App2Rest displays the main screen and the meal of the day is suggested. Next, the customer can accept this suggestion or select another dish via left side menu. To select another dish, the customer must access the `Menu` option in the side menu so that the dishes and all items available in the restaurant are displayed. The restaurant menu is performed by the second service (`WS_B`), which enables select some items by category. When selecting a dish (add button), an order pad is initialized and many items can be added to it (third service – `WS_C`). Next, a brief description of each phase is addressed.

In the “design” phase, developers must select the preferred services that will compose the primary service (e.g., `WS_01`). These services are inserted into a dynamic list called “Preferred Services” (i.e., `WS_A`, `WS_B` and `WS_C`, and the position of each service in such list is equivalent to its execution order. More complex choreographies require an external file to indicate the order of execution of each service. In addition, for each preferred service there will be a list of “Alternative Services”. For instance, for the preferred service `WS_A`, the alternative services `WS_A1`, `WS_A2` and `WS_A3` were selected.

In the runtime phase, primary services are monitored by the “Service Monitor” component. For instance, when the monitoring activity detects problems in one of the preferred services that compose a primary service, the list of alternative services is consulted. Thus, three situations can be observed: (i) the list has an alternate service that can replace a preferred one; (ii) the alternative service list may be empty, i.e., the developer has not prepared alternative services at the design

phase; or (iii) the list may be empty by attempts, i.e., services have become unavailable over time. In the last two situations, the “Service Monitor” component should launch a search in the “Web Service Repository” to find a service of greater similarity. Such search can return a service set ranked by statistical measures in relation to the satisfaction of the parameters initially provided.

## V. CONCLUSIONS AND FUTURE WORK

This paper presented a framework that intends to support the Self-MobApps development. Such framework uses a dynamic approach for service deployment. In other words, unavailable services can be replaced by a similar one in a transparent way without the perception of their stakeholders (i.e., client applications). Moreover, our framework can classify and analyze sensory data to autonomously detect and mitigate faults at runtime for a simple service or a composition of services [8]. As reported in Section I, our framework will address only services based on JAX-WS [9]. Based on this scenario, the main contributions of this article are: (i) for the SaS area by providing a means (i.e., framework) that facilitates the development of Self-MobApps; (ii) for the Service Computing area, since we have proposed a framework that enables the development of Self-MobApps or SOA-based systems by means of a dynamic approach for service deployment; and (iii) for both areas, since we have developed a means to replace unavailable services at runtime without the perception of their stakeholders. In this sense, the proposed mechanism based on dynamic deploy for substitution of services (i.e., preferred and alternative) must be highlighted. The previous selection of services can further minimize the possible impacts (i.e., computational cost and time) caused by the deploy of services.

Regarding future work, at least two activities are intended: (i) conduction of more case studies or proof of concepts intending to completely evaluate our framework, including different software domains; and (ii) use of this framework in the industry, since it is intended to evaluate its behavior when it is applied in larger real environment of development and execution. Therefore, based on the content exposed in this paper, a positive research scenario can be idealized, intending to have this framework become an effective contribution to the software engineering and SaS communities.

## ACKNOWLEDGMENT

This research is supported by PROPE/UNESP and Brazilian funding agencies (CNPq and CAPES).

## REFERENCES

- [1] J. Aghav and N. Sharma, “A software architecture for provisioning of mobile services: An osgi implementation,” in *MEMSTECH '11*. Polyana, Ukraine: IEEE, May 2011, pp. 24–27.
- [2] F. Gonçalves, C. E. T. Oliveira, I. Silva, and L. Moura, “An architectural model for applications based on mobile services,” in *ICCGI '07*. Guadeloupe City, Guadeloupe: IEEE, March 2007, pp. 1–6.
- [3] F. Gonçalves, C. E. T. Oliveira, I. Silva, L. Moura, and F. Franca, “A software architecture for the provisioning of mobile services in peer-to-peer environments,” in *ICIW '07*. Morne, Mauritius: IEEE, May 2007, pp. 1–6.
- [4] G. Cugola, C. Ghezzi, L. S. Pinto, and G. Tamburrelli, “Selfmotion: A declarative approach for adaptive service-oriented mobile applications,” *Journal of Systems and Software*, vol. 92, pp. 32 – 44, 2014.

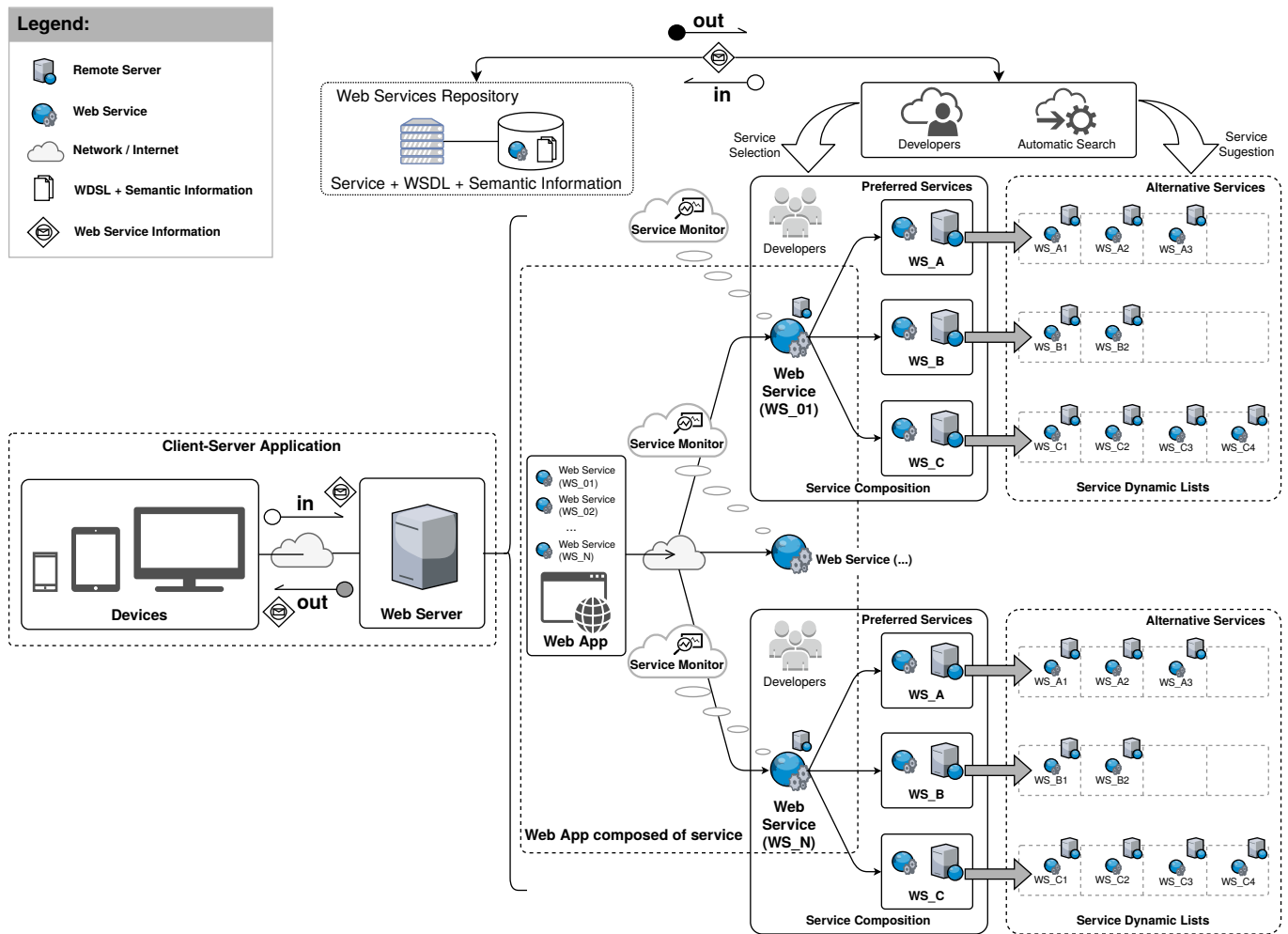


Figure 4. Generic Representation of Service for Restaurant Application

- [5] R. Angarita, M. Rukoz, M. Manouvrier, and Y. Cardinale, "A knowledge-based approach for self-healing service-oriented applications," in *MEDES '16*, ser. MEDES. Biarritz, France: ACM, 2016, pp. 1–8.
- [6] S. Cherif, R. Ben Djema, and I. Amous, "Remossa: Reference model for specification of self-adaptive service-oriented-architecture," in *AISC '14*, B. Catania, T. Cerquitelli, S. Chiusano, G. Guerrini, M. Kämpf, A. Kemper, B. Novikov, T. Palpanas, J. Pokorný, and A. Vakali, Eds. Cham: Springer International Publishing, 2014, pp. 121–128.
- [7] D. Menasce, H. Goma, s. Malek, and J. Sousa, "Sassy: A framework for self-architecting service-oriented systems," *IEEE Software*, vol. 28, no. 6, pp. 78–85, Nov 2011.
- [8] F. J. Affonso, G. Leite, R. A. P. Oliveira, and E. Y. Nakagawa, "A framework based on learning techniques for decision-making in self-adaptive software," in *SEKE '15*. Pittsburgh, USA: Knowledge Systems Institute, 2015, pp. 24–29.
- [9] Oracle, "Java platform, enterprise edition: The java ee tutorial," [On-line], 2018, available: <https://goo.gl/qGeGwd>, Accessed on February 28, 2018.
- [10] M. Salehie and L. Tahvildari, "Self-adaptive software: Landscape and research challenges," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 4, no. 2, pp. 1–42, 2009.
- [11] IBM, "An architectural blueprint for autonomic computing," [On-line], 2005, available: <https://goo.gl/wawGvi>, Third Edition, Accessed on February 28, 2018.
- [12] G. Li, L. Liao, D. Song, J. Wang, F. Sun, and G. Liang, "A self-healing framework for qos-aware web service composition via case-based reasoning," in *APWeb '13*, Y. Ishikawa, J. Li, W. Wang, R. Zhang, and W. Zhang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 654–661.
- [13] Q. Z. Sheng, X. Qiao, A. V. Vasilakos, C. Szabo, S. Bourme, and X. Xu, "Web services composition: A decade's overview," *Information Sciences*, vol. 280, pp. 218 – 238, 2014.
- [14] B. Sefid-Dashti and J. Habibi, "A reference architecture for mobile soa," *Systems Engineering*, vol. 17, no. 4, pp. 407–425, 2014.
- [15] L. M. Daniele, E. Silva, L. F. Pires, and M. van Sinderen, "A soa-based platform-specific framework for context-aware mobile applications," in *IFIP/IWEI '09*, R. Poler, M. van Sinderen, and R. Sanchis, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 25–37.
- [16] D. Weyns, S. Malek, and J. Andersson, "Forms: a formal reference model for self-adaptation," in *ICAC '10*. New York, NY, USA: ACM, 2010, pp. 205–214.
- [17] A. Nasridinov and J. Byun, "Ws-direct: Web service—discoverability, recoverability, classifiability and trustworthiness," in *CUTE '12*, Y.-H. Han, D.-S. Park, W. Jia, and S.-S. Yeo, Eds. Dordrecht: Springer Netherlands, 2013, pp. 879–887.
- [18] T. Erl, *Service-Oriented Architecture: Analysis and Design for Services and Microservices*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2016.
- [19] OASIS, "Reference architecture foundation for service oriented architecture version 1.0," OASIS Committee Specification 01, [On-line], 2012, available: <https://goo.gl/m2pEm4>, Accessed on February 28, 2018.
- [20] A. Mani and A. Nagarajan, "Understanding quality of service for web services, improving the performance of your web services," [On-line], 2002, available: <https://goo.gl/TqkVtX>, Published on January 01, 2002, Accessed on February 28, 2018.
- [21] A. Al-Moayed and B. Hollunder, "Quality of service attributes in web services," in *ICSEA '10*, August 2010, pp. 367–372.
- [22] jUDDI, "An open source implementation of oasis's uddi v3 specification," [On-line], 2018, available: <https://juddi.apache.org/>, Accessed on February 28, 2018.

# Modeling of Interlocking Systems based on Patterns

Wang Yan, Zhong Wen, Xiaohong Chen\*, Dehui Du

Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai

\*corresponding author, xhchen@sei.ecnu.edu.cn

**Abstract**—The equipment faults of the interlocking system in rail transit system has occurred frequently, and these faults can cause serious accidents. The modeling of the interlocking system must aim at the stochasticity and real-time characteristics of equipment faults. Therefore, this paper proposes to model the interlocking system using stochastic hybrid automata. In order to improve model efficiency, we try to extract the pattern of the interlocking system model, and reuses these patterns in system modeling. The main contributions include: (1) Based on the business analysis of the interlocking system, 12 model patterns of the interlocking system are extracted; and (2) the modeling process of the interlocking system based on patterns reuse is given to guide system modeling. Finally, a case study is presented to illustrate the feasibility and effectiveness of our approach.

**Keywords**—Interlocking system; Pattern reuse; Stochastic hybrid automata; Modeling; UPPAAL-SMC

## I. INTRODUCTION

The safety of rail transit systems is of great importance. However, in recent years, railway accidents happen from time to time. For example, July 23, 2011 Yong Wen line "7.23" major railway traffic accident. This is a serious accident caused by the design flaws of the equipment in the control center, inadequate checks on the road, and ineffective emergency response after equipment failure which caused by lightning. The accident led to a rear-end train collision, and caused 40 deaths, 172 injuries, and the direct economic losses of 19371.65 million RMB [1].

The interlocking system is one of the core subsystems in the rail transit system [2]. It has SIL4-level safety requirements, complex logic, and high requirements for real-time performance [3] [4]. It can endanger the safety of the vehicle when a failure occurs. Therefore, modeling and analyzing the interlocking system become very important and is one of the key methods to ensure system safety.

The interlocking system is composed by various equipment. The main cause of most accidents in interlocking system is equipment failure which is stochastic. For example, lightning strike causes the short circuit of track and leads to an accident. Therefore, we should consider the stochastic characteristic of equipment faults when modeling. Stochastic Hybrid Automata (SHA) [5] offers stochasticity and time modeling, and has been widely used in various fields such as electromechanical systems, computer simulation, automata and so on [6]. In this paper, we take the time constraints into consideration and propose to use SHA to construct the system model for verification. Considering

the modeling and verification, we choose the platform UPPAAL-SMC [7] for modeling and analysis.

In order to facilitate the construction of rail transit system model, we extract the patterns of the system model for the interlocking system. Using these patterns, one only needs to fill in the appropriate parameters to customize the specific system model. The final customized system model is verified.

The rest of this paper is organized as follows. Section II presents the framework of our approach; Section III gives the method for constructing the system model patterns and Section IV clarifies pattern based system model generation method; a case study is given in section V and the related work follows in section VI. Finally, Section VII concludes the paper.

## II. FRAMEWORK OF OUR APPROACH

The purpose of railway interlocking system is to control points and signal lights to prevent trains from collisions and derailments [8], while allowing its movement. The processing flow of the interlocking system is on <https://github.com/wymgal/IS.git> (for simplicity, we will refer it as our website in the following). Generally speaking, the physical domain of an interlocking system consists of 5 entities, *tracks*, *points*, *signal lights*, *routes*, and *interlocking table*. The *tracks* are divided into sections, and each section is associated with a circuit for detecting whether it is occupied or not. Track sections are joined by *points* which can guide trains into different directions depending on the positions of the points. A point can be in position normal or reverse, as well as unlocked to show that the tracks are unconnected at the crossing. *Signal lights* are placed between track sections and use red or green color to indicate proceed or stop signal respectively. *Routes* are established for authorizing a train to enter. It is often defined by *interlocking table*, which includes the conditions for locking and releasing the train route and for when the entry signals of the route is set to show proceed or stop signal.

According to the above descriptions, we get a context diagram of interlocking system, as shown in Figure 1. The system contains six entities, *Train*, *Signal Light*, *Point*, *Track*, *Interlocking Table* and the *Controller*. Except *Controller*, we name the other five entities the *environment* of the interlocking system. The *environment* interacts with the *Controller*. The interactions are the shared message between the *Controller* and the *environment* entities.

Based on the context diagram, we give a framework of our

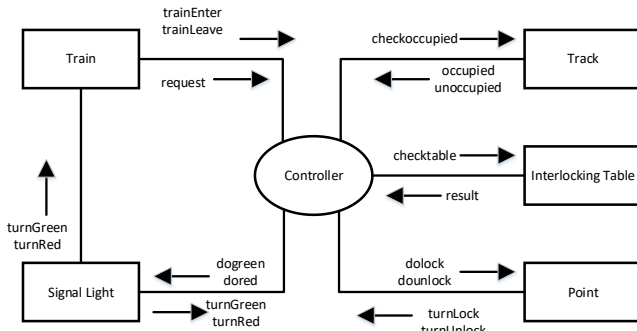


Fig.1 Context diagram of the interlocking system

approach as shown in Figure 2. Firstly, we extract the system pattern from the domain knowledge of the interlocking system. The system pattern consists of two parts: the environment patterns and the controller patterns. Based on the extracted system model, a system model is generated in combination with a model parameter table. The generated system model is simulated on the UPPAAL-SMC platform to verify the properties of the system.

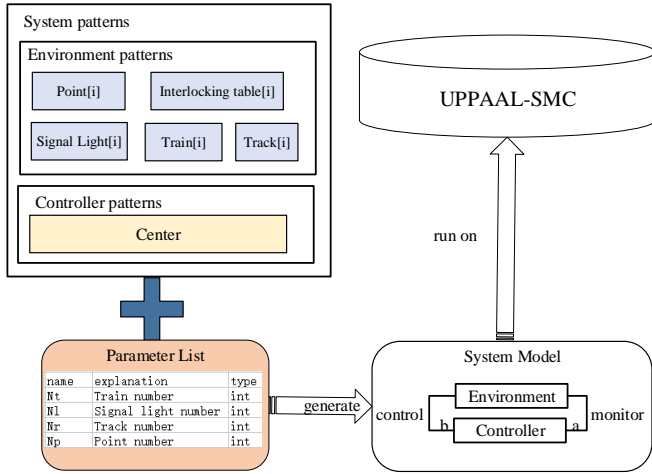


Fig.2 Framework of our approach

The environment patterns include the models of each environmental entity of the interlocking system, such as *Train*, *Point*, *Interlocking Table*, *Signal Light*, and *Track*. The controller patterns are also defined. Model parameter list is the key to realize the reusability which is given by domain experts. Using parameters in the parameter list, the extracted system patterns can be instantiated to generate a specific system model.

### III. INTERLOCKING SYSTEM MODEL PATTERNS

#### A. Constructing process

Firstly, we give a 3-step process to obtain the SHA of each system entity. The three steps are constructing the basic automata, modeling faults, and adding time constraints.

##### Step 1: Constructing the basic automata

The process description related to the entity is found according to the system processing flow and system context diagram, including all the behaviors related to the entity.

We give a guideline to get a basic automata of each entity. Each time the entity sends or receives a message (action), the entity's automata moves from one state to another state.

Therefore, an action of each entity is transformed into a state and a transition in a basic automata. The transition is an action.

##### Step 2: Modeling faults

In the entities, it is possible that the occurrence of an abnormal event can lead to a fault, and an abnormal event can be represented by the probability. Therefore, we find all the abnormal events, and use stochastic probability events to express them. Different events are performed with different probabilities. Based on the basic automata, stochastic probability events are added to model faults.

##### Step 3: Adding time constraints

This step is to add time constraints on the results of step 2. Firstly, the time constraints of entities are extracted from the domain experts and expressed as  $\langle message1, message2, \leq n \text{ time unit} \rangle$ . Then, the corresponding clock variable  $x$  are defined. The representation of the clock constraint in the automata is the time between  $message1$  and  $message2$ , that is, in the automata, the initial value of clock variable  $x$  on the "update" of  $message1$  is 0, and the inequality  $x \leq n$  of the clock variable is defined in the "guard" of the  $message2$ .

#### B. Environment entity patterns

##### a) Train Pattern

We obtain the processing flow of the train entity from the system processing flow. When the train enters the track, it sends a request signal to the *controller* and waits for the signals of signal lights. If the train is accepted within the stipulated time, it enters the track. If rejected, it stops and waits. According to the guideline, a basic automata of the train entity is obtained.

A fault may occur during the train running, that is, between sending "trainEnter" message and sending "trainLeave" message. Add an error state to the automata. Message sent from "trainEnter" is transferred to the error state with the probability of  $m\%$  and transferred to the starting point of "trainLeave" message with the probability of  $n\%$ , where,  $m + n = 100$ ,  $m$  and  $n$  are real numbers, and are decided by domain experts.

The time constraints are obtained from domain experts as follows. The not-all-lights-green signal "notallgreen" is received within specified time. The all-lights-green "allgreen" is received within specified time. The clock variable  $x$  is defined to indicate the waiting time for the signal light, that is, the time from sending "request" message to receiving "notallgreen" message or "allgreen" message. Therefore, the initial value of  $x$  on the "update" of the "request" transition is 0, and the inequality " $x < z$ " ( $z$  is a constant) is used as the "guard" of transition "notallgreen" or "allgreen". Therefore, the SHA pattern of the train is obtained, as shown in Figure 3(a).

##### b) Signal Light Pattern

We divide the activities involved in *Signal Light* into two parts. One is *SSignalLight*, which is responsible for setting the status of the signal light. The other part is *RSignalLight*, which is responsible for inquiring the status of the signal light.

##### Signal Light = SSignalLight||RSignalLight

**SSignalLight:** We get the processing flow of the *SSignalLight* from the system processing flow. The initial state of the *Signal Light* is red. After receiving the commands of the *controller*, the signal light changes its state. According to the guideline, the

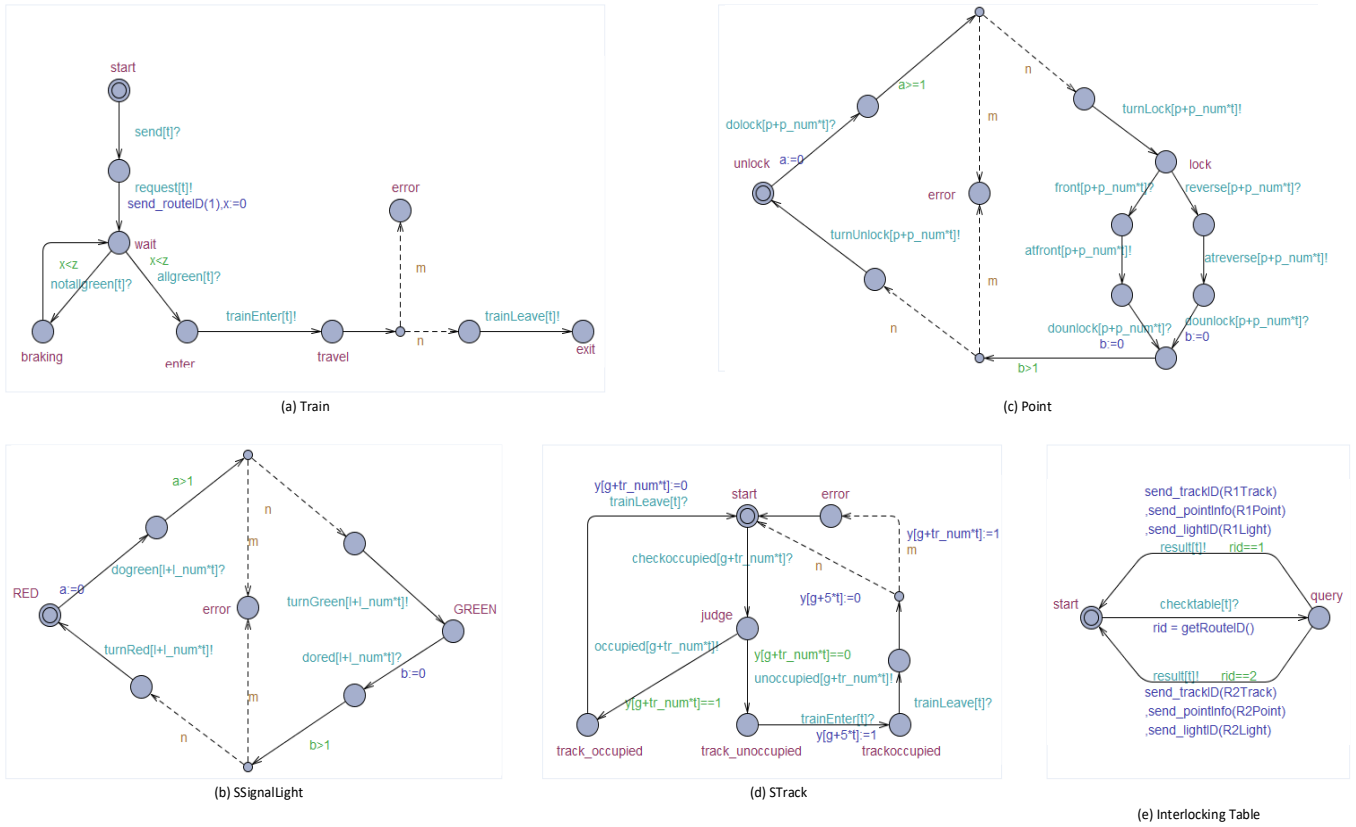


Fig.3 SHA patterns of environment entities

basic automata of the *SSignalLight* is obtained (see our website). A fault may occur during the change of the signal lights' states, that is, between receiving "dogreen" message and sending "turnGreen" message or between receiving "dored" message and sending "turnRed" message. Add an error state to the automata. Message sent from "dogreen" is transferred to the error state with the probability of  $m\%$  and transferred to the starting point of "turnGreen" message with the probability of  $n\%$ . Similarly, we can model faults in the situation where the signal light changes from green to red.

The time constraints are obtained as follows. There is a certain delay in the state change of Signal Light. A local clock  $a$  is given to indicate the delay time of signal light changing from red to green, and a clock  $b$  indicates the delay time of signal changing from green to red. The initial value on the "update" of the "dogreen" transition is 0, and the inequality " $a > 1$ " is used as the "guard" of the transition. Similarly, we can define the clock  $b$  in this automata. The SHA of the *SSignalLight* is thus obtained, as shown in Figure 3(b).

**RSignalLight:** In order to get a set of single light states, we first model one single light. According to the system processing flow, we can get the processing flow of the *RSignalLight*. According to the guideline, we build an automata for one signal light as shown in Figure 4.

For a group of lights, the modeling process is based on the situation of one signal light. Add the corresponding different signal lights' green identifier  $isLightGreen$  and the red light identifier  $isLightRed$ . For  $N$  signal lights, there should be  $n$

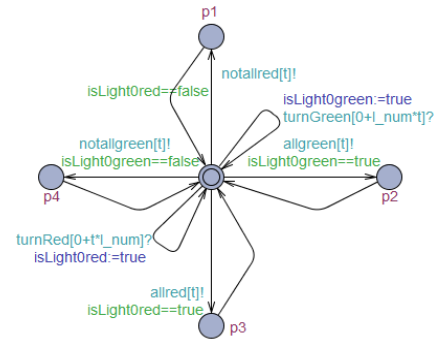


Fig.4 SHA pattern of RSignalLight for querying single signal light

$isLightGreen[0,1,\dots,n-1]$  and  $isLightRed[0,1,\dots,n-1]$ . Add  $n$  transitions with the message of "turnGreen[ $n-1$ ]?" from the initial state to itself. Make  $isLightRed[n-1]=1$ . The judgement condition of transition "allgreen!" is:  $isLightGreen[0]==1 \& \& isLightGreen[1]==1 \& \dots \& isLightGreen[n-1]==1$ . Similarly, we change the judgement condition of transition "notallGreen", transition "allred" and transition "notallred". The automata for a group of signal lights is in our website. We do not consider the error situation and the time constraints of *RSignalLight*.

#### c) Point Pattern

We can get the processing flow of the point entity from the system processing flow. The initial state of the *Point* is unlocked. When the point receives commands from the *controller*, the point changes its state. There is a certain delay in the state change of the point entity. According to the guideline, a basic automata of the point is obtained as shown in our website.

A fault may occur during the point changing from the locked state to the unlocked state or from the unlocked state to the locked state, that is, between receiving "dolock" message and sending "turnLock" message or between receiving "dounlock" message and sending "turnUnlock" message. Add an error state to the basic automata. Message sent from "dolock" is transferred to the error state with the probability of  $m\%$  and transferred to the starting point of "turnLock" message with the probability of  $n\%$ . Similarly, we can model faults in the situation where the point changing from locked state to the unlocked state.

The time constraint for point entity is that there is a certain delay in the state change of the point. The local clock  $a$  is defined to indicate that the delay time of the point changing from unlocked state to locked state, and the local clock  $b$  indicates the delay time of the point changing from locked state to unlocked state. That is, clock  $a$  is the time from receiving "dolock" message to the next state  $lock$ . Therefore, the initial value of  $a$  on the "update" of the "dolock" transition is 0, and the inequality " $a > 1$ " is used as the "guard" of the transition. Similarly, we can define the clock  $b$ . The SHA of the point is thus obtained, as shown in Figure 3 (c).

#### d) Track Pattern

We divide activities involved in *Track* into two parts. One is *STrack*, which is responsible for setting the status of the *Track*. The other part is *RTrack*, which is responsible for inquiring the status of the *Track*. So we get: **Track=STrack|RTrack**

**STrack:** The processing flow of the *STrack* is extracted from the system processing flow. After receiving commands from the *controller*, the track check whether it is occupied and return the results to the *controller*. According to the guideline, a basic automata is obtained ( see our website).

A fault may occur during the process of setting the track state, that is, between receiving "trainEnter" message and receiving "trainLeave" message. Add an error state to the automata. Message sent from "trainEnter" is transferred to the error state with the probability of  $m\%$  and transferred to the starting point of "trainLeave" message with the probability of  $n\%$ .

The method of adding transition with a probability is similar with the case of the train entity. Without taking the time constraints of track entity into account, we do not extract time constraints. The SHA of *STrack* is obtained by the above steps, as shown in Figure 3 (d).

**RTrack:** It is finished in two steps. One is for one track. We build an SHA model for one track, as shown in our website. The other one is for a group of tracks. The processing is similar with *RSignalLight*. We only need to change the judgment conditions and transitional messages.

#### e) Interlocking Table Pattern

According to the system processing flow, the processing flow of the Interlocking Table entity is obtained. After receiving the query command from the controller, the interlocking table queries the related information and returns results. According to the guideline, the automata of the interlocking table is obtained as shown in Figure 3(e). We do not consider the error situation and time constraints of the interlocking table entity.

### C. Controller Pattern

The controller is divided into two parts. One is the *Center* which is responsible for controlling all the tracks, points, signal lights, trains and the interlocking table. The other part called *Submodules*, which is in charge of controlling each track, point, signal light and train respectively. So we define:

**Controller=Center|Submodules**

**Submodules=CTrack|CPoint|CSignalLight|Dispatcher**

**Center:** The processing scenario of the *Center* is extracted as expressed in our website. According to the guideline, the basic automata of the *Center* is obtained as shown in Figure 5.

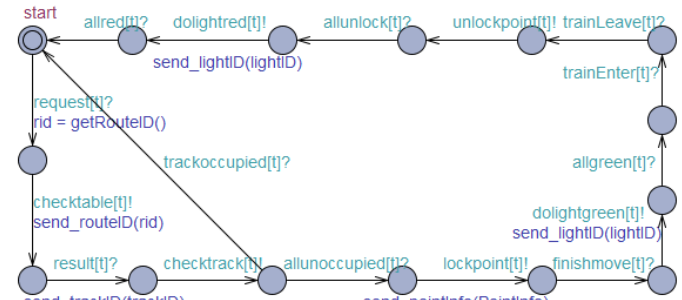


Fig.5 A Center Pattern

In the global declaration, we define 8 functions required by *Center* to interact with each entity. They are `send_routeID`, `getRouteID`, `send_trackID`, `getTrackID`, `send_pointInfo`, `getPointInfo`, `send_lightID`, and `getLightID`. Their functions are the meaning of their name. Due to limited space, the exact definition is shown in our website.

**CTrack, CPoint, and CSignalLight:** The *CTrack* is responsible for sending "checkoccupied" message to each track, and checking the occupancy situation of each track. After receiving the instruction of *Center*, the *CPoint* sends "dolock" and "dounlock" message to each point. After receiving instruction from *Center*, the *CSignalLight* sends "dogreen" and "dored" message to each signal light. The construction process of *CTrack*, *CPoint*, *CSignalLight* are similar to the construction process of *RSignalLight*. Similarly, we can get the *CTrack*, *CPoint* and *CSignalLight* ( see our website).

**Dispatcher:** It is responsible for sending dispatching instructions to control different trains entering the track at different time. How many trains to be sent is decided by detailed number. So we cannot give a graph pattern here. But the basic sentence can be recording as: for each train  $i$ , we add a state with the message of "send[i-1]!" and a transition with the message of "trainEnter[i-1]?".

## IV. A SYSTEM MODELING METHOD BASED ON PATTERNS

We give a 5-step process to model the system.

**Step 1: Declaring all the models in the system** Through analysis, we define that the system is composed by 12 models, so we make the following declaration:

*system Train, Track, Light, Point, Center, Dispatcher, CSignalLight, CPoint, CTrack, RSignalLight, RTrack, InterlockTable;*

**Step 2: Setting model instantiations** Get the number of trains, signals, tracks and points from the interlocking table. Suppose

$Nt$ ,  $Nl$ ,  $Nr$  and  $Np$  are the number of trains, signals, tracks and points respectively. The models could be instantiated by the following declarations:

```
const int TRAINS=Nt; typedef int[0,TRAINS-1] train_t;
const int LIGHTS =Nl; typedef int[0,LIGHTS-1] light_l;
const int TRACKS = Nr; typedef int[0,TRACKS-1] track_t;
const int POINTS=Np; typedef int[0,POINTS-1] point_p;
```

**Step 3: Reusing patterns** The parameters in each pattern can be modified according to specific situations. According to the number of entities, the *RSignalLight* and *RTrack* model can be instantiated. Reuse the *Center* pattern, create the SHA model of the *Center*, and instantiate the SHA models of the *Controller* submodules.

**Step 4: Defining the system interactions** The interactions are achieved by the communication between the entities in the context diagram. So defining the system interactions is to define all the messages in the communication between entities. Define all messages in the global declaration. For example, Chan green[ $Nr*Nl$ ]. According to the interlocking table, we can know that one train needs  $Nl$  signal lights, that is, needs  $Nl$  green signals. So for  $Nt$  trains, there should be  $Nt*Nl$  green signals. The other messages are declared in our website.

**Step 5: Declaring system variables** The global variables in the system are actually shared information between models. They should include the track occupancy identifier, and the number of instantiations of each entity in the global declaration. In addition, the variables used by the functions of *Center* should be declared too. The exact declaration is as follows.

```
int y[Nt*Nr]={0,0,...,0} // the track occupancy identifier
const int l_num=Nl, p_num=Np, tr_num=Nr, t_num=Nt;
// the number of instantiations of each entity
int route_id, trackID[Nr], PointInfo[Np][Np], lightID[Nl];
// the variables used by the functions
```

## V. CASE STUDY

In this paper, we use a case which interlocking table is shown in Table I. There are two routes, *Route1* and *Route2*, 5 signal lights  $S1, S2, S4, S5$ , and  $S7$  on the *Route1*, and 5 lights  $S1, S2, S3, S6$ , and  $S7$  on the *Route2*. Two points  $SW1$  and  $SW2$ , and five tracks  $T1, T2, T3, T4$ , and  $T5$  are included.

### A. Defining the system

According to the process, we declare the 12 models as listed in Section IV. From Table I, we get the numbers of the trains, signal lights, tracks and points, which are 2, 5, 5, 2 respectively. It means  $Nt=2, Nl=5, Nr=5, Np=2$ . Put them into the model declaration to declare the models of the system:

```
const int TRAINS=2; const int LIGHTS =5;
const int TRACKS = 5; const int POINTS=2;
```

Reuse the patterns of *Train*, *Signal Light*, *Point*, *Track* and *Interlocking Table*, and modify *RSignalLight* and *RTrack*

according to their numbers 5 and 5. We modify the number of transitions and judging conditions, and get these two models. *RSignalLight* model and *RTrack* model are in our website. Finally the *Controller* model is constructed. We reuse the *Center* pattern, and build *CTrack*, *CPoint*, *CSignalLight* according to their numbers. Reuse the *Dispatcher*. We add two clock variables to *Dispatcher*, clock variable  $m$  starts timing when the *train0* enter the track, and the clock variable  $n$  starts timing when the *train1* enter the track. The *CTrack* model is shown in our website, and the rest of the *Controller* submodules are displayed in our website. After this, put  $Nt=2, Nl=5, Nr=5, Np=2$  into the global declaration as follows. Finally the system is built.

```
int y[10] = {0,0,0,0,0,0,0,0,0,0};
const int l_num=5, p_num=2, tr_num=5, t_num=2;
int route_id, trackID[5], PointInfo[2][2], lightID[5];
```

### B. Simulation and verification

UPPAAL uses BNF syntax to describe the security requirements of the system, and the modeler can verify the related properties of the system according to the different design requirements. This paper only considers the part design requirements, including the system model is not deadlock (1), the *Signal Light* model can enter the green light state (2), and the *Monitor* model can detect the error of the system into the warning state (3). They are represented as follows:

```
A[] not deadlock (1)
E<> Light.GREEN (2)
E<> Monitor.warning (3)
```

The simulation model of the system in the UPPAAL platform, after repeated simulation and observation, meet the above three design requirements: the preliminary determination of each state of the model is deadlock and reachable; signal light can enter the green state; the monitor can determine the corresponding error and enter warning state.

As a result, the model meets the requirements of the system, the expected security requirements, and ensures the security and correctness of the model. The following results are obtained, as shown in Figure 6.

## VI. RELATED WORK

In order to ensure the correctness and safety of the system, there are many works for modeling the real time and fault stochastic characteristics. Formal methods are widely used in modelling and analysis [10,11], such as Timed Automata [12], Petri net [13], Z language [14] and so on. For example, Wang uses the time automata theory to model and verify the railway station signal interlocking system and the interlocking route control process [15]. Hei et al. use Petri net to model and verify the distributed control interlocking system [16]. Tiejia Wang describes the security requirements of computer interlocking software in Z language [17].

TABLE I. INTERLOCKING TABLE

Route			Signals		Points			Track
ID	From	To	Green	Red	Open		Close	
					Up	Down		
R1	S1	S7	S1,S2,S4,S5,S7	S3,S6	SW1,SW2		T1,T2,T3,T5,T6	
R2	S1	S7	S1,S2,S3,S6,S7	S4,S5	SW1,SW2		T1,T2,T4,T5,T6	

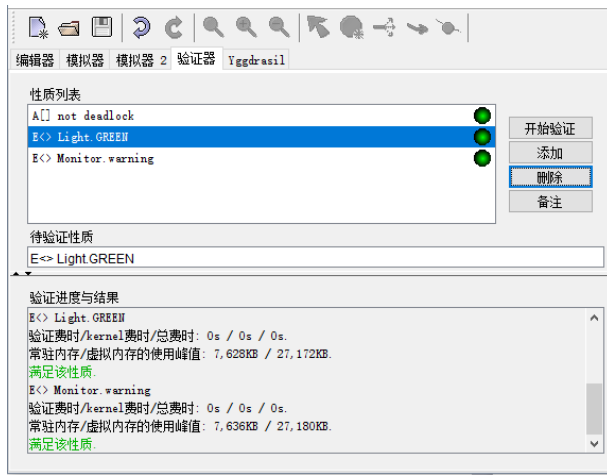


Fig.6 Verification results

However, these formal methods have shortcomings for modeling the interlocking system. Although timed automata consider the time requirement of the system with real-time characteristics, it has no stochastic and cannot model the complex system's stochastic faults. The Petri net and Z languages do not consider the time and stochasticity of the system when modeling. Compared with these methods, our approach of using stochastic hybrid automata can consider the stochastic and real time characteristics of interlocking system, which is more suitable for interlocking system modeling.

Another related work is pattern based modeling. Although the UPPAAL-SMC provides the train gate example, but it does not have patterns. There are many efforts in pattern based modeling. For example, Wu et al. propose a traffic pattern modeling approach for the urban intersection[18]. Zhang et al. propose an observer-pattern modeling method to eliminate the time-variance effect for two-stage boost inverter [19]. However, these pattern-based system modeling work is rarely related to the interlocking system. In addition, many modeling languages used in them do not pay particular attention to time constraints and stochasticity.

## VII. CONCLUSION AND FUTURE WORK

As one of the core systems of rail transportation system, the interlocking system ensures the safety of trains. Based on the SHA, this paper presents the modeling of the interlocking system using patterns. The modeled system could be analyzed using simulation and verification technology. The main contributions of this paper include:

- (1) The 12 model patterns for interlocking systems are extracted covering 6 entities consisting of train, signal light, point, track, interlocking table and controller;
- (2) An approach for reusing these patterns to construct an exact interlocking system model is proposed. Using this approach, novices of SHA could be quickly build a system for further analysis.

The next step work is to consider more fault types and apply this model to accident prediction.

## ACKNOWLEDGMENTS

This work was supported by the Natural Science Foundation of China (No. 61472140) and Defense Industrial Technology Development Program (JCKY2016212B004-2).

## REFERENCES

- [1] "7.23" Yong Wen line special major railway traffic accident investigation report [EB/OL].(2011-12-25) [2013-02-10]. [http://www.gov.cn/gzdt/2011-12/29/content\\_2032986.html](http://www.gov.cn/gzdt/2011-12/29/content_2032986.html)
- [2] Baofeng Xie. Status and development of computer interlocking system of station[J]. Transportation system engineering and information, 2004, 4(4):86-90.
- [3] EN501 28C . Railway application-SoRware for railway control and protectionsystem[J]. 2000.
- [4] Krishna,C . M . Real Time Systems[M] // Real-Time Systems. McGraw-Hill Higher Education,1 996 : 3-5.
- [5] Bortolussi L, Policriti A. Stochastic Programs and Hybrid Automata for (Biological) Modeling[C]// Conference on Computability in Europe: Mathematical Theory and Computational Practice. Springer-Verlag, 2009:37-48.
- [6] Bemporad A, Cairano S D. Optimal Control of Discrete Hybrid Stochastic Automata[J]. IEEE Transactions on Automatic Control, 2005, 56(6):1307-1321.
- [7] Bulychev P, David A, Larsen K G, et al. UPPAAL-SMC: Statistical Model Checking for Priced Timed Automata[J]. Electronic Proceedings in Theoretical Computer Science, 2012, 85.
- [8] Hartonas-Garmhausen V, Cimatti A, Clarke E, et al. Verification of a safety-critical railway interlocking system with real-time constraints[J]. Science of Computer Programming, 2000, 36(1):53-64.
- [9] Zengming Yu, Zhengdong Liu. The change of interlocking function in communication based train control system[J]. Railway Operation Technology,2011, 17(4):13-15.
- [10] Amir Pnueli. Formal Verification: All Qucstiom and Some Answers.CS Leading Teachers Course, WIS, May 9,1999.
- [11] Wing J M, A specifier'S introduction to formal methods [J]. Computer, 1990, 23(23) : 8-22.
- [12] Alur R. Timed Automata[J]. Lecture Notes in Computer Science, 1999, 126(94):183--235.
- [13] Yang yang, Ming Pan, Meifang He. Formal specification process of the interlocking software with Petri nets[J]. China Railway Sciences,2002,23(3):49-54.
- [14] Tiantian Tang. Research on Application of Software Reliability Design Method in Computer Interlocking System[D]. Hefei Polytechnic University,2004.
- [15] Guanning Wang. Modeling and Verification of Interlocking Route Control Process Based on UPPAAL[D]. Beijing Jiaotong University,2009.
- [16] Hei X, Takahashi S, Hideo N. Toward developing a Decentralized Railway Signalling System Using Petri Nets[C]// Robotics, Automation and Mechatronics, 2008 IEEE Conference on. IEEE, 2008:851-855.
- [17] Tiejiang Wang, Meng Li. Z specification for computer interlocking software [J]. ChinaRailway Society, 2003, 25(4):62-66.
- [18] Wu C E, Yang W Y, Ting H C, et al. Traffic pattern modeling, trajectory classification and vehicle tracking within urban intersections[C]// International Smart Cities Conference. 2017:1-6.
- [19] Zhang H, Li W, Ding H, et al. Observer-Pattern Modeling and Nonlinear Modal Analysis of Two-stage Boost Inverter[J]. IEEE Transactions on Power Electronics, 2017, PP(99):1-1.



# ComD2: Family of Techniques for Inspecting Defects in Models that Affect Team Communication

Adriana Lopes, Ursula Campos and Tayana Conte  
USES Research Group  
Instituto de Computação, Universidade Federal do  
Amazonas (UFAM)  
Manaus, AM - Brazil  
{adriana, usc, tayana}@icomp.ufam.edu.br

Clarisse Sieckenius de Souza  
Semiotic Engineering Research Group  
Departamento de Informática, PUC-Rio  
Rio de Janeiro, RJ - Brazil  
clarisse@inf.puc-rio.br

**Abstract** — Communication failures in software development teams can compromise the software quality. Therefore, identifying and mitigating risks for effective team communication are important activities in software development. Software models are one of the means of communication in development teams, because it communicates other members of the development team about the software. Thus, our research focuses on inspection techniques for identifying defects that affect the team communication through the software models. This paper presents a family of techniques for inspecting defects that affect team communication, called ComD2 (Communication between Designers and Developers). The ComD2 family was developed based on theories that investigate different ways of communication. For the time being, the ComD2 family has three specific inspection techniques for UML models, such as class diagrams, activity diagrams, and state machine diagrams. We performed a feasibility study and the results showed that the ComD2 family was considered useful for the identification of defects that affect the team communication through the software models.

**Keywords** - *inspection technique, communication artifact, UML diagrams, human-centered computing; software engineering;*

## I. INTRODUCTION

According to Reed and Knight [1], effective communication is one of the most critical components of working in software teams. In software development, the communication is carried out through face-to-face discussions in co-located or distributed teams [2], besides the support offered by tools [3]. Software models are also used as means of communication in software development teams [4]. In this paper, we explore the communication of software development teams through software models.

Software models that support the communication in different domains can be considered boundary objects. Boundary objects are used for different purposes and in different domains while maintaining their authenticity [5]. The term boundary object comes from the use of objects that facilitate the sharing of information across linguistic, cultural, or knowledge boundaries, such as the communication between a software development team and its client.

Communication failures from software models can come from information that is not clearly expressed by their producers (people who created the models). Thus, other members of the software development team (i.e. consumers, who comprehend the models for the creation of other artifacts) may have different interpretations of the ones intended by the producers. Different interpretations can introduce defects during the production of software; such as the omission of some necessary information or the vague definition of information, thus allowing multiple interpretations [6].

The propagation of defects associated with model representations can be costly, especially in large or complex system development projects. Can defects be detected and remedied by model consumers down the communication line? How do they affect communication in these and other (undetected) cases? Is there a way to prevent or at least minimize such defects? Our research aims to contribute to answer these questions and starts with the following interrogation: *What defects in software models can affect the communication of software development teams?* To answer this question, we have developed a family of techniques called ComD2 (Communication between Designer and Developers). The purpose of the ComD2 family is to support the identification of defects that affect the team communication, i.e. the communication of the designer<sup>1</sup> to the developers at development time. The collaboration between designer and developers is one of the factors for the success of software development [8]. We have initially developed three specific inspection techniques for UML class diagrams, activity diagrams, and state machine diagrams. The techniques were developed for these models because they are among the most frequently used in the industry [9].

The ComD2 family of techniques was developed based on theories related to Human-Centered Computing (HCC), a field of research that integrates theories and methodologies in research on machines, human and application domains [10]. In particular, the Semiotic Engineering, a theory originally proposed for Human-Computer Interaction [11][12], which has

---

<sup>1</sup> We use the term *designer* for the Software Designer, also called Information Architect, i.e., the professionals involved in designing the software solution.

been extended to account for HCC, investigating different forms of communication through and about software, both during development (between producers and consumers of software development artifacts) and at use time (between software producers and end users, through systems interfaces). De Souza et al. [12] propose that Grice’s Cooperative Principle [13] can be used to assess the effectiveness and efficiency of communication achieved through software products (or representations thereof). Thus, we also adopted this Gricean principle as a theoretical basis for the development of the ComD2 family.

The verification items of the ComD2 family help practitioners classify different defects that are found in software models [14] and detect which dimension(s) of model representation associated with the defect can lie at the origin of potential miscommunication. The employed dimensions of representation are Syntactic (relationship between model and the modeling language), Semantic (relationship of the model with the problem domain) and Pragmatic (relationship of the model with the stakeholders) [15].

We conducted a feasibility study with the initial version of the ComD2 family in an academic environment. This study was performed with 30 participants who had knowledge on class diagrams, activity diagrams and state machine diagrams. The results showed that defects in the Semantic and Pragmatic dimensions may affect the effective communication between producers and consumers of the models. With the support of ComD2 family, we have indications about the different defects that impact the team communication. Therefore, Grice’s Cooperative Principle and the different defects associated with the dimensions of representation are important concepts for reducing team communication failures using the models.

The remainder of this paper is organized as follows: Section 2 presents the theoretical background and related work. Section 3 presents the ComD2 family of techniques. Section 4 presents the feasibility study. Section 5 presents the discussion of the obtained results. Finally, Section 6 presents the final considerations and future work perspectives.

## II. BACKGROUND AND RELATED WORK

This section presents the concepts used in the definition of the ComD2 family of techniques. We also present the main works related to the concepts adopted in the techniques.

### A. Grice’s Cooperative Principle

Grice’s Cooperative Principle assists in the expression of essential characteristics of effective and efficient communication [13]. According to Grice, productive conversation (communication) depends on the observation of reciprocal cooperation, which is established by four maxims:

- (i) **Quantity** - Make your contribution as informative as necessary, and no more than necessary;
- (ii) **Quality** - Try to make your contribution true. Do not say what you believe to be false and do not say something that you do not have adequate evidence of;
- (iii) **Relation** - Be relevant, that is, do not introduce issues that do not come to the case under discussion; and

- (iv) **Manner** - Be clear, brief and organized with your input. Avoid obscurity of expression, ambiguity.

Breaking one or more of these maxims may lead to communication failure. However, an adequate use of Grice’s maxims involves the concept of *implicature*, that is, information that can be inferred from statements. Conventional implicatures can be inferred from the conventional meaning of word. There are also conversational implicatures, that is, inferences that can be drawn from participants of a given conversational context in order to fulfill certain gaps and omissions to (re)establish coherence and consistency in communication. Therefore, unlike conventional implicatures, conversational implicatures cannot be resolved by invoking the usual meaning of information represented in communication and require different kinds of inferences.

Grice’s maxims have been previously used by Santana et al. [16] to analyze interaction diagrams modeled with MoLIC (Modeling Language for Interaction as a Conversation). MoLIC diagrams, which are based on Semiotic Engineering, allow us to represent the interaction of the user with the system as a communication process. The results showed that Grice’s maxims can indeed help detect human-computer interaction problems in MoLIC diagrams. We thus extend the object of inspection and use Grice’s maxims to assess effective communication between producers and consumers of other kinds of software models.

### B. Defect Inspection in Software Models

Software defects may be related to an inappropriate comprehension of the information within the models. Granda et al. [14] present a classification for defects that are commonly found in UML models, which are presented in Table 1.

TABLE I. TYPES OF DEFECTS, ADAPTED FROM [14].

Type	Description
Omission	The required information has been omitted.
Incorrect Fact	Some information in the model contradicts the list of requirements or general knowledge of the system domain.
Inconsistency	Information in one part of the model is inconsistent with information in other parts in the model
Ambiguity	The information in the model is ambiguous. This can lead to different interpretations of information.
Extraneous Information	The information that is provided is not required in the model.
Redundant	Information is repeated in the model.

Inspection is a method used to identify defects with lower cost during the development process [17]. According to Qazi et al. [18], the main purpose of an inspection is to identify defects to reduce costs and improve software quality.

Travassos et al. [19] developed a family of seven techniques for inspection, called OORTs (Object Oriented Reading Techniques). The techniques of the OORTS family can be used to inspect object-oriented models with regards to: (i) the different models used, such as use case diagrams and class diagrams, ensuring consistency among such models; and (ii) the requirements and models, ensuring traceability within a domain in order to find defects between them. However, we

did not find inspection techniques with the purpose of supporting the defects identification that affects the communication of the team through the use of models. For this reason, we proposed inspection techniques to identify defects that impact the communication of the team through the use of models. We have developed our techniques in the context of HCC and related theories that study different ways of communication.

### III. THE COMD2 FAMILY

ComD2 offers specific techniques for inspecting defects that affect the team communication through the software models. The purpose of the ComD2 family is to assist experienced and novice practitioners by checking for defects in models that impair the communication of the team. The theoretical basis of ComD2 family is presented in the following paragraphs. From this theoretical basis, it is possible to develop specific techniques for other software models.

Using Grice’s Cooperative Principle [13], ComD2 family uses the four maxims. We created verification items to support the identification of discrepancies (these discrepancies may be defects or not) based on these four maxims. Based on the maxim of Quantity, we developed verification items for the necessary content (and no more than necessary), in the models e.g., for the class diagram: “*Are all necessary classes of the problem domain in the diagram?*”. Based on the maxim of Quality, we developed verification items for the identification of false information in the models, e.g. for the class diagram, we have: “*Do the classes have content that affects the quality of the model?*” Based on the maxim of Relation, we developed verification items for the identification of information that is not relevant to the models, e.g.: “*Are classes relevant to system modeling?*”. Based on the maxim of Manner, we developed verification items for the identification of information that is not clear in the model, e.g.: “*Are there classes and relationships with descriptions that are not clear?*”.

We observed that when the maxims are not respected in the models, they could cause defects in software. This occurs due to the lack of understanding of the consumers on the intention of the producers regarding the model. Thus, for each verification item, we used the defect classification that is presented by Granda et al. [14] (see Table 1). From the verification items, it is possible to classify the defects. For instance, for the verification item that is based on the maxim of Quantity, we relate this item as follows: *Are all necessary classes of the problem domain are in the diagram? If not, this may be an Omission discrepancy.*

Each verification item presents the dimensions of representation that was affected by the defects in the models, being these Syntactic, Semantic and Pragmatic [15]. These dimensions can help in comprehending the defects that cause the communication failures from software models. Defects related to the *form* of representation are associated with the syntactic dimension, whereas the defects related to the *content* of information are associated with the semantic and pragmatic dimensions. We highlight in the verification items the different dimensions of representation related to the defects. The following verification item shows the dimension of representation affected by the defect, which refers to the

*content* of system information in the modeling: *Are all necessary classes of the problem domain are in the diagram? If not, this may be an Omission discrepancy (Semantic).*

Verification items can be proposed for each one of the models. At this initial state, we have developed specific techniques for inspecting three UML models: class diagrams, activities diagrams, and state machine diagrams. These three models are commonly used in software development [9]. There are verification items for unique elements in the models, such as the following item for the Association element in the class diagram: *According to the problem domain, are all Association relationships established among classes? If not, this may be an Omission discrepancy (Semantic and Pragmatic).* In some cases, there are verification items for all elements of a model, such as the following item for the class diagram: *Are there elements with descriptions that are not clear? If so, there are probably Ambiguity discrepancy (Pragmatic).*

The Fig.1 presents some verification items for class diagrams. We use the following structure in the ComD2 family techniques: verification items for the elements of the respective models, which are related to the questions based on Grice’s maxims. Each verification item suggests the identification of one or more defects and support the classification of the information representation dimension that such defects affect in the models.

Class Diagrams Technique	
<b>Is the necessary information, and no more than necessary, present in the model?</b>	
Class	Are all necessary classes of the problem domain are in the diagram? If not, this may be an Omission discrepancy (Semantic).
...	...
<b>Does the information in the model contain statements that are not true?</b>	
Class	Do the classes have content that affects the quality of the model? If so, there are probably Inconsistency and/or Extraneous Information discrepancies (Semantic).
...	...
<b>Is the information relevant to system modeling?</b>	
Class	Are classes relevant to system modeling? If not, there are probably Extraneous Information and/or Redundant discrepancies (Semantic).
...	...
<b>Is the information difficult to understand in the model?</b>	
Class	Are there classes and relationships with descriptions that are not clear? If so, there are probably Ambiguity discrepancy (Pragmatic).
...	...

Figure 1. Extracts of the ComD2 techniques for class diagrams.

The verification items are divided into categories corresponding to the Grice’s maxims (highlighted in gray in Fig. 1.), such as: **Is the necessary information, and no more than necessary, present in the model?** (verification items related to the maxim of Quantity); **Does the information in the model contain statements that are not true?** (verification items related to the maxim of Quality); **Is the information relevant to system modeling?** (verification items related to the maxim of Relation); **Is the information difficult to understand in the model?** (verification items related to the maxim of Manner). Despite the structure used, we do not define the order for the use of each category. The inspection techniques for the activity and state machine diagrams use the same structure presented in Fig.1. These techniques are available in [20].

#### IV. FEASIBILITY STUDY WITH THE COMD2 FAMILY

In order to evaluate the initial techniques of the ComD2 family, we conducted a feasibility study in an academic environment. In this study, we analyzed the effectiveness (ratio between the number of detected defects and the total number of defects) and efficiency (ratio between the number of defects per inspection time) of each participant for the different techniques. The adopted measures of efficiency and effectiveness are often used in studies investigating inspection techniques [21] [22]. We also evaluated the participants' perceptions on the techniques.

##### A. Planning and Execution of the Feasibility Study

In the planning stage, we selected 30 participants for the study. The participants are undergraduate students and have a basic dimension of knowledge about software modeling with class diagrams, activity diagrams and state machine diagrams. We selected the UML models of a real web and mobile development project. In addition, we prepared all necessary artifacts, such as forms for the participants to report the identified discrepancies and post-study questionnaires. The package with the artifacts used also available in [20].

In the execution stage, we first gave lectures on the techniques of the ComD2 family. Then, the participants performed the inspection of the UML models individually. After the inspection, we applied the post-study questionnaires. During the study, two of the researchers took notes for later analysis.

##### B. Results of the Feasibility Study

After the execution of the study, we verified whether the technique achieved the goal of detecting defects. The oracle of defects contained a total of 25 defects in the three diagrams (10 defects in the class diagrams, 8 defects in the activity diagrams and 7 defects in the state machine diagrams).

Table 2 presents the participants (column P#), number of defects found by each participant (CD column for the class diagram, AD column for the activity diagrams and SD column for state machine diagrams), inspection time (in hours) and the effectiveness of the participants (EfC column for the class diagrams technique, EfA column for the activity diagrams technique, EfS column for state machine diagrams technique) and the average effectiveness of each technique (in the last line in the Table 2). As the participants performed the inspection of the created models at the same time, we did not evaluate the individual efficiency of each participant with the techniques.

Analyzing the effectiveness indicator, we noticed that the inspectors were able to identify an average of 53% of the defects with the class diagrams technique, 35.8% with the activity diagrams technique and 25.8% with the state machine. This is a positive result in terms of effectiveness when compared to the indicators achieved by other inspection techniques for models [22]. The results showed that ComD2 can support the detection of defects. Regarding efficiency, as the participants used three techniques in the inspection of the models, we analyzed the efficiency of the entire ComD2 family. The participants found an average of 10.94 defects per hour with the techniques. However, as the number of defects is directly dependent on the inspected models, is not suitable to

compare the results of efficiency from this study with the results of other techniques.

TABLE II. RESULTS PER PARTICIPANTS WITH THE COMD2 FAMILY.

P#	CD	AD	SD	time (hours)	EfC (%)	EfA (%)	EfS (%)	EfT (%)
P1	1	3	2	1,33	10	37,5	25	24,1
P2	6	4	3	1,21	60	50	37,5	49,1
P3	4	2	1	0,95	40	25	12,5	25,8
P4	5	3	1	0,81	50	37,5	12,5	33,3
P5	3	2	1	1,26	30	25	12,5	22,5
P6	4	3	4	1	40	37,5	50	42,5
P7	4	4	3	1,16	40	50	37,5	42,5
P8	8	2	3	0,63	80	25	37,5	47,5
P9	7	3	3	1,48	70	37,5	37,5	48,3
P10	7	1	1	0,83	70	12,5	12,5	31,6
P11	5	2	1	0,81	50	25	12,5	29,1
P12	9	2	1	0,81	90	25	12,5	42,5
P13	6	4	1	1,3	60	50	12,5	40,8
P14	9	3	3	0,95	90	37,5	37,5	55
P15	8	4	1	1	80	50	12,5	47,5
P16	4	4	2	1,01	40	50	25	38,3
P17	3	4	3	1,13	30	50	37,5	39,1
P18	4	4	4	0,96	40	50	50	46,6
P19	8	0	3	0,26	80	0	37,5	39,1
P20	5	2	1	1,16	50	25	12,5	29,1
P21	6	2	3	1,33	60	25	37,5	40,8
P22	2	3	1	1,06	20	37,5	12,5	23,3
P23	8	2	3	1,05	80	25	37,5	47,5
P24	4	1	2	1,06	40	12,5	25	25,8
P25	5	7	3	1,01	50	87,5	37,5	58,3
P26	4	3	2	1,05	40	37,5	25	34,1
P27	3	4	1	1,06	30	50	12,5	30,8
P28	7	2	2	1,21	70	25	25	40
P29	5	3	1	1,21	50	37,5	12,5	33,3
P30	5	3	2	1,21	50	37,5	25	37,5
<b>Average Effectiveness</b>					53	35,8	25,8	-

We analyzed the post-study questionnaires to understand participants' perceptions. The questionnaire had three open questions, the first question being: *What is your perception with the use of the techniques?* We analyzed the responses of participants P2, P12 and P13, who had more than 40% effectiveness in detecting defects with the ComD2 family (considering the effectiveness of the three techniques):

*"The techniques are very practical. The dimensions of representation facilitate the review and help understand the intent of the artifact's author. With this classification, it is also possible to correct defects more easily"* (P12)

*"The techniques show an intermediate dimension of representation between the requirements and implementation. Therefore, it is a great way to analyze the team's understanding of the requirements"* (P21)

Other participants reported perceptions that could be used to improve the techniques, such as joining some verification items that were considered repetitive. Some quotations from the participants were:

*“The techniques help to ensure the reliability of the model, but some points are repetitive”* (P12)

*“The techniques can be more unified in the description of the problems, since some errors are the same in different models”* (P22)

In spite of perceived issues, while answering the first question our participants considered the information representation dimensions useful, because they help gain better understanding of information that the model communicates. We believe this may improve the identification of defects that undermine the consumers’ understanding of the intention of the producers of the models.

The second question posed to participants was: *Do the information representation dimensions help to understand the defects that could undermine the understanding of the model?* Some participants reported the following:

*“Yes, especially the defects in the semantic and pragmatic dimensions that can compromise software development”* (P3)

*“Through the dimensions of representation, it is possible to see if we should change only something in the syntax or redesign parts of the system”* (P13)

The participant’s utterances showed that the defects in the semantic and pragmatic dimensions are the types of defects that may most affect the understanding of a model; since the syntactic dimension defects may not compromise both the understanding of the model language.

To understand if participants had difficulties with the ComD2 family, the post-study questionnaire included a third question: *What are the difficulties with using the techniques?* The following are excerpts from some of the answers.

*“Although the techniques help in the classification of the representation dimensions, I had doubts with the classification of defects in Semantic and Pragmatic dimensions”* (P6)

*“Certain defects fall into more than one dimension, so it is necessary to evaluate and interpret each case in order to avoid misunderstanding”* (P29)

Regarding the citations of participants P6 and P29, we noticed that there are difficulties with understanding the dimensions of representation and related defects. Although the techniques help in the classification of the dimension of representation associated to the defects, we can make improvements in the techniques with regards to the examples of the different dimensions of representation.

After the participants ended the study activities, we asked them which types of information in models could affect team communication. We noticed that some participants considered that unnecessary, irrelevant, ambiguous and false information affect the communication of the team when using these models. This type of information violates Grice’s four maxims and

indicates that this theory is adequate for analyzing communication between producers and consumers. Moreover, based on the results of this feasibility study, we define a prioritization of the categories in ComD2 family. The proposed prioritization follows this order: (1st) Quality, (2nd) Quantity, (3rd) Relation and (4th) Manner. This prioritization order should be followed in the next applications of the ComD2 family.

## V. DISCUSSION

The results of the study provided initial evidence to the feasibility of ComD2 family to inspect defects that may impact the communication between producers and consumers. Regarding our research question - *What defects in software models can affect the communication of development teams?* - the results obtained with the techniques showed that the defects in the Syntactic dimension do not always affect the consumers’ understanding, since they are related to the syntax of the language used for modeling. Defects at the Semantic (relationship of the model with the problem domain) and Pragmatic (relationship of the model with the stakeholders) dimensions can affect the communication of the team. Regarding defects in the Semantic dimension, communication failures occur because the consumers infer explicit content inconsistent with the problem domain (conventional implicature of the explicit content in the model). However, if the consumers have knowledge about the problem domain, these defects can be perceived and not propagated to other artifacts (e.g. when the consumer uses the class diagram for system coding and it perceives the lack of a domain class, then this class could be added). Defects in the Pragmatic dimension may not be perceived by the consumers due to lack of information context (conversational implicature of the implied content in the model). In this case, communication failures occur because consumers may not understand the intention of producers. However, the ComD2 family can help reduce risks for effective team communication through models.

In the feasibility study of the ComD2 family, there are limitations, such as the fact that the participants are undergraduate students and that the study is conducted in an academic environment. Regarding this limitation, Fernandez et al. [8] state that undergraduate students who do not have experience in the industry may have similar skills to less experienced practitioners; and one of the goals of the techniques is to assist practitioners with no experience in the inspection process of models. Another limitation is that the inspected models were from a development project, since it is not possible to state that these models represent all types of class diagrams, activity diagrams and state machine diagrams. Therefore, we intend to carry out new studies with the set of techniques for different models. Regarding the indicators of effectiveness and efficiency that were adopted, they are often used in studies investigating inspection techniques [22].

## VI. CONCLUDING REMARKS AND FUTURE WORK

The purpose of this paper was to answer the following research question: *“What defects in software models can impact the communication of development teams?”*. To do so, we developed a family of techniques called ComD2 that helps

practitioners identify defects that affect software team communication. We initially proposed and evaluated specific techniques of the ComD2 family for inspecting class diagrams, activity diagrams, and state machine diagrams. The results of the evaluation provided initial evidence to the feasibility of ComD2 family.

As future work, we intend to improve the ComD2 techniques and perform an empirical study in comparison with other specific techniques for inspecting class diagrams, activity diagrams and state machine diagrams. Furthermore, we intend to carry out a longitudinal study with the ComD2 family to evaluate the identification of defects that affect the team communication through the models employed during the software development.

#### ACKNOWLEDGMENT

We thank the undergraduate students for their participation in the feasibility study. We would like to thank the financial support granted by UFAM, CNPq through processes numbers 423149/2016-4, 311494/2017-0 and 304224/2017-0, and CAPES through process number 175956/2013.

#### REFERENCES

- [1] A. H. Reed and L.V. Knight, "Effect of a virtual project team environment on communication-related project risk", *International Journal of Project Management*, vol. 28 (5), 2010, pp. 422–427.
- [2] E. Diel, S. Marczak, D. S. Cruzes, "Communication Challenges and Strategies in Distributed DevOps", *Proceedings of the 11th International Conference on Global Software Engineering (ICGSE 2016)*, 2016, pp. 24-28.
- [3] V. Käfer, "Summarizing software engineering communication artifacts from different sources", *Proceedings of the 11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2017)*, 2017, pp. 1038-1041.
- [4] M. Pikkarainen, J. Haikara, O. Salo, P. Abrahamsson, J. Still, "The impact of agile practices on communication in software development", *Empirical Software Engineering*, vol. 13 (3), 2008, pp. 303-337.
- [5] P. Ralph, M. Chiasson and H. Kelley, "Social theory for software engineering research", *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering (EASE '16)*, 2016, pp. 44-55.
- [6] R. M. de Mello, E. N. Teixeira, M. Schots, C. M. L. Werner and G. H. Travassos, "Verification of software product line artefacts: a checklist to support feature model inspections", *Journal of Universal Computer Science*, vol. 20(5), 2014, pp. 720-745.
- [7] A. M. Qazi, S. Shahzadi and M. Humayun, "A comparative study of software inspection techniques for quality perspective", *International Journal of Modern Education and Computer Science*, vol. 8 (10), 2016, pp. 9-16.
- [8] J. M. Brown, G. Lindgaard, and R. Biddle, "Collaborative events and shared artefacts: Agile interaction designers and developers working toward common aims," *Proceedings - 2011 Agile Conference, Agile 2011*, pp. 87–96.
- [9] G. Reggio, M. Leotta, F. Ricca, and D. Clerissi, "What are the used activity diagram constructs? A survey", *Proceedings of the 2nd International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2014)*, 2014, pp. 87–98.
- [10] Sebe, N. Human-centered computing. In Nakashima, H., Aghajan, H., & Augusto, J (Eds.), *Handbook of ambient intelligence and smart environments*, pp. 349–370, 2010. DOI: 10.1007/978-0-387-93808-0\_13.
- [11] C. S. De Souza, *The Semiotic Engineering of Human-Computer Interaction (Acting with Technology)*. The MIT Press, 2005.
- [12] Clarisse Sieckenius de Souza, Renato Fontoura de Gusmão Cerqueira, Luiz Marques Afonso, Rafael Rossi de Mello Brandão and Juliana Soares Jansen Ferreira. 2016. *Software Developers as Users: Semiotic Investigations in Human-Centered Software Development*. In Springer International Publishing Switzerland. DOI 10.1007/978-3-319-42831-4.
- [13] H. P. Grice, "Logic and conversation". *Syntax and Semantics 3: Speech arts*, ed. Peter Cole and Jerry Morgan, 1975, pp. 41–58.
- [14] M. F Granda, N. Condori-fernández, T. E. J. Vos, O. Pastor, "What do we know about the defect types detected in conceptual models?", *Proceedings of the IEEE 9th Int. Conference on Research Challenges in Information Science (RCIS 2015)*, 2015, pp. 96–107.
- [15] M. Priyanka and R. Phalnikar, "Generating UML diagrams from natural language specifications", *International Journal of Applied Information Systems*, vol. 1(8), 2012, pp. 19-23.
- [16] B. S. Silva, V. C. O. Aureliano, S. D. J. Barbosa, "Extreme designing: binding sketching to an interaction model in a streamlined HCI design approach", *Proceedings of the VII Brazilian Symposium on Human Factors in Computer Systems*, 2006, pp. 101 – 109.
- [17] P. C. Rigby and C. Bird., "Convergent contemporary software peer review practices", *Proceedings of the 9th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2013)*, 2013, pp. 202–212.
- [18] A. M. Qazi, S. Shahzadi and M. A. Humayun, "Comparative study of software inspection techniques for quality perspective", *International Journal of Modern Education and Computer Science*, vol. 10 (8), 2016, pp. 9-16. DOI: 10.5815/ijmecs.2016.10.02.
- [19] G. Travassos, F. Shull, M. Fredericks, V. Basili, "Detecting defects in object-oriented designs: using reading techniques to increase software quality", *Proceedings of XIV ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, And Applications*, 1999, pp. 47-56.
- [20] A. Lopes, T. Conte, C. S. de Souza. 2018. *ComD2 (Communication between Designer and Developers): A Family of Techniques for Inspecting Defects that Affect Communication from Models*. USES Research Group Technical Report. TR-USES-2018-0003. Available: <http://uses.icomp.ufam.edu.br/wp-content/uploads/2018/03/TR-USES-2018-0003.pdf>
- [21] A. Fernandez, S. Abrahão, E. Insfran, and M. Matera, "Further analysis on the validation of a usability inspection method for model-driven web development", *International Symposium on Empirical Software Engineering and Measurement (ESEM 2012)*, 2012, pp. 153-156.
- [22] N. M. C. Valentim, J. Rabelo, A. C. Oran, S. Marczak, T. Conte, "A Controlled Experiment with Usability Inspection Techniques Applied to Use Case Specifications: Comparing the MIT 1 and the UCE Techniques", *Proceedings of the 18th International Conference on Model Driven Engineering Languages and Systems (MODELS 2015)*, 2015, pp. 206-215.

# Effects of Model Composition Techniques on Effort and Affective States: A Controlled Experiment

Mateus Manica, Kleinner Farias, Lucian J. Gonçalves, Vinícius Bischoff  
PPGCA, Unisinos, RS, Brazil  
mateusman@edu.unisinos.br  
kleinnerfarias@unisinos.br,  
{lucianj,viniciusbischof}@edu.unisinos.br

Bruno C. da Silva  
Department of Computer Science,  
Software Engineering, Cal Poly.  
San Luis Obispo, USA  
cbcda Silv@calpoly.edu

Everton Guimarães  
Department of Computer Science  
Drexel University, Philadelphia, USA  
etg38@drexel.edu

**Abstract**—Even though existing heuristics and specification-based techniques support composing design models, it is still considered a time-consuming and highly intensive task. In addition, there is a lack of studies exploring the effects of composition techniques on software developers’ affective state and development effort. This study reports a pilot study to investigate these effects while developers apply composition techniques to detect and resolve inconsistencies in output-composed models. In this sense, a widely known wearable EEG headset, namely Emotiv EPOC, with 14 channels was used, while developers made use of heuristic-based and specification-based composition techniques to evolve design models. Our results suggest that using heuristic-based techniques produced a higher effect on the developers’ affectivity, compared to specification-based techniques. Moreover, the higher the effects on the developers’ affectivity, the higher the odds to invest less effort and produce correctly composed design models.

**Keywords**- UML; Cognitive Effort; Empirical Evaluation; EEG.

## I. INTRODUCTION

Model composition has been used for supporting many software-development activities, e.g., adding new features to evolve design models. The term *model composition* may be briefly seen as a set of tasks that should be realized over two (or more) input models,  $M_A$  and  $M_B$ , so that an output-intended model,  $M_{AB}$ , can be produced [1]. Given the input model elements of  $M_A$  and  $M_B$  may conflict with each other, developers must often spend some effort to solve such conflicts before producing,  $M_{AB}$ . In recent years, researchers and practitioners have focused on producing composition techniques [2][3] as means to obtain  $M_{AB}$  from  $M_A$  and  $M_B$ , by investing as little effort as possible.

These techniques could be classified into two categories: (1) *specification-based techniques* (e.g., Epsilon [2]), where developers explicitly specify both correspondence and composition relations between the elements of  $M_A$  and  $M_B$  to produce  $M_{AB}$ ; and (2) *heuristic-based techniques* (e.g., override, merge and union algorithms [4]), in which developers use a set of predefined heuristics for “guessing” the correspondence and composition relations between the elements of  $M_A$  and  $M_B$ . However, in fact, both categories may lead *composition conflict problems*. Composition conflicts consist of contradictions between the values assigned to the properties of design models [7][8]. For example, a UML class *Researcher* assigned as a concrete class (i.e., *Researcher.isAbstract* = false), whereas in

the  $M_B$  delta model the class *Researcher* is set as an abstract one (i.e., *Researcher.isAbstract* = true). These contradicting values assigned to *isAbstract* represent a conflict that must be solved by developers. However, if this issue is not properly addressed, inconsistencies are inserted into the output-composed model  $M_{CM}$ . For example, *Researcher.isAbstract* = false represents an inconsistency as the expected value would be true. To sum up, composition techniques cannot guarantee an optimal solution, and usually developers produce an output-composed model,  $M_{CM}$ , with inconsistencies—typically after solving conflicts between  $M_A$  and  $M_B$  improperly. That is, usually  $M_{CM}$  and  $M_{AB}$  do not match ( $M_{CM} \neq M_{AB}$ ) [1][5]. In this sense, developers often need to invest some extra effort to detect and resolve such inconsistencies.

For this reason, composing design models is still considered as being an error-prone and time-consuming task [5]. Recent studies have shown the relationship between development practices and their effects on the developers’ cognitive activities [13]. However, nothing has been done to reveal to what extent the use of composition techniques might influence the developers’ affective states, including frustration, excitement, and meditation. Therefore, this paper seeks to apply neuroscience methods to analyse and understand affective states, while software developers perform compositions to support the evolution of design models. This study is important to identify cognitive factors that may affect the practice of software modelling. Using too much effort to manipulate models may explain the selective use of UML models [12]. For this, a pilot controlled experiment was performed for grasping the effects of composition techniques on the developers’ affective states by monitoring their cognitive processes. We have used Emotiv EPOC wireless EEG headset [6] with 14 channels to collect and process affective states in 288 integration scenarios of model elements of UML class diagrams [13] in the context of 18 evolution scenarios.

## II. EXPERIMENTAL DESIGN

### A. Objective and Research Questions

The goal of this study is to **analyze** compositional techniques **for the purpose of** investigating their effects **with respect to** effort, correctness and affectivity **from the perspective of** software developers **in the context of** evolution of software

design models. The goal of this work was formalized using the GQM template [9]. Therefore, three Research Questions (RQ) emerged from this objective:

- **RQ1:** What is the relative effort of composing two input models using specification-based composition techniques with respect to heuristic-based composition techniques?
- **RQ2:** Is the number of correctly composed models higher using specification-based techniques with respect to heuristic-based ones?
- **RQ3:** Does the use of heuristic-based technique cause a higher effect on the developer's affectivity than technical based Specification?

### B. Hypotheses

These research questions seek to explore how cognitive processes triggered to perform composition techniques may influence (1) the effort invested by developers to integrate design models, (2) the correctness of the compositions, and (3) the affective states generated during composition tasks respectively. For this, to answer these questions the following null-hypothesis were formulated.

**H1: Null Hypothesis 1,  $H_{1-0}$ :** the specification-based composition technique requires less effort (or equal to) than heuristic-based technique to produce  $M_{AB}$  model from  $M_A$  and  $M_B$  by the developer.

**H2: Null Hypothesis 2,  $H_{2-0}$ :** The specification-based composition technique produces a smaller or equal number of models properly composed than the heuristic-based composition technique.

**H3: Null Hypothesis 3,  $H_{3-0}$ :** The use of the specification-based composition technique causes a lower (or equal) impact on the affectivity than the heuristic-based technique.

### C. Study Variables

The dependent variables of H1 are associated with each part of the model composition effort equation:  $effort(M_A, M_B)$ , representing the general effort for composing two models;  $f(M_A, M_B)$ , the effort to apply composition techniques;  $diff(M_{CM}, M_{AB})$ , the effort required to detect inconsistencies; and  $g(M_{CM})$ , the effort required to resolve the inconsistencies. All these variables are measured in minutes.

The dependent variable of H2 is the correctness (Cor) of the output-composed models. If  $M_{CM} = M_{AB}$ , then the composition was correct (Cor = 1); otherwise  $M_{CM} \neq M_{AB}$ , then the composition was incorrect (Cor = 0). The hypothesis regarding the correctness (H2<sub>1</sub>) evaluates which technique produced more correctly composed models.

The dependent variables of H3 are affective indicators: (i) *engagement* (Engaj( $M_{CM}$ )) is the result of experience in the alert, attention to the task status consciously. When its levels are negative mean a high surveillance or even boredom; (ii) *meditation* (Medit( $M_{CM}$ )) represents how much calm or relaxed the user has been during the experimental task; (iii) *excitement* (Excit( $M_{CM}$ )) determines a condition of sensorial alert and response readiness; and (iv) *frustration* (Frustr( $M_{CM}$ )) reflects the reaction from the feedback obtained in practice, or can even express stress. These variables are quantified using the Emotiv EPOC. Finally, the independent variables of H1, H2, and H3 are

the heuristic-based and specification-based composition techniques. We observed the use of these techniques in the proposed scenarios to evaluate the impact on dependent variables.

### D. Context and Subject Selection

Our pilot study had three subjects, which used two techniques (i.e., Epsilon and traditional algorithms) to perform six evolution scenarios of design models. These models were used because they were already validated in a previous empirical study [1]. In this way, we can also reduce threats to validity of the study results. All the subjects of this study have experience on software modeling and programming. Each participant was exposed to the same level of training about the modeling and composition techniques so we can make sure they will share the same knowledge related to model composition.

### E. Experimental Process and Study Setup

The controlled experiment had been organized into 4 different activities as depicted in Figure 1. We assigned the participants to treatments randomly and equally distributed following the within-subjects design, in which all participants in the study will run all activities related to the treatments [10]. In each treatment, participants used a modeling composition technique to perform the 6 experimental tasks.

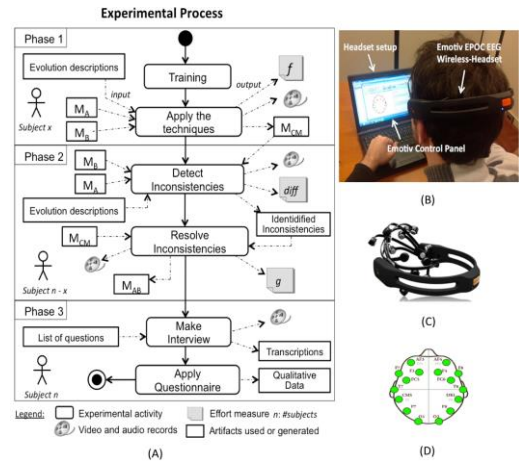


Figure 1. Experimental process (A), Experimental setup (B), Emotiv EPOC (C) and its electrode positions (D).

The activities performed by the participants are described as follows: (1) *training*, each subject received training to assure that all participants had familiarity with the composition techniques; (2) *apply the techniques*, the subjects used the techniques Epsilon and traditional algorithms to integrate two input models,  $M_A$  and  $M_B$ ; (3) *detect inconsistencies*, the next step was to read the composed model produced to detect inconsistencies; (4) *resolve inconsistencies*: Having identified inconsistencies, the participants were encouraged to solve them, seeking to produce the  $M_{AB}$ ; and (5) *make interview and Answer questionnaire*, the authors inquired the participants to think aloud about their experience throughout the experiment.

## III. STUDY RESULTS

### A. RQ1: Effort and Composition Techniques

Figure 5 highlights the effort invested in each required



activity to combine two input design models. We observed developers tend to invest less effort to produce the output-intended model ( $M_{AB}$ ) using heuristic-based techniques rather than the specification-based technique. That is, when participants used Epsilon language they ended up investing more than twice effort to produce the output model. On average, developers invested by about 28 min to run the experimental tasks using a heuristic-based technique. Thus, they spent about 63 min to perform similar tasks using the specification-based techniques. For example, the first participant (a) invested twice more effort using the specification-based technique than using the heuristic-based one.

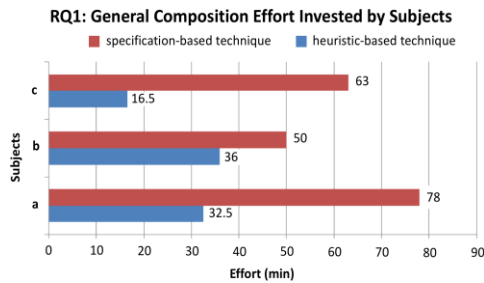


Figure 2. The invested effort to compose design models.

This upward trend was also observed with the other participants. In the second participant (b), the effort increased from 36 min using the heuristic-based technique to 50 min, representing a rise of 38%. Similarly, the third participant (c) also spent a superior effort to compose design models using the specification-based technique. The participant invested 16.5 min using the heuristic-based technique, and 63 min using the specification-based technique. When using heuristic-based technique the participant invested 41% of the effort to compose the design models, compared to the specification-based technique. The results confirmed the findings reported in [6].

Finally, the general result of the H1 is that for all participants applied more effort using the specification-based technique, which suggests the rejection of the null hypothesis ( $H1_{1-0}$ ), and the possible confirmation of the alternative hypothesis ( $H1_{1-1}$ ). We also observed the specification-based technique required more effort to detect and resolve inconsistencies (Figure 5), also confirming the alternative hypothesis H1.

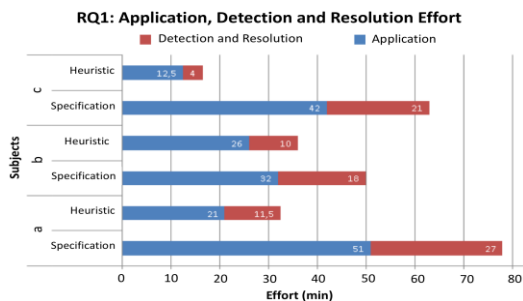


Figure 3. Effort invested in each step of the composition process.

**Conclusion of RQ1:** Developers tend to invest more effort to combine two input models, detect and resolve inconsistencies using a specification-based technique, compared to a heuristic-based technique.

### B. RQ2: Correctness and Composition Techniques

Figure 6 shows the obtained results. The general correctness, i.e., the number of correctly composed model, was 5 using the heuristic-based technique, while 2 using the specification-based technique.

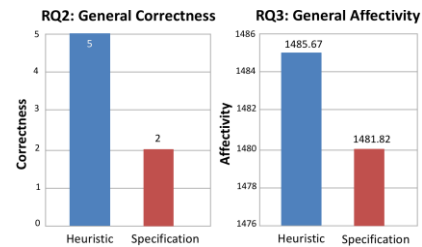


Figure 4. The correctness of the output-composed model (left), and the general affectivity of subjects using heuristic- and specification-based technique (right).

We observed the heuristic-based technique produced a higher number of correctly composed model compared to the specification-based one. As previously mentioned, when using a specification-based technique, the expectation is that the number of models produced correctly,  $M_{CM} = M_{AB}$ , is enhanced, given its flexibility to elaborate the composition rules. However, the results showed the opposite. The specification-based technique has not produced a higher number of correctly composed models, neither generated models with a lower inconsistency rate. Thus, the result suggests the rejection of the null hypothesis ( $H2_{1-0}$ ) and confirmation of the alternative hypothesis ( $H2_{1-1}$ ).

**Conclusion of RQ2:** Developers tend to produce a higher number of correctly composed models using heuristic-based technique, compared to specification-based technique.

### C. RQ3: Affectivity and Composition Techniques

Figure 7 presents the results about the impact of the composition techniques on the developers' general affectivity. The main finding is that the heuristic-based technique produced a higher effect on the developers' affectivity than the specification-based technique. Thus, the result suggests the rejection of the null hypothesis ( $H3_{1-0}$ ) and confirmation of the alternative hypothesis ( $H3_{1-1}$ ).

To better explore this issue, we have examined five facets of developers' affective state, including frustration, engagement, mediation, long-term excitement, and instantaneous excitement. Figure 7 shows the measures the affectivity using heuristic-based technique, while Figure 8 presents the results regarding the specification-based technique. The results show developers' affectivity using the heuristic-based technique outnumbers the ones produced using the specification-based technique.

The results revealed a pattern regarding the measures of affectivity, since the two participants (a and c) obtained very similar indicators, and the participant (b) had different measures. On the other hand, Figure 7 does not present the same pattern regarding the use of the specification-based technique. However, participant (b) had still the higher engagement and frustration indicators. By examining the average engagement of the participants, we observed the value is 5% higher using the heuristic than specification-based technique.

Due to the results presented by the participant (b), we questioned the influence of the frustration indicator on the

quality of the compositions. If the relationship between high levels of frustration and engagement and low excitement would not reflect the participant (b) was under pressure. We point out the pressure can have many different causes, such as personal competitiveness, time available for executing activities, social pressure, among others. Based on the initial results, it is still necessary to carry out further studies to improve knowledge, understanding, as well as to translate it into more realistic results about the influence of affectivity of the participants in the quality of the compositions.

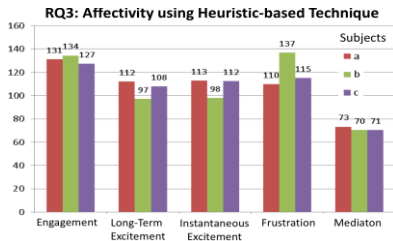


Figure 5. The impact on affectivity using heuristic-based technique.

The specification-based technique (Figure 8) had a minor impact on developers’ affectivity compared heuristic-based approach. Therefore, the data suggest that null hypothesis H3 (H3<sub>1-0</sub>) is confirmed as the specification-based technique showed a lower impact on affectivity. While the specification-based technique requires a greater effort and composition produces a lower amount of correctly composed model, it had a minor impact on the variable affectivity of developers. We also observed the participants had a higher engagement and less frustration using the heuristic techniques.

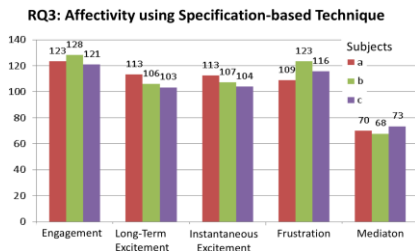


Figure 6. The impact on affectivity using specification-based technique.

**Conclusion of RQ3:** Specification-based technique tends to cause a lower impact on the affectivity of the developers, compared to heuristic-based technique.

#### IV. RELATED WORK

A controlled experiment for exploring the benefits of UML models on the comprehensibility of Java source-code deprived of comments is presented in [14]. The authors concluded that an additional effort was observed to read UML models, but this was paid back in the form of an improved comprehension of source code. In [11], the author mentions program comprehension as the main activity of the software developers. They emphasize that even though a huge amount of research to support the programmer has been done, the high amount of time developers has to grasp source code remained constant over thirty years. The author mentions that the use of EEG could be a reliable way

to measure cognitive load of programmers. However, nothing is presented in this sense. To the best of our knowledge, this work is the first to (1) explore the influence of composition techniques on the developers’ effort, correctness, affective states in the current literature, and (2) provide an initial discussion on the interplay between composition techniques and affective states.

#### V. CONCLUSIONS AND FUTURE WORK

The results suggest the specification-based technique required a higher composition effort, produce a lower amount of output-intended model, as well as caused a lower impact on the developers’ affectivity, compared to its counterpart. As future work, we seek to replicate the study so that a larger sample of data can be produced, allowing hypotheses testing using statistical methods. Finally, this study showed that it is possible to explore affective states to mitigate their impacts on the correctness of composed models. Our expectation is that the issues outlined throughout the paper can encourage other researchers to replicate our study in the future under different circumstances. Finally, we see this paper as a first step in a more ambitious agenda to support empirical assessment of model composition techniques, as well as understanding cognitive and emotional aspects of software developers.

#### REFERENCES

- [1] K. Farias, A. Garcia, J. Whittle, J., C. Chavez, C. Lucena. “Evaluating the effort of composing design models: a controlled experiment”, *Journal on Software & Systems Modeling*, vol. 14, n. 4, pp. 1349-1365, 2015.
- [2] The Epsilon Book, <http://www.eclipse.org/epsilon/doc/book/>, Accessed in 10 March 2018.
- [3] S. Clarke. “Composition of Object-Oriented Software Design Models”, PhD Thesis. Dublin City University, 2001.
- [4] IBM Rational Software Architect, <https://www.ibm.com/developerworks/downloads/r/architect/>, accessed 10 March 2018.
- [5] M. La Rosa, M. Dumas, R. Uba, R. Dijkman. “Business Process Model Merging: An Approach to Business Process Consolidation”, *ACM Transactions on Software Engineering Methodology*, vol. 22, num. 2, pp. 1-42, 2013.
- [6] EMOTIV user manual, <http://emotiv.com/developer/SDK/UserManual.pdf>, accessed 16 March 2018.
- [7] K. Farias. “Empirical evaluation of effort on composing design models”. Ph.D. thesis, Department of Informatics, PUC-Rio, 2012.
- [8] T. Mens “A State-of-the-Art Survey on Software Merging”, *IEEE Transactions on Software Engineering*, vol. 28, num. 5, pp 449-562, 2002.
- [9] V. Basili, G. Caldiera, H. Rombach. “The Goal Question Metric Paradigm”, *Encyclopedia of Software Engineering*, vol. 2, (John Wiley and Sons, 1994), pp. 528–532.
- [10] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, A. Wesslén. “Experimentation in software engineering”, Kluwer Academic Publishers, 2012.
- [11] J. Siegmund. “Program Comprehension: Past, Present, and Future”, *IEEE 23rd Int. Conf. Software Analysis, Evolution, and Reengineering*, Suita, Japan, pp. 13-20, 2016.
- [12] M. Petre. “UML in practice”. *Int. Conf. Software Engineering*, San Francisco, USA , 2013, pp. 18–26.
- [13] “UML: Infrastructure specification, version 2.5”, <http://www.omg.org/spec/UML/2.5/PDF>, accessed 10 March 2018.
- [14] G. Scanniello et al. “Do software models based on the UML aid in source-code comprehensibility? Aggregating evidence from 12 controlled experiments.” *Empirical Software Engineering*, pp 1-39, 2018.

# Testing Android Applications Using Multi-Objective Evolutionary Algorithms with a Stopping Criteria

Anshuman Rohella\*, Shingo Takada\*

\*Dept. of Information and Computer Science, Keio University, Yokohama, Japan

{anshuman.rohella, michigan}@doi.ics.keio.ac.jp

**Abstract**—The ever increasing usage of Android devices and apps has created a demand for faster and reliable testing techniques. While the quality of test cases can be summed up based on the amount of code they cover, fault detection in applications is one of the main objectives for testing. We introduce an Android app testing approach which uses multi-objective genetic algorithm with elitism which finds optimal test cases by minimizing their length, maximizes the code coverage and fault detection capability, and minimizes the whole test suite for re-usability. In addition to that, we also incorporate a progress indicator which checks for improvements in test suite quality after subsequent generations and use it as a stopping criterion. The effectiveness of our approach is shown in our evaluation where it is able to perform better than the existing state-of-the-art tools.

**Keywords:** *Android Testing; Evolutionary Testing; Multi-Objective Testing*

## I. INTRODUCTION

The boom in the use of smartphones in personal and business related services has created a rapid increase in the demand for fast delivery of quality software. This demand has left Google Play Store with over 3.5 million apps, as of December 2017 [5] and this number is increasing every day. These apps are used by a huge number of people and to stay in business, the providers have to make sure that the apps are bug free and behave as planned.

Unlike a desktop application which runs as a single monolithic process, Android apps have to be a lot more flexible and a tightly coupled architecture among the components with event based control flow may make automated testing difficult and this causes heavy reliance on labour intensive manual testing [24]. Several attempts have been made in the past to automate Android Testing. Some of the techniques include random testing, model-based testing, testing with dynamic exploration etc.

Search-based software engineering techniques [15], [16] have been used in the past for GUI [11], [12], [14], as well as Android testing [10], [26] and have shown promising results. Even though code coverage is used as the basis of most of the testing tools, multiple objectives like fault detection, code coverage, shorter optimized test sequences have to be considered in order to generate optimal test suites.

We propose an Android testing approach which uses an elitist based multi-objective genetic algorithm which covers all these objectives while exploring the application under test (AUT) and creates optimal test suites. In addition to that, since search-based testing requires execution of a large number of tests, a stopping criteria is required to stop the process with enough confidence to guarantee sub-optimal solutions. Most of the existing tools set a pre-defined upper bound on either the number of test cases executed or the number of iterations (generations) for which the process will run. This may result in either pre-mature convergence or waste of CPU time. We use a progress indicator, MDR [25] to calculate the improvement of every generation and stop the process if no improvement, or degradation is found.

The remainder of this paper is organized as follows: Section II provides related works on Android testing, Section III describes the methodology as well as the architecture, Section IV evaluates our approach, Section V outlines the limitations of our approach, and Section VI concludes the paper with the future work.

## II. RELATED WORK

Google’s Android Monkey [1], a random input generator for Android apps stress tests the app in order to achieve coverage and detect faults. Due to its integration with Android development toolkit, it is widely available, is easy to use and is therefore considered the state-of-the-art testing software for Android apps [8].

Dynodroid [30] is another state-of-the-art automated testing tool [8] which uses heuristics to explore the application. It is a bit “smarter” than Monkey since it uses frequency and relevance of events to generate test sequences. In addition, it allows the user to give input, for example login info, when the search is stalled.

EvoDroid [10] is one of the first search-based testing tools for Android and is closely related to our work. It uses a call graph (obtained using MoDisco [5]) and an interface model to generate a *Population* of test cases which achieve a fitness score based on unique paths/line in the code they cover in the call graph.

Sapienz [26] is another tool that combines fuzz testing and search based exploration for testing. It uses NSGA-II [34] to maximize coverage and fault revelation while decreasing the sequence length. It considers multiple levels

of instrumentation and can be used for both white and black box testing.

### III. METHODOLOGY

We use app’s source code to generate two models. A *Screen Flow Graph* model and a *Widget-Screen Map* model. Details of these models are discussed later in this section. These two models are then used to create a population of Individuals (test cases). Finally, these test cases are then executed using the multi-objective algorithm and the process stops when the stopping criteria is met.

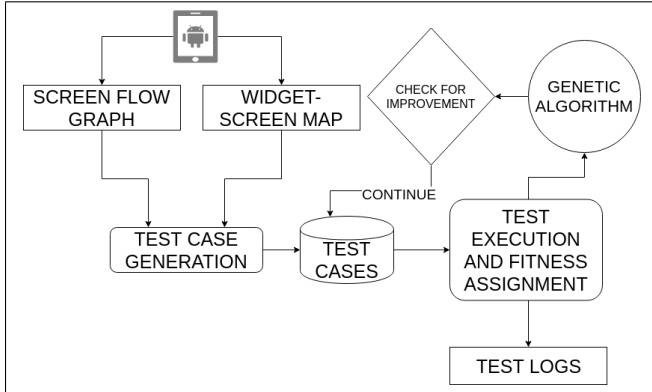


Fig. 1. The overall architecture

The overall architecture of our approach is shown in Fig. 1. Our approach can be divided into the following steps:

- Generate *Screen Flow Graph* to get the transition information of various activities, dialogues, options menu, etc.
- Generate *Widget-Screen Map* by obtaining the information about all the widgets/components available for user interaction on different screens.
- Generate test individuals based on the two models obtained earlier.
- Execute the test cases generated using the multi-objective algorithm.

#### A. Model Generation

In order to generate test cases, we generate the following two models:

- **Screen Flow Graph:** We used GATOR [29], an open source analysis toolkit which performs a static analysis on the app to capture the possible control/data-flow by tracking the event handling callbacks and window (Activities, dialogue boxes, listviews, etc) life cycle callbacks. This information is used to create a Screen Flow Graph, as shown in Fig. 2, which is then used to create test cases with *sections* (discussed later) of events valid for a particular screen.
- **Widget-Screen Map:** In an Android app, each Activity is coupled with a layout XML file which contains organized information about that activity’s components. These components include the type of layout (e.g.,

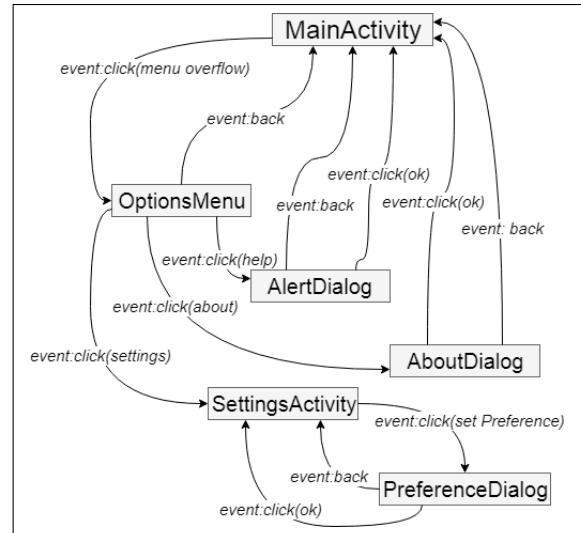


Fig. 2. An example of a Screen Flow Graph for an Android app. The nodes represent the various screens in the app and the edges represent the events that cause a transition.

*LinearLayout, RelativeLayout, etc.*), and GUI interaction widgets known as *views* (e.g., *TextView, Button, ImageView, etc.*).

Each view has a unique ID which is used in the application code’s context. We first parse the Android-Manifest XML file to list all the activities in the app. To create the *Widget-Screen Map* model, each layout file attached to these activities/screens is parsed and the obtained widgets information is mapped to their respective screens. This information is used to create events for a particular screen.

We now discuss how these two models are used to create test individuals for the initial population.

#### B. Generating Test Individuals

Individual representation is an important aspect in evolutionary algorithms as it depicts how a single solution on the whole population is encoded. Prior researches like [11] and [23] have used entire test suites as individuals wherein each individual consists of a random number of test cases (genes). This approach can be ideal for test suite minimization. However, the overall quality of each gene (test case) remains the same throughout the process because the crossover and mutation takes place at test suite level and not the test case level. Therefore, we use individual representation similar to EvoDroid [10] where each test case represents a single individual in the population.

Fig. 3 shows an example of the representation of a single individual. To create an individual, the events for the launch screen (first screen shown to the user when the app starts) section are randomly selected from the *Widget-Screen Map* and are checked against the *Screen Flow Graph* for any transitions. If the selected event causes a transition to another screen, a new section is created for that screen and events are selected randomly for that screen. This process repeats itself until the number of sections reach  $n\_sec$ , where  $n\_sec$

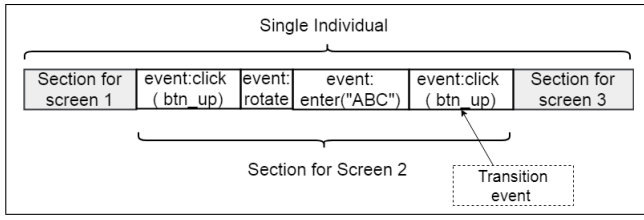


Fig. 3. Representation of a single Individual used in our approach.

is chosen randomly for each individual before its creation. If there is only one screen in the app, the test individuals contain only one section with  $n_{ev}$  random events, where  $n_{ev}$  is chosen randomly for each individual before its creation.

### C. Evolutionary Approach

A number of multi-objective evolutionary algorithms [18], [19], [20], [21], [22], [34] have shown to be quite effective in solving problems with multiple constraints/objectives. We use CBGA-ES [22] genetic algorithm in our approach to optimize test cases by selecting shorter individuals that have high code coverage and high fault revelation capability. In addition, the algorithm selects  $l$  elite solutions after the first generation which further reduces the test suite as well as the search space for subsequent generations. The main idea of the approach is to cluster the individuals with similar fitness into  $k$  clusters, sort the cluster based on *cluster dominance strategy*, and then select the individuals(elite individuals) for the best cluster as a population for next generation. The details are as follows:

1) *Clustering*: After the execution of all the individuals in the first population, each individual gets a fitness vector containing its fitness value for each objective. For example, an individual  $I_a$  will have a fitness vector  $F_{I_a} = \{fval_1, fval_2, \dots, fval_n\}$ , where  $n$  is the number of objectives.

Next,  $k$  individuals are chosen randomly and their fitness vectors are assigned as the centres of the corresponding clusters. The CBGA-ES algorithm uses Lloyd's algorithm [31] to cluster all the individuals. The next step is to sort these  $k$  clusters using *cluster dominance strategy* [22].

2) *Cluster Dominance Strategy*: Consider two clusters  $C_a$  and  $C_b$  with centroids  $m_a$  and  $m_b$ , respectively. These two clusters are checked for either dominance or partial dominance:

- Cluster Dominance:  $C_a \succ C_b$  (i.e.,  $C_a$  completely dominates  $C_b$ ) iff

$$\forall_{i=1-n} m_{ai} \leq m_{bi} \wedge \exists m_{ai} > m_{bi}$$

where  $m_{ai}$  and  $m_{bi}$  are the fitness values for the  $i^{th}$  objective in the two centroids, respectively.

- Cluster Partial Dominance:  $C_a \succeq C_b$  (i.e.,  $C_a$  partially dominates  $C_b$ ) if one of the two cases holds *true*.

Case 1.

$$n(\forall_{i=1-n} m_{ai} > m_{bi}) > n(\forall_{i=1-n} m_{ai} < m_{bi})$$

where  $n(\forall_{i=1-n} m_{ai} > m_{bi})$  is the number of fitness values of centroid  $m_a$  which are better than those in  $m_b$ .

Case 2.

$$\text{If } n(\forall_{i=1-n} m_{ai} > m_{bi}) = n(\forall_{i=1-n} m_{ai} < m_{bi}),$$

$$\left( \sum_{i=1}^n \frac{m_{ai} - m_{bi}}{m_{ai}} \times 100\% \right) > 0$$

which means that for all the objectives,  $m_a$  is able to get higher percentage of centre values than  $m_b$ .

Once the clusters have been sorted, an *elite population* of size  $l$  is created by adding the individuals from the best cluster. If the size of the best cluster is less than  $l$ , the individuals from the next best cluster are added to the elite population. This elite selection strategy reduces the search space as compared to the initial population and helps create a minimized test suite with optimal test cases. This population then undergoes *crossover* and *mutation* and proceeds to the next generation unless the stopping criteria is met.

### Crossover

Crossover is a genetic operator used in Evolutionary Algorithms to create individuals for the next generation. We use a single-point random crossover where two random points are chosen in the selected individuals and the crossover operation is performed. Test case level crossover increases the possibility of creating better test cases since the created individual contains events from the parent individuals. Fig. 4 shows an example of a crossover.

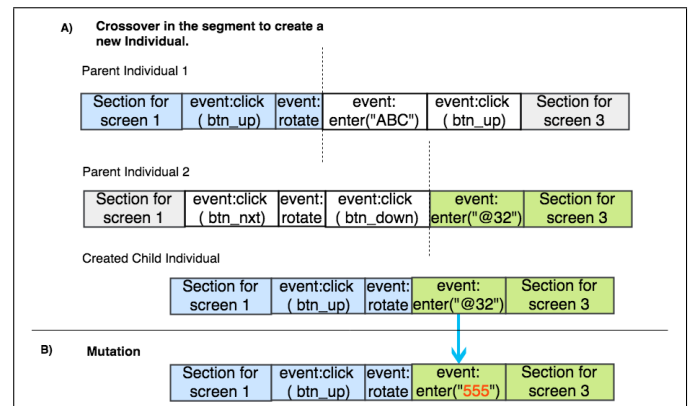


Fig. 4. Examples of crossover (A) and mutation (B).

To preserve the validity of the produced test case, crossover points are chosen in the sections which corresponds to the same screen (refer section B). A similar approach has been used in EvoDroid where they use multi-point probabilistic crossover in *segments* [10] belonging to the selected parent individuals.

## Mutation

Mutation is another genetic operator used in Genetic Algorithms which mutates/changes a gene of an offspring with some probability to maintain genetic diversity in the population and avoid getting stuck in a local minima. In our case, we apply mutation at a gene (event) level by randomly selecting a segment in the new individual created and applying one of the following mutation operations.

- **Adding/Removing:** A new event valid for that segment is added/removed at a randomly selected position in the segment. Some screens may need a specific event to be performed for a specific number of times. This mutation may help an individual (test case) to cover those cases and possibly explore a new state.
- **Changing the order:** This mutation operator changes the sequence of events in a segment randomly. Events performed in a specific order may trigger a transition to a new state. For example, a game application may need button clicks in a specific order to advance to the next stage.
- **Modifying gene properties:** This specific mutation operation is only valid for events which need some sort of input (Text, Number inputs). For example, invalid/fuzzy inputs like blank spaces, number zero and special characters may not be handled properly by an app and may throw an exception.

## Stopping Criterion

A genetic algorithm requires some criteria for termination. Some of the traditionally employed termination criteria include having an upper bound on the number of generations, giving the process a pre-defined time limit, and stopping the algorithm if there is no improvement in subsequent generations.

In our case, the number of generations and the initial population depends on the complexity of the problem. The chosen parameters may lead to either pre-mature convergence or CPU time wasted through executing redundant test cases with no improvement [9]. In addition to that, Android Emulators [2] are known to be slow and may further increase the total execution overhead.

According to previous researches [25], [27], [28], finding a stopping criteria for a multi-objective problem is a multi-objective problem itself since the improvement has to be considered for all the defined objectives. We use Mutual Domination Rate indicator (MDR) [25], a progress indicator that is specially designed for multi-objective evolutionary algorithms to check the quality of solutions after each generation. The details are as follows.

Consider  $PF_{t-1}^*$  and  $PF_t^*$  as the non-dominated (elite) solutions obtained at generation  $t-1$  and  $t$ , respectively.  $I_{mdr}(t) \in [-1, 1]$  contrasts between the number of non-dominated individuals of generation  $t$  that dominate the non-dominated individuals of generation  $t-1$  and vice-versa.

Consider a function  $C = d(A, B)$  that returns a set of elements of  $A$  that are dominated by at least one element of  $B$ , and  $|C|$  is the number of elements in the set, then,

$$I_{mdr}(PF_t^*, PF_{t-1}^*) = \frac{|d(PF_{t-1}^*, PF_t^*)|}{|PF_{t-1}^*|} - \frac{|d(PF_t^*, PF_{t-1}^*)|}{|PF_t^*|} \quad (1)$$

$I_{mdr} = 1$  indicates that generation  $t$  is completely better than  $t-1$ ,  $I_{mdr} = 0$  indicates that there has been no improvement since the last generation, and in the worst case,  $I_{mdr} = -1$  indicates that the quality of the generation has degraded. If there is no improvement for a pre-defined number of generations, the execution process stops with the last test suite considered as the optimal test suite.

## IV. EVALUATION

We establish and answer three research questions to compare our evaluation results with the study conducted by Choudhary et al. [8].

- **RQ1 (Code Coverage):** How does the code coverage achieved by our approach compare to the state-of-the-art existing approaches as mentioned in study by Choudhary et al. [8]?
- **RQ2 (Fault Detection):** How effective is our approach in finding faults?
- **RQ3 (Multiple Objective Handling):** How effective is our approach in handling the trade-off between the three objectives used for evaluation?

Creating a test oracle is another problem in the field of automated testing and has been covered in various researches [32], [33]. However, our main focus here is not the behaviour of the application or validity of the test cases, but to increase the code coverage, detect faults, reduce the test sequence length and minimize the test suite.

### A. Experimental Environment and Settings

All our experiments were done on Nexus 5's emulator with 1586 MB of RAM. The emulator was run on a 64-bit MacOS 10.12.4 machine with 2.5 GHz Intel Core i5 processors and 8 GB of RAM. We ran the evaluation on a set of open source Android applications. We set the initial population  $n$  to 100 test cases and the elite population  $l$  to 50 and the number of clusters  $k$  to 2. The mutation probability was 0.2 with 50% of the elite solutions undergoing crossover (25 solutions in our case). Since our test cases represents an individual with varying number of events in it, they cannot be compared directly to the event inputs used in DynoDroid and Monkey in terms of number of test cases executed. Hence, we evaluated our approach first, and the time spent on each app using our approach was then given to DynoDroid and Monkey to compare the performance. Since our approach is non-deterministic, we ran the evaluation 10 times and took an average of the results. We used JACOCO [4] to obtain the code coverage and all the test cases were in Espresso [3] and UI Automator [6] format.

TABLE I

ACCUMULATED CODE COVERAGE FOR OUR APPROACH (#CovE), MONKEY(#CovM) AND DYNODROID(#CovD) WHERE LOC IS THE LINES OF CODE IN THE APP AND THE EXECUTION TIME, TIME IS IN MINUTES.

AUT	LOC	#CovE	#CovM	#CovD	Time
Munchlife v1.4.2	163	93.86%	58%	76%	112
Munchlife v1.4.4	186	94.62%	63.55%	74.82%	101
BatteryCircle	251	82%	62%	79.38%	34
Triangle	281	91%	69%	81.23%	38
JustSit	276	75%	43%	66%	102
CalorieMate v1.0.0	132	84%	53%	81.22%	109
CalorieMate v1.1.0	197	87.30%	67.65%	80.27%	119
TippyTipper	996	88%	81.54%	51.33%	125
LearnMusicNotes	398	62%	50.7%	47%	117
Bats-HIIT	316	40%	24%	45%	89
BatteryDog	466	64%	51%	62.54%	34
SpriteText	1165	60.51%	58.21%	58.76%	48
SwiFTP	2160	22.8%	12.44%	16.22%	92
PasswordMaker	1436	42.96%	22%	32.22%	56
Translate	711	37.97%	19.44%	32%	78

TABLE II

FAULTS DETECTED DURING THE EVALUATION.

AUT	Faults	Exception
JustSit	1	java.lang.SecurityException
Bats-HIIT	1	java.lang.NullPointerException
CalorieMate v1.0.0	2	java.lang.IllegalArgumentException java.lang.NumberFormatException

### B. Evaluation Results and Discussion

We chose a set of 15 (2 of which were different versions of the respective apps) open source apps to compare the line coverage of our approach, Monkey, and DynoDroid.

**RQ1(Code Coverage):** As shown in Table I, our approach is able to achieve a higher coverage than Monkey and DynoDroid for most of the apps.

Some of the reasons for not achieving higher code coverage are due to dependence on external native apps (e.g., camera, contacts, messaging) and unavailability of external services on an Emulator (e.g., the app *SwiFTP* requires a WiFi connection to start a FTP server). Also, some apps may have asynchronous behaviour based on timers/thread events. For example, apps like *Bats-HIIT* (exercise timer) and *LearnMusicNotes* have features where a new state is created (register high score/personal best) only after the timer runs out. These events cannot be handled unless a specific timeout is defined, regardless of the technique used.

**RQ2(Fault Detection):** The fault finding capability of our approach is shown in Table II

Our approach was able to find a total of four new exceptions which lead to app crashes. These issues were reported to the app’s respective online repositories. These exceptions were not detected by Monkey or DynoDroid during our evaluation.

**RQ3(Multiple Objective Handling):** To answer the third research question, we compared the first generation and the last generation in terms of decrease in average length of the fault revealing test cases for all the apps in Table II.

TABLE III

DECREASE IN THE LENGTH OF FAULT REVEALING TEST CASES.

AUT	Decrease %
JustSit	74%
Bats-HIIT	82.21%
CalorieMate v1.0.0	64.12%

TABLE IV

EXECUTION TIME AND COVERAGE RESULTS FOR OPTIMAL TEST SUITES.

App Name	Coverage%	Time
JustSit	75%	16
Bats-HIIT	42%	15
CalorieMate v1.1.0	74.12%	24

The results are shown in Table III. This shows the effectiveness of our approach in choosing test sequences of shorter length if the same faults are found by multiple test cases. In addition, to check the re-usability of the optimized test suite, we ran the optimal test suites on newer versions/fixed versions of the apps. The coverages obtained on a single run are shown in Table IV.

The effectiveness of the optimal test suites on newer versions of the apps depends on the type of changes made. The test suites may have limited coverage if new features like widgets/functions are added. For example, new views were added in *CalorieMate v1.1.0* and therefore the coverage obtained is less than the coverage obtained when the app was used for normal execution (Table I).

### C. Threats to Validity

Two important threats to validity are the following:

- First, even though the evaluation apps were chose at random, our results cannot be generalized since our technique may not be applicable to all type of apps.
- Second would be the type of stopping criteria used. Even though the MDR improvement index has been found to be effective in our implementation, it still checks the improvement of the search locally, i.e., comparing  $t^{th}$  generation with  $t - 1^{th}$  generation. This may limit the whole search to a local optima.

## V. LIMITATIONS OF OUR APPROACH

This section outlines the current limitations of our tool.

### A. Static Analysis Limitations

Currently, our approach is limited to the scope of the static analysis techniques available for Android. As discussed in GATOR [29], modeling transitions caused by asynchronous events (timers and sensors) is difficult and is yet to be handled. This can lead to an incomplete model.

### B. Limited Information in Layout

Depending on the way the app is written, complete information about a View may not be obtainable. For example, a developer may choose to have views without specifying

their resource ID, or, options in a ListView may be added in through code (sometimes dynamically while the user exercises the app) rather than defining them in resource XML files. A dynamic modeling/assertion may handle these cases in a better way.

## VI. CONCLUSION AND FUTURE WORK

We proposed an approach to test Android applications which uses CBGA-ES algorithm to maximize code coverage, find short fault revealing test cases and minimize test suite for possible re-usability. In addition, we incorporate a stopping criterion to limit the testing time in case no improvement is detected over the generations. Our approach was able to achieve significantly higher coverage than current state-of-the-art tools. Also, we were able to use the minimized test suite for newer versions of three apps.

Future work includes incorporating better model generation to our approach to increase the variety of applications to which our approach can be applied to. Also, it is possible to check improvement on a global level rather than checking for local improvements after every generation. This can be done by analyzing the complexity of the application and/or predicting a global pareto front by analyzing the local improvements using MDR [25].

## REFERENCES

- [1] Android Monkey: <https://developer.android.com/studio/test/monkey.html>
- [2] Android Virtual Device: <https://developer.android.com/studio/run/managing-avds.html>
- [3] Android Espresso : <https://developer.android.com/training/testing/espresso/index.html>
- [4] JACOCO: <http://www.eclEmma.org/jacoco/>
- [5] Total Apps on Google Play: <https://www.statista.com/statistics/266210/number-of-availableapplications-in-the-google-play-store/>.
- [6] UI Automator: <https://developer.android.com/training/testing/ui-automator.html>
- [7] MoDisco: <https://www.eclipse.org/MoDisco/>
- [8] S. R. Choudhary, A. Gorla and A. Orso, "Automated Test Input Generation for Android: Are We There Yet?," 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE), Lincoln, NE, 2015, pp. 429-440.
- [9] Z. Li, M. Harman and R. M. Hierons, "Search Algorithms for Regression Test Case Prioritization," IEEE Transactions on Software Engineering, vol. 33, no. 4, pp. 225-237, April 2007.
- [10] R. Mahmood, N. Mirzaei, and S. Malek. 2014. "EvoDroid: segmented evolutionary testing of Android apps," Proc. of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2014). ACM, New York, NY, USA, 599-609.
- [11] F. Gross, G. Fraser, and A. Zeller. 2012. "EXSYST: search-based GUI testing," Proc. of the 34th International Conference on Software Engineering (ICSE '12). IEEE Press, Piscataway, NJ, USA, 1423-1426.
- [12] F. Gross, G. Fraser, and A. Zeller. 2012. "Search-based system testing: high coverage, no false alarms," Proc. of the 2012 International Symposium on Software Testing and Analysis (ISSTA 2012). ACM, New York, NY, USA, 67-77.
- [13] K. Inkumsah and T. Xie, "Improving Structural Testing of Object-Oriented Programs via Integrating Evolutionary Testing and Symbolic Execution," 2008 23rd IEEE/ACM International Conference on Automated Software Engineering, L'Aquila, 2008, pp. 297-306.
- [14] S. Carino and J. H. Andrews, "Dynamically Testing GUIs Using Ant Colony Optimization," 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE), Lincoln, NE, 2015, pp. 138-148.
- [15] M. Harman. 2007. "The Current State and Future of Search Based Software Engineering," 2007 Future of Software Engineering (FOSE '07). IEEE Computer Society, Washington, DC, USA, 342-357.
- [16] M. Harman, S. Afshin Mansouri, and Y. Zhang. 2012. "Search-based software engineering: Trends, techniques and applications," ACM Comput. Surv. 45, 1, Article 11 (December 2012), 61 pages.
- [17] M. Harman, S. Afshin Mansouri, and Y. Zhang. "Search based software engineering: A comprehensive analysis and review of trends techniques and applications," Department of Computer Science, King's College London, Tech. Rep. TR-09-03 (2009).
- [18] E. Zitzler, M. Laumanns, and L. Thiele. "SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization," Evolutionary Methods for Design, Optimization, and Control (2002): 95-100.
- [19] J. Knowles and D. Corne, "The Pareto archived evolution strategy: a new baseline algorithm for Pareto multiobjective optimisation," Proc. of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), Washington, DC, 1999, pp. 105 Vol. 1.
- [20] A. J. Nebro, J. J. Durillo, F. Luna, B. Dorronsoro, and E. Alba. , "MO-Cell: A cellular genetic algorithm for multiobjective optimization," Int. J. Intell. Syst.(2009), 24: 726-746.
- [21] E. Zitzler, K. Deb, and L. Thiele. "Comparison of multiobjective evolutionary algorithms: Empirical results," Evolutionary computation 8.2 (2000): 173-195.
- [22] D. Pradhan, S. Wang, S. Ali, T. Yue and M. Liaaen, "CBGA-ES: A Cluster-Based Genetic Algorithm with Elitist Selection for Supporting Multi-Objective Test Optimization," 2017 IEEE International Conference on Software Testing, Verification and Validation (ICST), Tokyo, 2017, pp. 367-378.
- [23] G. Fraser and A. Arcuri. 2011. "EvoSuite: automatic test suite generation for object-oriented software," Proc. of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering (ESEC/FSE '11). ACM, New York, NY, USA, 416-419.
- [24] M. E. Joorabchi, A. Mesbah and P. Kruchten, "Real Challenges in Mobile App Development," 2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement, Baltimore, MD, 2013, pp. 15-24.
- [25] J. L. Guerrero, J. Garcia, L. Marti, J. Manuel Molina, and Antonio Berlanga. 2009. "A stopping criterion based on Kalman estimation techniques with several progress indicators," Proc. of the 11th Annual conference on Genetic and evolutionary computation (GECCO '09). ACM, New York, NY, USA, 587-594.
- [26] K. Mao, M. Harman, and Y. Jia. 2016. "Sapienz: multi-objective automated testing for Android applications," Proc.s of the 25th International Symposium on Software Testing and Analysis (ISSTA 2016). ACM, New York, NY, USA, 94-105.
- [27] H. Trautmann, T. Wagner, B. Naujoks, M. Preuss and J. Mehnen, "Statistical Methods for Convergence Detection of Multi-Objective Evolutionary Algorithms," Evolutionary Computation, vol. 17, no. 4, pp. 493-509, Dec. 2009.
- [28] L. Martí, J. García, A. Berlanga and J. M. Molina, "A progress indicator for detecting success and failure in evolutionary multi-objective optimization," IEEE Congress on Evolutionary Computation, Barcelona, 2010, pp. 1-8.
- [29] S. Yang, H. Zhang, H. Wu, Y. Wang, D. Yan and A. Rountev, "Static Window Transition Graphs for Android," 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE), Lincoln, NE, 2015, pp. 658-668.
- [30] A. Machiry, R. Tahiliani, and M. Naik. 2013. "Dynodroid: an input generation system for Android apps," Proc. of the 2013 9th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2013). ACM, New York, NY, USA, 224-234.
- [31] S. Lloyd, "Least squares quantization in PCM," IEEE Transactions on Information Theory, vol. 28, no. 2, pp. 129-137, March 1982.
- [32] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz and S. Yoo, "The Oracle Problem in Software Testing: A Survey," IEEE Transactions on Software Engineering, vol. 41, no. 5, pp. 507-525, May 1 2015.
- [33] G. Fraser and A. Zeller, "Mutation-Driven Generation of Unit Tests and Oracles," IEEE Transactions on Software Engineering, vol. 38, no. 2, pp. 278-292, March-April 2012.
- [34] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182-197, Apr 2002.



# An Empirical Study on the Impact of Android Code Smells on Resource Usage

Johnatan Oliveira<sup>1</sup>, Markos Viggiano<sup>2</sup>, Mateus Santos<sup>1</sup>,  
Eduardo Figueiredo<sup>2</sup>, Humberto Marques-Neto<sup>1</sup>

<sup>1</sup>Department of Computer Science, Pontifical Catholic University of Minas Gerais (PUC Minas)

<sup>2</sup>Department of Computer Science, Federal University of Minas Gerais (UFMG)

Belo Horizonte, Brazil

{johnatan.oliveira,mateus.freira}@sga.pucminas.br, {markosviggiano, figueiredo}@dcc.ufmg.br,  
humberto@pucminas.br

## Abstract

*Code smells are symptoms that something may be wrong with the app. Aiming at removing code smells and improving the maintainability and performance of the app, we may apply the refactoring technique, which could reduce hardware resource use, such as CPU and memory. However, a few studies have evaluated the impacts of the refactoring in Android. This paper presents a study to assess the effects of smartphone resource use caused by refactoring of 3 classic code smells: God Class, God Method, and Feature Envy. To this purpose, we selected 9 apps from GitHub. The results show that refactoring used in desktop software may not be appropriate for Android apps. For example, the refactoring of God Method had increased CPU consumption by more than 47%, while the refactoring of the 3 code smells reduced memory consumption in average 6.51%, 8.4%, and 6.37%, respectively, in one app. Our results can support the community in conducting research and future implementation of new tools. Also, it guides app developers in refactoring and thus improving the quality of their apps.*

**keywords:** Code Smells, Android Smells, and Consumption of Smartphone Resources

## 1 Introduction

In recent years, mobile applications (apps) have become one of the most popular and profitable software products in society, especially context-aware apps that are pervasive in people's lives [10]. Applications developed for the Android platform are mainstream [13]. This platform has around 80% of the market for mobile Operating Systems (OS), and it has far surpassed its main competitor Apple [10]. One of the critical factors for the success of this software segment is related to the ease of rapidly developing and making the apps available to millions of users with lower costs [12].

Mobile app development is different from desktop software development since the mobile devices, like smartphones and tablets, have limited resources, such as CPU, battery, and memory [10]. Furthermore, app development is affected by deadlines even more restricted than those for desktop, once apps are usually designed for many users that

get used to regular addition of new features and high speed in the correction of flaws [1, 7]. Mobile apps of context-aware can identify the context in which they are inserted and adapt their behavior according to the environment [4, 12].

When these apps are erroneously programmed, they can quickly drain device resources, such as the memory and CPU [13]. The presence of code smells can lead to poor software quality, which makes the evolution of features harder, deteriorated quality, and therefore, causes a bad experience for the final user [9]. In the literature, the presence of these issues in Android apps is known as Android smells [3].

Performance and optimization of the resources are crucial factors for the success of mobile apps [10]. Users can uninstall the app from the device if it starts to lock-in or drains resources quickly [3, 10]. Therefore, correction of Android smells can not only improve the performance of the apps without affecting their behavior and also improve user experience [3, 9, 10]. For this reason, applying refactoring techniques in Android smells could contribute to the evolution and maintenance of these apps.

Previous works [1, 2, 10] investigated the impact of resources usage, such as memory, battery, and CPU. However, we still lack empirical study on the effect of fixing Android smells through refactoring, mainly concerning memory and CPU usage. Studies regarding automatic detection and analysis of the impact of the refactoring on the Android platform are still premature [5].

This paper starts to fill this gap by analyzing the impacts caused by the adoption of refactoring in the Android platform. For this purpose, we selected 100 mobile apps from GitHub and filtered this data set to choose the most representative apps, because the focus this paper are context-aware apps. After the filtering process, 9 apps remained to be analyzed. To detect code smells in the Android platform, we rely on JDeodorant tool, and 3 classic code smells for analysis: God Class, God Method, and Feature Envy. We believe our results might better support the community in conducting researches about Android smells and also guide app developers in refactoring activity, aiming to improve the quality of the mobile apps.

## 2 Background

This section presents an overview of the main concepts used in this paper. Section 2.1 introduces the 3 selected code smells. Section 2.2 describes some the tools able detect and refactor code smells.

### 2.1 Code Smells

A code smell is any symptom that may indicate a deeper quality problem in the software [9]. The code smells documented by Fowler [9] are considered classic in object-oriented software. In this study, we evaluated 3 types of code smells, God Class, God Method and Feature Envy, mainly because of two reasons. First, these code smells are classic in software engineering. Second, several tools are able to detect these code smells. However, a few tools can detect and automatically apply refactoring. Next, we present the selected code smells.

*God Class* occurs when a class has many attributes or methods in its interface, and it does not use all of them [9]. Usually, in this kind of situation, it is possible to extract part of these attributes or methods to another class, thus separating responsibilities and leaving them more coherent [11]. *God Method* can be described as a long method or a method that does more than a task. Therefore, it is not related only to the size of the method itself [9]. This code smell might get solved by creating smaller methods and moving code from the main method to others, keeping the same behavior [11]. *Feature Envy* is a code smell that occurs when a method uses more attributes or methods from another class than from its own class [9]. The suggested refactoring for this code smell is to move the envy method, replacing then a large number of calls to the other class by only one call to the moved method [9].

### 2.2 Automated Code Smell Detection and Refactoring

Several tools have been developed to automate the process of detecting code smells. For instance, PMD [8], which is a general purpose tool for code smells detection in some languages, such as Java and JavaScript. However, a few tools are able to apply refactoring technique automatically. Therefore, we selected JDeodorant tool because it is able to detect and apply the refactoring technique automatically to the 3 kinds of code smells selected.

Some tools are specific to Android platform, such as aDoctor [14]. Rigid Alarm Manager, Durable Wake lock, and Debuggable Release are some examples of code smells detected by aDoctor. These code smells would not occur on other platforms because they refer to Android-specific problems. We have not analyzed these Android-specific code smells because our goal is to target well-known code smells as discussed in Section 2.1. Besides, aDoctor cannot perform refactoring automatically.

## 3 Study Settings

This section describes the evaluation settings. Section 3.1 presents the study goal and the research questions. Section 3.2 presents the evaluation steps. Section 3.3 presents our data set.

### 3.1 Goal and Research Questions

The primary goal of this study is to evaluate whether refactoring in Android improves the source code concerning the use of mobile devices resources. In particular, we aim to verify if this technique can reduce the consumption of CPU and memory. Based on this goal, we also conceived the following research questions (RQs) to guide our study.

**RQ1.** *Does the refactoring of Android code smells improve the CPU consumption of the Smartphone?*

**RQ2.** *Does the refactoring of Android code smells improve the memory consumption of the Smartphone?*

### 3.2 Evaluation Steps

We analyze the impacts of applying the refactoring in Android from a set of 9 apps. To minimize the risk of bias in our study, we executed each app 18 times, and we use the arithmetic mean (and its standard deviation) to evaluate the impact of refactoring. The executions were conducted 3 times for each code smell before applying the refactoring (resulting in  $3 \times 3 = 9$  executions) and 3 times for each code smell after using the refactoring, i.e., a total of 18 performances for each app. Since we ran each app 18 times and given that we have 9 apps, we performed a total of 162 executions. Due to the high number of executions and the fact that all tests are performed manually, it would not be feasible to investigate a higher number of apps at this moment, and therefore we kept our data set with the selected apps.

To performer our study, we used a smartphone running Android OS version 5.1, model Moto G XT1032, equipped with a quad-core CPU of 1.2 GHz. The smartphone consists of 1 GB DDR3 of RAM and 16 GB of disk. The original OS of this smartphone was Android 4.3, called Jelly Bean, with later updates to 5.1. We considered this smartphone suitable for our experiments, mainly because there is a lot of apps compatible with it from Google Play Store<sup>1</sup>. We used in our work a smartphone with only essentials apps of the Android OS to avoid interference in the results of other apps. Moreover, for each execution, the app was uninstalled and installed again with the Android APK of the version under analysis. By following this procedure, all user data were erased at the beginning of a new test, and each execution of the test had a similar initial state. In our study, we performed five steps to analyze the adopted refactoring. These steps are illustrated in Figure 1 and described as follows.

**(1) Code smell detection** – We run the selected tool called JDeodorant to detect the 3 types of code smells. In

<sup>1</sup><https://play.google.com/store>

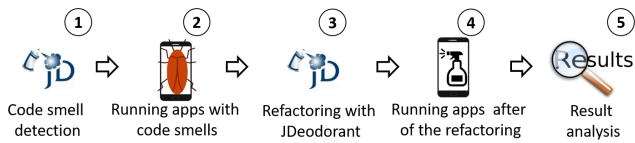


Figure 1: Evaluation Steps

this step, each app was imported into Eclipse, because the detection tool is a plug-in exclusive for this IDE. After importing these apps, we identified the 3 types of code smells in each app. We create a list of classes, methods, and code smells of each analyzed app.

**(2) Running apps with code smells in Android** – From the data obtained in the previous step, we run each app individually, 18 times on the smartphone. At the end of the 18 tests per app, we achieve the results of memory use and CPU use through the mean of 3 executions in each app. We collect the data about use of CPU and memory through an app named *AnotherMonitor*, version 3.1.0. *AnotherMonitor* was selected for several reasons. First, this app is compatible with the Android version evaluated. Second, it has an excellent reputation in Play Store with more than 4 stars. Third, it is possible to export the results about CPU and Memory usage to a CSV file.

**(3) Refactoring using JDeodorant** – After the previous steps, we obtained the consumption of the apps in the smartphone, such as CPU and memory. In this step, we re-import the apps to Eclipse, similar to Step 1, to apply the automated refactoring with JDeodorant to the 3 selected code smells.

**(4) Running of the Apps after of the Refactoring** – After applying the automated refactoring using the JDeodorant tool, we evaluated whether the consumption of memory and CPU improved or deteriorated. These characteristics of usage can be assessed based on the access provided by Android. To this goal, we run again each app 18 times and compute the average memory and CPU consumption.

**(5) Result analysis** – At the end of all the previous steps, we obtained the data regarding the consumption of resources in Android before and after refactoring. Through these results, it is possible to compare the feasibility of this technique in mobile devices.

### 3.3 Corpus of Android Apps

In order to investigate the impacts caused by Android smells and refactoring techniques on resource consumption, we chose only apps from the context-aware domain for several reasons. First, there are many apps available for download on GitHub. Second, these apps are actively used. Also, the authors of this paper believed that it would be easy to find code smells in this app domain. The reason might be because this type of app can contain source code with high complexity caused by the use of several sensors to identify characteristics of the context [16]. Third, one of the features

of this type of app is the need to use various hardware resources to identify the context, such as GPS and movement sensors [6].

The apps that compose our data set were retrieved from GitHub in November 2017. We searched for apps of context-aware sorted by stars. To retrieve apps, we used the following keywords related to the context-aware domain: *context-aware*, *context awareness*, and *pervasive computer*.

To minimize the risk of biasing our results, we apply a strict set of criteria for defining our data set as illustrated in Figure 2 and described by the three phases as follows.



Figure 2: Phases for Collecting Apps from GitHub

**Phase 1: Preliminary Search** – We performed a preliminary search in GitHub to evaluate the feasibility of collecting these apps of the context-aware domain. We conducted this phase manually to identify the diversity of apps on GitHub. Also, this pre-evaluation was necessary to obtain the domain variation names.

**Phase 2: Automated Download** – We implemented an algorithm to clone the apps from GitHub automatically. This phase is necessary because we know that several apps it are hosted on GitHub, and manual cloning would be unfeasible. In this phase, we obtained 100 apps.

**Phase 3: Filtering** – From the cloned apps, we used the following exclusion criteria to filter these apps: i) non-Java app, once the chosen tool requires apps developed in Java, ii) apps with less than 1k lines of code (LOC), because we considered that these apps might represent only toy apps, iii) apps that are not compatible with Eclipse, since the selected tool requires the use this IDE, and iv) all apps that required login due to evaluation convenience. After applying all these filters, we obtained 9 apps able to be evaluated.

## 4 Results and Discussion

This section discusses the results obtained in this study. Section 4.1 presents of amount of code smells detected in each app. Section 4.2 focuses on answering each research question. Section 4.3 presents some guidelines to support future implementation of tools.

### 4.1 Detection of Code Smells

In this section, we present the results about code smells detected by JDeodorant tool in our data set. Table 1 shows the occurrences for each code smell investigated in this study. In the last column of Table 1, we show the total number of code smells for each app, and the last line presents the total number of occurrence of each code smell across all

the apps. By looking at the column Total, we note that RunnerUp was the app with the highest number of code smells. Besides, in the line Total, it is possible to identify that the code smells Feature Envy and God Class occurred most frequently, with 102 and 94 occurrences, respectively. For instance, Feature Envy appeared 34 times in RunnerUp and 27 times in Activity Tracker, while God Class showed up 50 times in RunnerUp and 15 times in Calendula.

Table 1: Code Smells Detected

App	God Class	God Method	Feature Envy	Total
Activity Tracker	4	0	27	31
Calendula	15	8	14	37
CycleStreets	13	2	8	23
Forecastie	3	0	1	4
NoiseApp	0	1	1	2
Pedometer	1	2	3	6
RunnerUp	50	16	34	100
Steptastic	5	2	3	10
Travel-Mate	3	1	11	15
<b>Total</b>	<b>94</b>	<b>32</b>	<b>102</b>	

## 4.2 Answering the Research Questions

In a first moment, we present the results regarding the CPU use. Therefore, we answer RQ1 as follows. *Does the refactoring technique of Android code smells improve the CPU consumption of the Smartphone?*

Tables 2 to 4 present some descriptive statistics of the experiments results. The column *Mean* ( $\bar{x}$ ) presents the mean of the analyzed data. A *Standard Deviation* is represented by  $\sigma$ . Also, all columns have letter S or R. S means app with code smells, and R stands refactored apps

Table 2 shows a particular case of the one app named CycleStreets that before refactoring the average use of CPU was 26.2% and it started to use 32.47% of CPU after the refactoring. The consumption of CPU increased by more than 6%. This number may seem somewhat small, but we are investigating code smells on a platform that already has limited resources, and any unnecessarily consume harmfully. It is also possible to observe that from 9 apps evaluated, the consumption of CPU increased in 7 apps and drastically decreased in only one app (Steptastic). Also, the standard deviation changed from 11.64 to 14.11 in CycleStreets, an increase of 2.47%.

Table 2: Descriptive Statistics for CPU (God Class)

App	$\bar{x}$ (S)	$\bar{x}$ (R)	$\sigma$ (S)	$\sigma$ (R)
Activity Tracker	17.07	20.01	6.42	4.8
Calendula	16.35	17.44	7.74	9.2
CycleStreets	26.2	32.47	11.64	14.11
Forecastie	17.07	20.01	6.42	4.8
NoiseApp	22.55	23.65	12.66	12.54
Pedometer	19.47	21.17	9.75	10.43
RunnerUp	14.26	17.33	7.34	9.42
Steptastic	24.77	15.12	13.23	9.27
Travel-Mate	19.7	19.61	10.01	10.34

Regarding God Method, Table 3 shows the results achieved. In this table, we may highlight the Forecastie app,

in which the consumption of CPU almost doubled. As we can observe, the use of CPU in this app soared from 25.43% to 47.14%. However, for the app Steptastic, it was possible to reduce CPU consumption by more than 12% through the refactoring technique. Finally, refactoring God Method caused an increase in CPU consumption in 5 apps and a decrease in 3 apps.

Table 3: Descriptive Statistics for CPU (God Method)

App	$\bar{x}$ (S)	$\bar{x}$ (R)	$\sigma$ (S)	$\sigma$ (R)
Activity Tracker	26	30.93	14.68	16.82
Calendula	20.97	26.08	13.23	15.51
CycleStreets	25.29	34.01	14.23	20.35
Forecastie	25.43	47.14	15.72	12.87
NoiseApp	26.49	26.24	16.18	15
Pedometer	32.98	28.17	19.3	16.3
RunnerUp	19.32	26.42	10.37	15.9
Steptastic	30.87	18.16	18.09	12.11
Travel-Mate	22.35	17.77	12.64	11.08

Table 4 presents the results for Feature Envy. In all apps, but Travel-Mate it was possible to improve source code quality through refactoring technique. Overall, it was possible to improve the source code quality with an average increase of less than 1% of use of resources.

Table 4: Descriptive Statistics for CPU (Feature Envy)

App	$\bar{x}$ (S)	$\bar{x}$ (R)	$\sigma$ (S)	$\sigma$ (R)
Activity Tracker	16.82	16.88	8.11	8.14
Calendula	19.14	20.82	10.35	9.33
CycleStreets	21.52	23.09	10.94	11.61
Forecastie	23.24	24	12.28	11.6
NoiseApp	20.02	22.86	10.55	12.11
Pedometer	25.13	23.84	12.52	13.07
RunnerUp	21.29	21.56	11.89	11.89
Steptastic	21.55	21.68	11.54	12.04
Travel-Mate	16.35	17.44	7.74	9.2

We believe that the refactoring technique adopted by JDeodorant tool and as indicated by Fowler [9] is not appropriate to use in Android since most apps consumed more CPU after we performed refactoring. A potential reason for the increase can be found in the Android's site of the excellent programming practice<sup>2</sup>. For instance, when there are several recursive calls, it is necessary to call the onDestroy() method more times, which may increase CPU usage since it will be required to destroy more objects.

The refactoring adopted to solve this problem is Extract Method. This refactoring consists in splitting the God Method and creating a new method [9]. Therefore, applying the refactoring Extract Method may cause the app to invoke these resources several times, increasing CPU consumption. After presenting the use of CPU of each app under analysis, we show the results concerning the memory consumption. Therefore, we answer RQ2 as follows. *Does the refactoring technique of Android code smells improve the memory consumption of the Smartphone?*

<sup>2</sup><https://developer.android.com/training/best-performance.html>

In general, the adoption of the refactoring increased the memory usage, but in some cases, it can help to avoid unnecessary use of resources. We believe that the refactoring technique in the case of the app Steptastic decreases the memory usage because of two main reasons. First, Steptastic, for example, is smaller regarding LOC and their methods are well distributed among classes. Second, this app has only one developer, which may be a indicate that the number of developers may be correlated with the number of code smells. Tables 5 to 7 presents the same configurations presented in RQ1.

Table 5 presents a similar result of Table 4, but this table shows the results regarding the code smell God Class and usage of memory. For 7 apps, namely: Calendula, CycleStreets, Forecastie, NoiseApp, Pedometer, RunnerUp, and Steptastic, it was possible to reduce the use of memory by 26, 26, 58, 13, 13, 7, and 33, respectively. These values were measured in megabyte (MB).

Table 5: Descriptive Statistics for Memory (God Class)

App	$\bar{x}$ (S)	$\bar{x}$ (R)	$\sigma$ (S)	$\sigma$ (R)
Activity Tracker	9.54	67.6	2.29	41.01
Calendula	60.45	34.24	17.89	12.55
CycleStreets	73.8	47.08	9.28	9.23
Forecastie	9.54	67.6	2.29	41.01
NoiseApp	145.22	159.02	24.29	21.09
Pedometer	144.82	158.24	24.63	20.86
RunnerUp	88.53	80.66	17.48	8.5
Steptastic	61.99	28.53	13.71	14.72
Travel-Mate	80.65	83.86	39.64	39.21

Table 6 shows the results of the use of memory concerning God Method. Overall, it is possible to observe that the results demonstrate an increase in memory usage after refactoring. Also, in this table, there is a specific case in which the app (Travel-Mate) used 98.39 MB before we apply refactoring and started to consume 625.58 MB after the refactoring. This number represents an increase of approximately 500 MB, i.e., the app began to use an alarming quantity of memory. Also, the app Activity Tracker had a high disparity regarding the standard deviation.

Table 6: Descriptive Statistics for Memory (God Method)

App	$\bar{x}$ (S)	$\bar{x}$ (R)	$\sigma$ (S)	$\sigma$ (R)
Activity Tracker	94.38	268.38	22.17	148.84
Calendula	51.95	48.43	9.44	13.19
CycleStreets	74.1	47.91	10.65	11.05
Forecastie	93.49	41.46	15.36	0.04
NoiseApp	144.73	158.2	24.67	21
Pedometer	74.95	171.91	17.65	21.61
RunnerUp	87.38	81.09	17.49	9.33
Steptastic	61.95	29.89	13.48	15.45
Travel-Mate	98.39	625.58	28.34	13.64

In general, all descriptive statistics for memory showed that standard deviation varied too much, except for code smell Feature Envy. Table 7 presents, in general, a lower variation in relation to the standard deviation. A low standard deviation shows that the data are clustered tightly

around the mean. Besides, in general, all apps started to use more memory after the refactoring, such as the Activity Tracker app.

Table 7: Descriptive Statistics for Memory (Feature Envy)

App	$\bar{x}$ (S)	$\bar{x}$ (R)	$\sigma$ (S)	$\sigma$ (R)
Activity Tracker	94.82	264.83	21.49	150.22
Calendula	77.5	98.59	21.35	38.26
CycleStreets	62.51	46.78	22.15	11.26
Forecastie	93.26	41.43	14.63	0.02
NoiseApp	143.39	158.69	23.85	20.93
Pedometer	74.33	171.7	17.92	22.2
RunnerUp	87.86	80.42	18.28	9.17
Steptastic	61.23	28.36	13.58	15.22
Travel-Mate	60.45	34.24	17.89	12.55

### 4.3 Guidelines for Android Smell Tools

From this study, we uncover 2 guidelines (G1 and G2) to support future implementation of tools for Android.

**G1.** In order to support results analysis through statistical methods, graphical visualization or at least numerical indicator, such as percentage – We identified that the evaluated tool does not show statistical numbers related to the code smells for each type. Also, when using the detection and refactoring tool in apps, we should be able to determine if the refactoring will lead to more methods calls. Then, if more method calls will be necessary, applying the refactoring will probably imply in increasing the consumption as the OS adds more data in the Android memory stack.

**G2.** In order to combine different techniques (token, tree, etc.) [14, 8, 15] for detecting several codes smells in Android – It could be interesting to combine different techniques that can be more useful to detect code smells. This characteristic may increase the precision of the tool, mainly in refactoring technique. A solution to this problem is to identify the number of cycles of the clock that refactoring can cause.

## 5 Related Work

Studies have investigated code smells in Android [1, 2, 10, 13]. For instance, Boussaa et al.[2] proposed an automated approach to generate rules based on software quality metrics and threshold to detect code smells in Android. This work argues that identifying Android code smells is extremely important, since the presence them may lead to higher use of CPU, memory, and battery. We also found other of papers related to refactoring and energy efficiency, considering a different type of refactoring. For instance, Banerjee and Roychoudhury [1] present a lightweight refactoring technique to assist app development regarding energy efficiency. Their results show that refactoring the app can reduce the energy consumption up to 29%.

Existing studies investigate the identification of code smells in Android and how they relate to energy efficiency. However, works to identify positive and negative impacts of refactoring on code smells regarding resources usage are

premature. Furthermore, there is no study investigating both memory and CPU usage before and after applying the refactoring. In this context, our study investigates how the refactoring of code smells usually common on desktop systems may impact resources usage in Android.

## 6 Threats to Validity

We based our study on related works to support our research. However, some threats to validity may affect our research findings. We conducted careful filtering of context-aware apps from GitHub. Considering that the exclusion criteria for app selection were applied in an automatic process, we may have discarded compatible apps. In our viewpoint, the selected apps are representative, given that they are well-defined regarding the diversity of use of resources.

We used defaults configuration of JDeodorant to detect code smells and apply refactoring techniques automatically. Besides, another threat is concerning the use of the apps. We decided not to emulate them through Android Studio to achieve more trusted results. To test these apps, we have adopted a rigorous manual test script, whereby all apps have been tested. For example, if the app requires GPS positioning and the user walks with the smartphone, all other apps with the same characteristics have passed by the same route without any interference with the time or distance covered.

## 7 Conclusion and Future Work

The empirical study reported in this paper evaluated 9 different apps from the context-aware domain. These apps were tested 162 times. In particular, we analyzed 3 types of smells, namely God Class, God Method, and Feature Envy. We aimed to verify the possibility of improving the quality of the source code through refactoring technique and reduce the use of resources, such as memory and CPU.

Our findings point that the refactoring technique, in general, causes an increase in the use of resources. We also observed that the JDeodorant is not able to adequately perform the refactoring in Android since techniques usually adopted in desktop software are the opposite to the best-practices indicated by Android developers site. The tool used in this study was developed with the purpose of detecting the code smells and apply refactoring in Java, the same language used in the tested apps.

We believe that the results of this study will benefit developers by helping them to avoid inappropriate use of refactoring technique in the mobile device. Additionally, we provide two guidelines for developing a new tool able to detect code smells and apply refactoring. As future work, we plan to extend our study to investigate other code smells on Android. We also plan to develop an automated script to test different apps in large-scale studies and implement a tool based on the guidelines uncovered in this study.

## 8 Acknowledgments

This research was partially supported by Brazilian funding agencies: CAPES, CNPq (Grant 424340/2016-0), FAPEMIG (Grant PPM-00651-17 and APQ-02924-16), and FIP-PUC Minas.

## References

- [1] Abhijeet Banerjee and Abhik Roychoudhury. Automated refactoring of Android apps to enhance energy-efficiency. In *38th Proc. of the Int'l Conf. on Mobile Software Engineering and Systems (MOBILESoft)*, 2016.
- [2] Mohamed Boussaa, Wael Kessentini, Marouane Kessentini, Slim Bechikh, and Soukeina Ben Chikha. *Competitive Coevolutionary Code-Smells Detection*. Springer Berlin Heidelberg, 2013.
- [3] A. Carette, M. A. A. Younes, G. Hecht, N. Moha, and R. Rouvoy. Investigating the energy impact of Android smells. In *24th Proc. of the Int'l Conf. on Software Analysis, Evolution and Reengineering (SANER)*, 2017.
- [4] L. Cruz and R. Abreu. Performance-based guidelines for energy efficient mobile applications. In *4th Proc. of the Int'l Conf. on Mobile Software Engineering and Systems (MOBILESoft)*, 2017.
- [5] L. Cruz, R. Abreu, and J. N. Rouvignac. Leafactor: Improving energy efficiency of Android apps via automatic refactoring. In *4th Proc. of the Int'l Conf. on Mobile Software Engineering and Systems (MOBILESoft)*, 2017.
- [6] Quan Chau Dong Do, Guowei Yang, Meiru Che, Darren Hui, and Jefferson Ridgeway. Mybatrecommender: Automated optimization of energy consumption for android smartphones in software layer. In *13th Proc. of the Int'l Conf. on Software Engineering and Knowledge Engineering (SEKE)*, 2016.
- [7] Jacky Estublier. Software configuration management: A roadmap. In *22nd Proc. of the Conf. on The Future of Software Engineering (ICSE)*, 2000.
- [8] F. A. Fontana, M. Zanoni, A. Marino, and M. V. Mntyl. Code smell detection: Towards a machine learning-based approach. In *29th Proc. of the Int'l Conf. on Software Maintenance (ICSM)*, 2013.
- [9] Martin Fowler. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, 1999.
- [10] Geoffrey Hecht, Naouel Moha, and Romain Rouvoy. An empirical study of the performance impacts of Android code smells. In *3th Proc. of the Int'l Conf. on Mobile Software Engineering and Systems (MOBILESoft)*, 2016.
- [11] Michele Lanza and Radu Marinescu. *Object-Oriented Metrics in Practice: Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems*. Springer Publishing Company, Incorporated, 2010.
- [12] U. A. Mannan, I. Ahmed, R. A. M. Almurshed, D. Dig, and C. Jensen. Understanding code smells in Android applications. In *4th Proc. of the Int'l Conf. on Mobile Software Engineering and Systems (MOBILESoft)*, 2016.
- [13] R. Morales, R. Saborido, F. Khomh, F. Chicano, and G. Antoniol. Anti-patterns and the energy efficiency of Android applications. *ArXiv e-prints*, 2, 2016.
- [14] F. Palomba, D. Di Nucci, A. Panichella, A. Zaidman, and A. De Lucia. Lightweight detection of android-specific code smells: The adocor project. In *24th Proc. of the Int'l Conf. on Software Analysis, Evolution and Reengineering (SANER)*, 2017.
- [15] Santiago A. Vidal, Claudia Marcos, and J. Andrés Díaz-Pace. An approach to prioritize code smells for refactoring. *Automated Software Engineering*, 23, 2016.
- [16] H. Zhu, H. Xiong, Y. Ge, and E. Chen. Discovery of ranking fraud for mobile apps. *IEEE Trans. on Knowledge and Data Engineering*, 27, 2015.

# Mining Intentions to Improve Bug Report Summarization

Beibei Huai, Wenbo Li, Qiansheng Wu, Meiling Wang  
University of Chinese Academy of Sciences

Institute of Software, Chinese Academy of Sciences

National Engineering Research Center of Fundamental Software, Beijing, 100190, China

{beibei, wenbo, qiansheng, meiling}@nfs.iscas.ac.cn

**Abstract**—In recent years, various automatic summarization techniques have been proposed to extract important information from bug reports. However, existing techniques mainly focus on common text features and ignore human intentions implied in bug reports. In fact, each bug report generally contains multiple intentions which are distributed in different sentences. Bug report readers are usually more interested in content that contains intentions of certain categories (e.g. fix solution, bug description). Based on the above observation, we introduce an intention taxonomy and implement the intention classification algorithm in this paper. Furthermore, we propose a new Intention-based Bug Report Summarization approach, namely IBRS, which leverages intention taxonomy to enhance bug report summarization. We evaluate our approach on Intention-BRC corpus and the experimental result shows that IBRS outperforms the state-of-the-art approaches in terms of precision, recall, F-score, and pyramid precision.

**Index Terms**—Bug Report, Text Summarization, Intention Taxonomy, Intention Mining

## I. INTRODUCTION

A software project's bug report repository provides a rich source of information for a software developer working on the project. A bug report is composed of a title, descriptions and comments from several developers. We analyse 40,000 bug reports from different open source projects and find that bug reports are usually very long (more than 10 comments in one bug report on average) and their content is mixed with code and debugging information. An example bug report is shown in Figure 1. It is part of the #434108 bug report of Eclipse which contains 14 comments. In such a long bug report, it is time-consuming for a reader to grasp the important information they need.

One of the most effective ways to save bug report readers' time is to provide them with summaries of bug reports. The main idea of the state-of-the-art bug report summarization techniques is to extract useful sentences in bug reports. Rastkar et al. [1] first proposed BRC model. They marked 36 bug reports (BRC corpus) and trained 3 classification models on BRC corpus, meeting corpus and email corpus to score each sentence in a bug report. Finally, they found that the result is sensitive to the type of corpus. It suggests that the text of bug reports has unique characteristics compared with other common texts. Mani et al. [2] used an unsupervised method.

**Bug 434108- [Perspectives] Copy Workbench Layout option does not work in Eclipse 4.x**  
**Status:** ASSIGNED      **Product:** Platform      **Component:** UI (show other bugs)  
**Version:** 4.4      **Hardware:** PC Windows 7      **Importance:** P3 normal (vote)  
**Assignee:** Bartosz Popiela CLA Friend  
**Reported:** 2014-05-05 08:31 EDT by Wojciech Sudol CLA Friend  
**Modified:** 2017-05-26 09:23 EDT (History)  
**Description** Wojciech Sudol CLA Friend 2014-05-05 08:31:14 EDT  
Scenario:  
1. Run Eclipse with a fresh workspace  
2. Close Welcome page  
3. Move and resize some views  
4. From the main menu choose: File > Switch Workspace > Other...  
5. In the "Workspace Launcher" dialog provide a path to a new workspace; select "Copy Settings" > "Workbench Layout" option and press OK.  
6. After restart close Welcome page and compare new layout with the previous one.  
Expected behaviour:  
New workbench have the same layout as the old one.  
Current behaviour:  
New workbench have a new, default layout.  
It was working in 3.x, does not work in 4.x.  
**Comment 1** Eric Moffatt CLA Friend 2014-05-05 15:14:06 EDT  
Wojciech, this may not be perfect but it's at least working somewhat. I just opened a new workspace from my existing dev environment and it seems to have moved everything over (at least I started with both a Java and Debug perspective open and my EGit views where I expected them... This is in  
**Comment 2** Eric Moffatt CLA Friend 2014-05-05 15:16:28 EDT  
I tested this on M7 and it's certainly copying most of the environment. I did the 'switch' from my regular dev workbench and the new workspace came up with both a Java and a Debug perspective and my EGit views where where I expected them.  
What system are you on ?  
**Comment 3** Wojciech Sudol CLA Friend 2014-05-05 15:38:37 EDT  
My OS is Windows 7 x64.  
I was testing it again and I see that it works if you switch to already existing workspace, but not for new workspaces. In 3.x it works in both cases.

Fig. 1. A Bug Report Sample from Eclipse.

They classified the sentences roughly and dropped two kinds of the sentences (Question and Code). Finally they applied several unsupervised methods such as Grasshopper, Diverse Rank to get summaries of bug reports. The sentences classification in their paper plays a role as a noise reduction mechanism. Moreover, the concept of intention has been applied to various software artifacts, e.g. app reviews [3], development emails [4]. We observe that sentences posted in bug reports also contain different purposes.

In this paper, we define 7 intention categories in bug reports and then explore the connection between summaries and sentence intentions in bug reports. Finally we improve the bug report summarization approach by taking sentences intentions into account. This paper mainly makes following 3 contributions:

- 1) We introduce an taxonomy of intentions in bug reports,

which classifies intentions into seven categories: *Bug Description*, *Fix Solution*, *Opinion Expressed*, *Information Seeking*, *Information Giving*, *Meta/Code* and *Emotion Expressed*.

- 2) We propose a mixed model of linguistic patterns matching and machine learning to classify the sentences of a bug report by their intentions.
- 3) We propose an intention-based bug report summarization approach (IBRS), which takes advantage of the informations that intentions implies.

The rest of this paper is organized as follows. Section II describes the related work. Section III gives an overview of IBRS approach. In Section IV, we describe the experiments we conducted and the evaluation results. We present threats to validity and future work in Section V. And we concludes our work in Section VI.

## II. RELATED WORK

Summarization techniques are mainly classified into two categories: extractive [5] approach and abstractive approach [6], [7]. In recent years, summarization researches began to expand to software artifacts such as user stories [8], source code [9] and bug reports. For bug reports, sentence-level extractive model is the main summarization technique, which extracts the central sentences from the original text in accordance with a certain compression ratio.

Rastka et al. [10] first proposed a model to extract summaries automatically from bug reports and created a bug report corpus called BRC corpus. They extracted more than 20 features from each sentence and trained Logistic Regression models to select sentences from a bug report. He Jiang et al. [11] found the relationship between the writing style consistency and the quality of the written report by mining authorship characteristics in bug repositories. And they proposed a new summarization method named Authorship Characteristics based Summarization (ACS). In addition, they also proposed a PageRank-based Summarization Technique (PRST) [12], which utilizes the textual information contained in bug reports and additional information in associated duplicate bug reports. Lotufo et al. [13] proposed an unsupervised bug report summarization approach that estimates the attention a user would hypothetically give to different sentences in a bug report when pressed with time. In addition, Yeasmin et al. [14], [15] applied the Lotufo's model to practice and constructed a visualized summarization model.

Previous researches have yielded good results by mining different features of bug reports. Moreover, intentions of sentences have been used in many other software artifacts except for bug reports, providing us with a new perspective to explore the characteristics of bug reports. Sorbo et al. [4] proposed a novel, semi-supervised approach named DECA (Development Emails Content Analyzer) that used Natural Language Parsing to classify the sentences in development emails according to their purpose. They said that their work can be used in summarizing work and they have already generated app review summary based on intention classification [3].

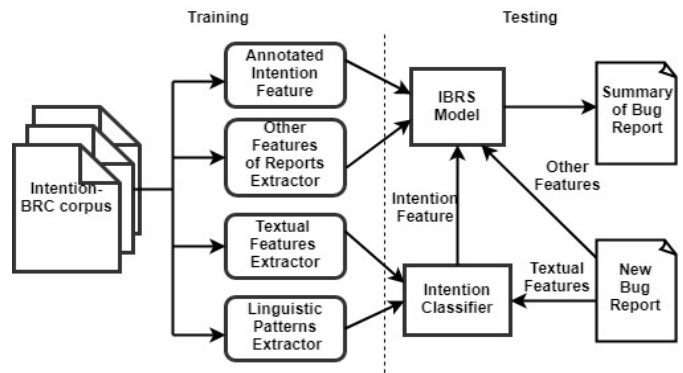


Fig. 2. IBRS Approach Overview

## III. APPROACH

This section introduces the framework of IBRS as shown in Figure 2. It consists of features extractor, intention classifier and Intention-based Bug Report Summarization Model. Our goals are to build a taxonomy for intentions in bug reports, construct an automatic intention classifier to get sentences intentions and take advantage of the sentences intentions to improve the original bug report summarization approach.

### A. Intention Taxonomy

In the work of Sorbo et al. [4], they divided the intention of development email sentences to 6 categories, including feature request, opinion asking, problem discovery, solution proposal, information seeking and information giving. Through observation, we find that many of their intention categories are similar to intentions in bug reports. For example, problem discovery in development emails is closely related to bug description in bug reports. Solution proposal in development emails is closely related to fix method in bug reports. However, bug reporters mainly focus on the discussion of bugs. Through reading bug reports and referring to the intentions of development emails, we define a total of 7 different intentions for bug reports as shown below.

- **Bug Description.** To describe a bug (i.e. What is the problem and how does it occur). E.g. *“The problem here is that nsSearch-Suggestions.js is passing the wrong previous Result to form history.”*
- **Fix Solution.** To describe how to fix a bug. E.g. *“The simple fix it to just discard the service’s form history result copy when startSearch() is called with a null previous result.”*
- **Opinion Expressed.** To Express the developers’ ideas. E.g. *“Yes, agreed the pref is not ideal for this purpose.”*
- **Information Seeking.** Developers ask for information. E.g. *“Can I commit gtkshow.c to gtk+ when I remove gtk\_show\_help?”*
- **Information Giving.** Developers give their suggestions or other information. e.g. *“You could winding kde-base/apps/konsole back a few revisions to see if the problem disappears.”*



- **Meta/Code.** A sentence that mainly consists of code, stack information and other meta data. E.g. “*CreateFileOperation op1 = new CreateFileOperation(file,...);*”
- **Emotion Expressed.** Greeting words or feeling expressed of something. E.g. “*Good point.*” “*Hi Martin*”

### B. Intention-BRC Corpus

In order to mine the intentions in bug reports, we annotate the intention of each sentence in the BRC corpus. BRC corpus is created by Rastka et al [1]. There are 36 bug reports (2360 sentences) in it, which from 4 different open-source software projects: Eclipse Platform, Gnome, Mozilla and KDE. They assigned three annotators to each bug report to select sentences that should occur in the summary of this report. For each bug report, the set of sentences which be marked as summary by more than one annotators is called the gold standard summary (GSS). They also roughly classified each sentence and labelled the categories. However, the categories annotations in the corpus are not very accurate and some of the categories annotations are missing. We re-annotate each sentence in the corpus according to the intention taxonomy we defined. Moreover, we correct some incorrect ID of sentences in BRC corpus. After that, we call the modified corpus as Intention-BRC Corpus in this paper and make it public<sup>1</sup>.

### C. Intention Mining

The category information of intentions can be used as a distinctive feature in bug reports. To prove this, in this section, we analyse the Intention-BRC Corpus.

The annotation result shows that there are 374 sentences labelled as *Bug Description*, 164 sentences labelled as *Fix Solution*, 239 sentences labelled as *Opinion Expressed*, 85 sentences labelled as *Information Seeking*, 588 sentences labelled as *Information Giving*, 744 sentences labelled as *Meta/code*, 166 sentences labelled as *Emotion expressed*. We wonder whether different categories of intentions are well-differentiated from other features of a bug report. So we analyse the following data:

**The probability of a sentence that appears in the summary.** In fact, when people reading a bug report, they always pay different attention to sentences of different intentions. For example, “*Good point*” and “*The simple fix it to just discard the service’s form history result copy when startSearch() is called with a null previous result*” are sentences with different intentions. Obviously, the last sentence is more likely to appear in summary. We counted the number of sentences with each intention that appeared in summary in Intention-BRC Corpus as  $c_1$ , the number of sentences that not appeared in the summary as  $c_2$ . Figure 3 shows the result. The probabilities for sentences with *Bug Description* and *Fix solution* intentions to appear in summary are significantly higher than the others.

We define parameters  $P_i$  as the probability to be selected as summary for intention category  $i$ . In our definition,  $P_i$  is proportional to the ratio of the number of occurrences in the

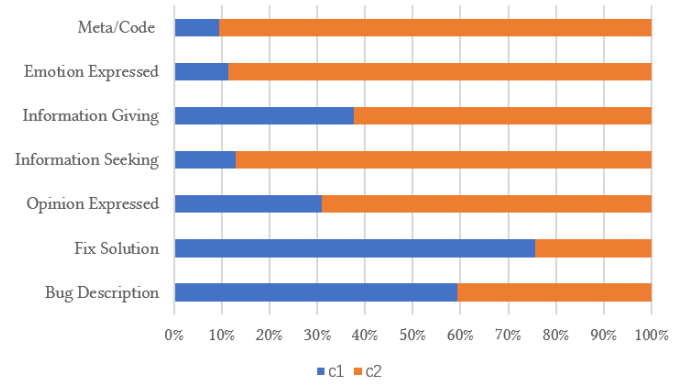


Fig. 3. Sentence Count for Each Intention

summary and the total number of sentences of intention  $i$ . Formally,

$$P_i = \frac{\sum_{s \in \text{summary}} \mathbb{I}(s \in \text{summary})}{\sum \mathbb{I}(s \in \text{sen}_i)} = \frac{c_1}{c_1 + c_2} \quad (1)$$

$\text{sen}_i$  represents the set of sentences with intention  $i$ .

**Sentences length of each intention.** In general, the length of sentences with different intentions are different due to the amount of information and information types involved are different. For example, the *Bug Description* is to describe a problem (e.g. how the problem occurs) and the sentences of this intention is usually very long. We calculate the average sentence length for each category, formally,

$$L_i = \frac{\sum \text{length}(s \in \text{sen}_i)}{\sum \mathbb{I}(s \in \text{sen}_i)} \quad (2)$$

**Sentences position of each intention.** We observe that there is also a big difference in the location distribution of sentences with different intentions. For example, problems are usually described first in a bug report, therefore *Bug Description* sentences usually appear at first comment. Similarly, we calculate the average location of sentences for each category, formally,

$$C_i = \frac{\sum \text{comment position in report}(s \in \text{sen}_i)}{\sum \mathbb{I}(s \in \text{sen}_i)} \quad (3)$$

$$S_i = \frac{\sum \text{position in comment}(s \in \text{sen}_i)}{\sum \mathbb{I}(s \in \text{sen}_i)} \quad (4)$$

Table I shows the calculation result of PLCS for each intentions. Through the above data analysis, we can see that the categories of intentions can show differences in multiple feature dimensions. It shows that the intention implicitly expresses several features of bug reports to some extent.

### D. Intention Classification

In order to automatically obtain the intention of sentences in bug reports, we train an intention classifier. Sorbo et al. [4] proposed an approach called DECA to classify the

<sup>1</sup><https://github.com/HuaiBeibei/IBRS-Corpus>

TABLE I  
PLCS FOR EACH INTENTION

Intention category	P	L	C	S
Bug Description	0.594	16.660	3.786	5.805
Fix Solution	0.756	17.226	8.726	4.006
Opinion Expressed	0.310	16.272	8.732	3.967
Information Seeking	0.129	12.388	8.612	3.967
Information Giving	0.378	17.252	7.932	6.129
Meta/Code	0.114	9.212	6.056	23.659
Emotion expressed	0.095	4.723	6.837	5.753

sentences in development emails. They created 231<sup>2</sup> heuristics to detect common linguistic patterns in sentence to predict their intention. However, finding linguistic patterns is a tedious work and it is impossible to define all patterns. Therefore, we create a mixed model which consists of linguistic patterns matching and machine learning classifying. For each sentence to classify, we first try to find the linguistic patterns matching it to get the intention category. If no patterns matched we input it to the trained machine learning classifier to get the intention category.

**Machine learning classifier.** First, we pre-process the textual content by applying stop-word removing, stemming and lowering. Then we use bag-of-words model to construct a Term-by-Documents Matrix  $M$  where  $M_{i,j}$  represents the weight of the  $i$ -th term contained in the  $j$ -th sentence. We weight word using the  $tf$  (term frequency), which weight each word  $i$  in document  $j$  as:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \quad (5)$$

where  $n_{i,j}$  is the frequency of word  $i$  appear in document  $j$ . We use  $tf$  instead of  $tf-idf$  indexing for the use of the inverse document frequency ( $idf$ ) penalizes too much on terms appearing many times in documents. In our work, we need these frequent words (such as “bug”, “think”) guide our machine learning classifier.

**Linguistic patterns.** We define several regular expressions and reuse part of the 231 heuristics (Some of the heuristics can also apply to our intention taxonomy, e.g. problem discovery could be regard as bug description, solution proposal could be regard as fix solution). Following are some examples to identify intentions using linguistic patterns. Each example consists of a linguistic pattern and a matching sentence from bug reports.

- Example 1: Identify Code Category.
  - `\w+\s\w+\s\(.+\)`
  - “`int ftp_connection (FtpConnection *conn, const Ftp550Handler *handlers, const FtpFile *file)`”.
- Example 2: Identify Information Seeking Category.
  - `[how|what|left|Do|Does|Is].+\?`
  - “*Does this happen every time?*”
- Example 3: Identify Information Giving Category.

<sup>2</sup>[http://www.ifi.uzh.ch/seal/people/panichella/DECA\\_Implemented\\_Heuristics.pdf](http://www.ifi.uzh.ch/seal/people/panichella/DECA_Implemented_Heuristics.pdf)

- `[someone] could [verb]`
- “**You could winding** `kdebase/apps/konsole` back a few revisions to see if the problem disappears”

### E. Improved Automatic Summarization of Bug Reports

Different from the previous methods, we combine the intention feature to the summarization model. The score of sentence  $s$  can be expressed by the following formula:

$$F_s = (1 - \alpha) * BRS_s + \alpha * P_{intention(s)} \quad (6)$$

$BRS_s$  is the score output by the original BRC model.  $intention(s)$  is the intention category of sentence  $s$  and  $P_{intention(s)}$  is the probability weight for this intention category. In fact, we can add the intention feature to machine learning model to learn the  $\alpha$ . Following are the features we used to train IBRS model:

- 1) *structural features* are related to the structure of the bug report (e.g., the position of the sentences).
- 2) *participant features* are directly related to the conversation participants (e.g., whether the sentence is made by the same person who filed the bug report).
- 3) *length features* related to the length of the sentence
- 4) *lexical features* are related to the occurrence of unique words in the sentence that we could use to calculate the sentence similarity with bug report title.
- 5) *Intention features* are related to the weight of intention category of the sentences.

## IV. EXPERIMENTS

### A. Research Questions

In this paper, we are interested in the following research questions and conduct two experiments to evaluate our approach:

**RQ1: How does our intention classifier perform?** For this question, we construct the intention classifier and evaluate it on Intention-BRC corpus.

**RQ2: Does the sentence intention feature improve the result of bug report summary model?** To answer this question, we add the intention feature to the original summary model to construct IBRS and evaluate it in experiment II.

**RQ3: What is the impact of missclassification to IBRS?** To answer this question, we use labelled intentions in corpus rather than the predicted intentions of the intention classifier to construct IBRS and evaluate it in experiment II.

### B. Experiment I on Intention classifier

Algorithm 1 show the main algorithm of our intention classification approach. We choose Random Forest (RF) [16] classifier and implement leave-on-out method on RF to make sure the sentence to classify not appear in the train set.

To evaluate the result of the intention classification, For each category, we calculate the precision, recall and F-score. The result of the classifier is shown in Table II.

Answer for RQ1: Through experimental verification, we can correctly identify the intention of 59% of the sentences from

**Algorithm 1** Intention Classification

---

Split the BRC corpus by sentences  
**for** each sentence  $s \in \text{BRC corpus}$  **do**  
  **for** each linguistic patterns defined **do**  
    **if** the sentence  $i$  match the pattern **then**  
      **return** the intention the pattern represent  
    **end if**  
  **end for**  
**return** the sentence predicted by the trained RF classification  
**end for**

---

TABLE II  
EVALUATION MEASURES OF EXPERIMENT I

CLASS	TP	FP	FN	Precision	Recall	F-score
Bug Description	182	186	192	0.49	0.47	0.48
Fix Solution	19	63	145	0.23	0.12	0.16
Opinion Expressed	27	39	212	0.41	0.11	0.17
Information Seeking	60	36	25	0.63	0.71	0.67
Information Giving	364	375	224	0.49	0.62	0.55
Meta/Code	679	251	65	0.73	0.92	0.81
Emotion Expressed	52	27	114	0.66	0.31	0.42
Overall Performance	1383	977	977	0.59	0.59	0.59

bug reports. Especially, our intention classification approach performs better at several categories, for example, *Meta/Code*, *Information Seeking*, *Information Giving*. This is because the sentences with these intentions have more obvious structural features. For example, code sentences' keywords have a strong distinction, such as "public", "static", etc. *Information Seeking* sentences are often in the form of questions and usually contain obvious keywords such as "what", "how", "why", etc. The worse performing results are from *Fix Solution* and *Opinion Expressed*. There are two main reasons. One reason is that the count of these two category samples is much smaller than the others (7% *Fix solution* sentences and 10% *Opinion Expressed* sentences). The other reason is that the diversity of sentence structure of these two categories. In addition, since bug report repository is a relatively open platform, developers do not write a standardized language when reporting (for example, they usually use abbreviated form).

### C. Experiment II on IBRS

This experiment combines the intention classifier and BRC model. Similar to the method in experiment I, we also use a leave-one-out procedure (i.e. leave one bug report out). The process of the IBRS experiment is as following:

- 1) Leave one bug report out of the training set. For every sentence in bug report corpus, we get their intention category using the intention classifier trained without sentences in this bug report.
- 2) We use the remained set in intention-BRC corpus to train the IBRS model. We map the intention label to get the sentence intention feature(P) from Table I and add the intention features so that our new summary model can take the category of intention into account.

TABLE III  
EVALUATION MEASURES OF EXPERIMENT II

Approach	Precision	Recall	F-score	Pyramid Precision
BRC	0.57	0.35	0.43	0.66
BRC*	0.54	0.34	0.42	0.64
IBRS	0.59	0.37	0.45	0.69
IBRS*	0.65	0.41	0.50	0.72

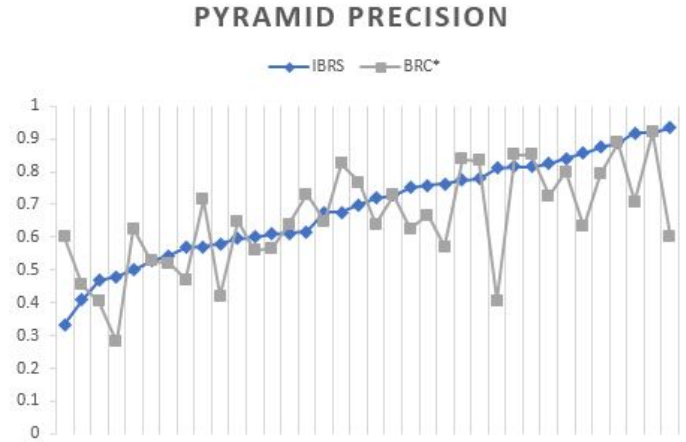


Fig. 4. Pyramid Precision For IBRS and BRC\*

- 3) For the left bug report, we map the sentences to get the sentence intention feature. Then use the IBRS to get the sentence score. We sort the sentences by their scores and select top 20% (for that 20% is approximately equal to the proportion of GSS in the corpus) as summary.

To evaluate the result, we calculate the precision, recall, F-score and pyramid precision [1] for the IBRS model. In order to compare with original models, we also reproduce the BRC experiment proposed by Rastkar et al. To answer RQ3, we also change the experiment to use the labelled intentions rather than the predicted intentions at step 1). The result is shown in Table III.

In our reproduced experiment, we get precision of 54% and recall of 34% (We correct several errors in the corpus of BRC annotations that may have caused a slight difference between the reproduced results and the experimental results of Rastkar et al.) as showed in second row in Table III marked as BRC\*. The original result of BRC approach is shown in first row. The result of IBRS is shown is third row. IBRS\* shown in last row is the result of IBRS model trained using correctly labelled intentions.

We also calculate pyramid precision for every single bug report. Figure 4 shows the values of pyramid precision for the BRC\* and IBRS approach. The bug reports have been sorted based on the pyramid precision of IBRS. The figure shows that 22 of 36 reports (61% of total reports) get better summary using IBRS.

Answer for RQ2: Our IBRS out-performs the BRC\* approach on precision (5% improved), recall (3% improved), F-score (3% improved) and pyramid precision (5% improved).

61% of bug reports get summaries with better pyramid precision using IBRS. The only different between the BRC\* and IBRS is that IBRS takes the sentence intention category feature into account. It proves that the intention feature indeed improves the work of automatic summarization for bug reports.

Answer for RQ3: From the result of IBRS\*, it shows that IBRS\* outperforms the IBRS 6% on precision. IBRS\* is trained with labelled intention while IBRS trained with the intention classifier of 59% precision. Although IBRS improves the summarization for bug reports, the misclassification limits the promotion of summarization to some extent. It proves that intentions can enhance the summarization work better. Improving the accuracy of the intention classifier is one of our future work.

## V. DISCUSSION

### A. Threats to Validity

In this paper, we use Intention-BRC corpus that consists of 36 bug reports. This may bring threats to the validity of intention taxonomy for the corpus are not larger enough. However, the 36 bug reports are from 4 different popular project and can represent the characteristics of most of bug reports. The size of the corpus also bring threats to the evaluation of the experiment. We apply the leave-one-out method to ensure that the assessed samples do not participate in training and make maximum use of corpus. In addition, we select 20% sentences as summary of a bug report may bring threats to the quality of the summary content. The state-of-the-art researches usually count the average proportion of sentences in summary to total sentences as an indicator of the number of sentences to extracted. 20% is the proportion of GSS to total sentences in corpus, so it is a relatively reasonable extraction ratio.

### B. Future Work

There is a lot of room to improve on intention classifier. In our future work, we are interested in improving the accuracy of the intention classifier by adding more heuristics and trying new ways to characterize bug report text (e.g. DBRNN-A [17] instead of our bag-of-words (BOW)).

Moreover, the intentions of sentences can be used in many other ways. For example, we can rearrange or even reconstruct the summary text of the bug report based on the intention of the sentence. There are many redundant sentences with similar meaning in the summary text, we can further abstract the sentences with same intention into a abstract text. [18].

## VI. CONCLUSION

In this paper, We achieve significant improvement to the automatic summarization for bug report by introducing intention feature to original approach. We assume that bug report text usually contains different intentions. We introduce the intention taxonomy and propose **IBRS** (Intention-based Bug Report Summarization). To evaluate the performance of the intention classifier and IBRS model, we design two experiments. The result shows the precision rate of intention classifier is 59%.

Based on the intention classifier, we implement IBRS model, which outperforms the original BRC model with precision of 59% (5% improved), recall of 37% (3% improved) , f-score of 45% (3% improved) and pyramid precision of 69% (5% improved). The results of experiments show that mining intentions indeed improves the bug reports summarization.

## REFERENCES

- [1] S. Rastkar, G. C. Murphy, and G. Murray, "Automatic summarization of bug reports," *IEEE Transactions on Software Engineering*, vol. 40, no. 4, pp. 366–380, 2014.
- [2] S. Mani, R. Catherine, V. S. Sinha, and A. Dubey, "Ausum: approach for unsupervised bug report summarization," in *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*. ACM, 2012, p. 11.
- [3] A. Di Sorbo, S. Panichella, C. V. Alexandru, J. Shimagaki, C. A. Visaggio, G. Canfora, and H. C. Gall, "What would users change in my app? summarizing app reviews for recommending software changes," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 2016, pp. 499–510.
- [4] A. Di Sorbo, S. Panichella, C. A. Visaggio, M. Di Penta, G. Canfora, and H. C. Gall, "Development emails content analyzer: Intention mining in developer discussions (t)," in *Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on*. IEEE, 2015, pp. 12–23.
- [5] G. Erkan and D. R. Radev, "Lexrank: Graph-based lexical centrality as salience in text summarization," *Journal of Artificial Intelligence Research*, vol. 22, pp. 457–479, 2004.
- [6] R. Nallapati, B. Zhou, C. Gulcehre, B. Xiang *et al.*, "Abstractive text summarization using sequence-to-sequence rnns and beyond," *arXiv preprint arXiv:1602.06023*, 2016.
- [7] A. M. Rush, S. Chopra, and J. Weston, "A neural attention model for abstractive sentence summarization," *arXiv preprint arXiv:1509.00685*, 2015.
- [8] R. Krasniqi, S. Jiang, and C. Mcmillan, "Tracelab components for generating extractive summaries of user stories," in *IEEE International Conference on Software Maintenance and Evolution*, 2017, pp. 658–658.
- [9] P. W. Mcburney and C. Mcmillan, "Automatic source code summarization of context for java methods," *IEEE Transactions on Software Engineering*, vol. 42, no. 2, pp. 103–119, 2016.
- [10] S. Rastkar, G. C. Murphy, and G. Murray, "Summarizing software artifacts: a case study of bug reports," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering—Volume 1*. ACM, 2010, pp. 505–514.
- [11] H. Jiang, J. Zhang, H. Ma, N. Nazar, and Z. Ren, "Mining authorship characteristics in bug repositories," *Science China Information Sciences*, vol. 60, no. 1, p. 012107, 2017.
- [12] H. Jiang, N. Nazar, J. Zhang, T. Zhang, and Z. Ren, "Prst: A pagerank-based summarization technique for summarizing bug reports with duplicates," *International Journal of Software Engineering and Knowledge Engineering*, vol. 27, no. 06, pp. 869–896, 2017.
- [13] R. Lotufo, Z. Malik, and K. Czarnecki, "Modelling the 'hurried' bug report reading process to summarize bug reports," *Empirical Software Engineering*, vol. 20, no. 2, pp. 516–548, 2015.
- [14] A. Yeasmin, C. K. Roy, and K. A. Schneider, "Interactive visualization of bug reports using topic evolution and extractive summaries," in *Software Maintenance and Evolution (ICSME), 2014 IEEE International Conference on*. IEEE, 2014, pp. 421–425.
- [15] —, "How should we read and analyze bug reports: an interactive visualization using extractive summaries and topic evolution," in *Proceedings of the 25th Annual International Conference on Computer Science and Software Engineering*. IBM Corp., 2015, pp. 171–180.
- [16] A. Liaw, M. Wiener *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.
- [17] S. Mani, A. Sankaran, and R. Aralikatte, "Deeptriage: Exploring the effectiveness of deep learning for bug triaging," *arXiv preprint arXiv:1801.01275*, 2018.
- [18] G. Murray, G. Carenini, and R. Ng, "Generating and validating abstracts of meeting conversations: a user study," in *Proceedings of the 6th International Natural Language Generation Conference*. Association for Computational Linguistics, 2010, pp. 105–113.

# Evaluating the Effort of Integrating Feature Models: A Controlled Experiment

Vinicius Bischoff, Kleinner Farias, Lucian José Gonçalves

Post Graduate Program on Applied Computing (PPGCA)

University of Vale do Rio dos Sinos (Unisinos)

São Leopoldo, Brazil

viniciusbischof@edu.unisinos.br, kleinnerfarias@unisinos.br, lucianj@edu.unisinos.br

**Abstract**—The integration of feature models plays a key role in many tasks in software development, such as evolving Software Product Lines (SPL) to add new features. However, based on our experience in previous empirical studies, one of the main shortcomings to the widespread adoption of integration techniques is the lack of empirical knowledge about its effects on the effort of analysts and developers. This problem applies to integration techniques involving a set of operations (union and intersection) as well as the relationships between features and their elements. This article, therefore, reports on a controlled experiment that investigated the effort of (1) applying the integration techniques of feature models by professionals and students, and (2) detecting and resolving inconsistencies in the output-integrated models. The integration effort was evaluated through 10 evolution scenarios. The main results suggest that there is no significant difference regarding (1) the integration effort invested by professionals and students to produce a desired integrated model, and (2) the correctness rate of the integrations performed by professionals and students.

**Keywords** - Feature Model; Integration; Experimental Study.

## I. INTRODUCTION

Feature models (FM) can be seen as a “big-picture” of the functionalities of a software system. The integration of feature Models plays a pivotal role on software engineering tasks. For this, each developer performs tasks, such as changing or adding new features in a specific feature delta model,  $FM_B$ . These changes are often performed in parallel, and then each developer accommodates these changes into a base feature model,  $FM_A$ . Developers need to integrate these modifications to update the “big picture” of a software system. Specifically, integration of feature models might be briefly defined as a set of activities that should be performed over two input models,  $FM_A$  and  $FM_B$ , to produce a desired output-composed feature model,  $FM_{AB}$ .

However, developers may end up not producing the  $FM_{AB}$ . Instead, developers often produce an output-composed model,  $FM_{CM}$ , with problems (i.e.,  $FM_{CM} \neq FM_{AB}$ ) [1][11]. This happens because developers are usually unable to properly detect and resolving integration problems, such as conflicts and inconsistencies, given the problem at hand. Hence, to produce the  $FM_{AB}$  they must invest effort to resolve such conflicts and inconsistencies in the  $FM_{CM}$ . Conflicts are contradicting information found in  $FM_A$  and  $FM_B$ . In other words, conflicts are different values assigned to the properties of feature models

(e.g., name). For example, the variability property of a given feature *Researcher* defined as *mandatory* in  $FM_A$ , while in  $FM_B$  its variability property is defined as *optional*. These contradicting values assigned to these specific features represent a conflict that developers must resolve. However, if this issue is not properly resolved, inconsistencies are inserted into the output-composed feature model,  $FM_{CM}$ . For example, *Researcher* variability property defined as *optional* denotes an inconsistency as the expected value should be *mandatory*. Model inconsistencies [10] can be briefly defined as a mismatch between  $FM_{CM}$  and  $FM_{AB}$ .

Previous works already investigated the effects of composition tasks on developers’ effort, and their experiences [1][2]. In [1], the authors evaluated the effort invested to compose UML models using specification-based and heuristic-based techniques.; however, the integration of features models was not explored. In [2], the authors evaluated the impact of experience level on comprehension of C++ lambdas functions. Integration of feature models was not also the focus of the authors. To sum up, none of them investigated the effects of integration tasks of feature models on developer’s effort. Also, there is a lack of empirical evidence regarding the effort of software developers on integrating feature models.

To account for this, this work conducts a controlled experiment to analyze the effort that developers invest on activities related to the integration of feature models. In particular, we seek to explore the effort invested by two categories of participants, including students and professionals. This experiment was executed based on well-defined guidelines [6]. In [1], the authors argue that this kind of study is important because it provides scientific evidence about the developer’s performance on software engineering tasks. This prevents the development team’s decisions limited to only on opinion of experts and evangelists, thus providing strong empirical evidence.

## II. STUDY METHODOLOGY

### A. Objective and research questions

The objective of this work is to *analyze* the integration techniques of feature models *for the purpose of* investigating *with regard to* effort and correctness *from the perspective of* students and professionals *in the context of* evolution of feature models. This objective is based on the GQM template [5].

Based on this objective, two Research Questions (RQ) are formulated:

- **RQ1:** What is the effort required to integrate FMs?
- **RQ2:** What is the rate of correctly integrated FMs?

### B. Hypothesis formulation

This Section formulates the hypotheses that guide our experiment to answer the respective two formulated research questions. These hypotheses are described below:

**H1. Null Hypothesis 1, (H<sub>1-0</sub>):** Professionals apply less or equal effort to integrate FMs (IE) manually than students.

$$H_{1-0}: IE_{\text{professional}}(FM_A, FM_B) \leq IE_{\text{student}}(FM_A, FM_B)$$

**H2. Hypothesis Null 2, (H<sub>2-0</sub>):** The rate of correctly integrated elements (CIR) performed by professionals is equal or greater than one produced by students.

$$H_{2-0}: CIR_{\text{professional}}(FM_{CM}) \geq CIR_{\text{student}}(FM_{CM})$$

As in any experiment, the main objective is to reject these null hypotheses. In the context of this work, we conjecture that the professionals presented better results compared to the students.

### C. Study variables

The dependent variable of the first hypothesis (H1) is Integration Effort (IE). The IE represents the time (in minutes) spent to integrate two input-feature models, FM<sub>A</sub> and FM<sub>B</sub>, to produce FM<sub>CM</sub>. The dependent variable in the second hypothesis (H2) is the Rate of Correctly Integrated Features (CIR). CIR is a correctness rate. The CIR formula is the result of the number of participants who correctly answered the investigated question (NPAC), divided by the total number of participants (NPT), i.e.,  $CIR = NPAC/TNP$ .

The independent variable of this experiment is the experience level of participants, which can assume two values: Students and Professionals. Professionals are active persons on software industry, while students are persons that studies in universities. Therefore, students are organized in tree groups, i.e., technical, graduate and postgraduate.

### D. Context and participants

The context of this study is related to the evolution of feature models. This means that, users must properly integrate the changes on a delta model FM<sub>B</sub> into the base model FM<sub>A</sub>. Therefore, the participants must choose the right answer among five options. 10 Evolution Scenarios (ES) were developed to evaluate the integrations.

A total of 25 participants attended this experiment. The professionals group contains 07 persons. The student group contains 05 undergraduate students; 03 graduate students, and 10 students from IT courses.

### E. Experimental process

The experimental process consists of three steps: (1) Training; (2) Execution of feature integration activities; and (3)

Participant Background and Data Collection. In the first step (1) all participants were trained to ensure that they acquired the necessary familiarity with model integration techniques. In the next step (2), developers concerned on feature integration tasks, i.e., participants analyze the input models (FM<sub>A</sub> and FM<sub>B</sub>) of each scenario based on descriptions of changes. In this step, they also resolved conflicts. Participants should resolve conflicts according to the change requests listed in each question to produce a composed model, FM<sub>CM</sub>. Finally, they tried to produce the desired feature model. This activity consists of integrating the models, i.e., producing the FM<sub>AB</sub>. In the last step (3), the participants provided background information such as their professional experience, graduate level, level of experience on software modeling and development. Finally, all produced data related to the experiment were collected.

### F. Analysis procedures

**Quantitative analysis.** We performed descriptive statistics to analyze their normal distribution and statistical inference to test the hypotheses [6][7]. Our analysis was performed to test the hypotheses in both groups in all experimental tasks. We applied the *Student's t-test* to validate the hypotheses intrinsic to this research to check the normality of the variables, the *Kolmogorov-Smirnov - (Lilliefors)* test, which is a broad test of the distribution function of at the same time [6][7]. Although the data distribution is subdivided into treatment (groups), the validity hypothesis refers only to the group (professionals and students), however, an individual evaluation of the categories will be presented.

## III. STUDY RESULTS

This Section presents the results regarding the investigated research questions. Section III.A presents the results in relation to the RQ 1 that investigates the effort on integration techniques. Section III.B presents the results in relation to the RQ2 that investigates the influence of experience level on the rate of correctly integrated feature models. Finally, Section III.C presents some additional observations.

### A. Effort and integration techniques

**Descriptive statistics.** This section discusses the descriptive statistics regarding the impact of experience level (professional and students) on the effort on integrating feature models (IE). Table 1 shows the descriptive statistics of the collected data. Group 1 shows that university students (GR and PG) apply less effort to integrate feature models, i.e., on average the effort is 1.88 minutes. Specifically, they applied 25.24% less effort to integrate the FMs in relation to technical students. In Group 2, University Students (Graduates and postgraduates) also spent less effort compared to professionals to integrate feature models. The effort is 1.88 min., which represents 2.09% less than professionals to integrate the FMs. In Group 3, professionals spent less effort to integrate features than technical students. Specifically, industry professionals applied average of 1.92 min. integrating features, i.e., 23.64%, less effort to integrate the FMs in relation to technical students. Finally, in Group 4, that is the general comparison between

Table 1. Descriptive statistics and hypothesis tests.

Group 1 Technical Students vs University Students	Variables	Treatment 18 participants	SD	Min	25 <sup>th</sup>	MD	75 <sup>th</sup>	Max	Avg.	% Diff	t-test
											p-value
	CIR	TECH	0.18	0.30	0.40	0.60	0.70	0.80	0.55	6.78	0.0716
		GR and PG	0.29	0.13	0.25	0.69	0.63	1	0.59		
	IE	TECH	0.58	1.61	2.33	2.65	2.67	3.57	2.51	25.24	0.015
		GR and PG	0.48	1	1.40	1.94	1.67	2.50	1.88		
Group 2 Professional vs University Students	Variables	Treatment 15 participants	SD	Min	25 <sup>th</sup>	MD	75 <sup>th</sup>	Max	Avg.	% Diff	t-test
											p-value
	CIR	PRO	0.23	0.14	0.20	0.43	0.43	0.86	0.44	26.17	0.197
		GR and PG	0.29	0.13	0.25	0.69	0.63	1	0.59		
	IE	PRO	0.71	1	1.33	1.71	1.67	3.40	1.92	2.09	0.884
		GR and PG	0.48	1	1.40	1.94	1.67	2.50	1.88		
Group 3 Professional vs Technical Students	Variables	Treatment 17 participants	SD	Min	25 <sup>th</sup>	MD	75 <sup>th</sup>	Max	Avg.	% Diff	t-test
											p-value
	CIR	PRO	0.23	0.14	0.20	0.43	0.43	0.86	0.44	20.91	0.276
		TECH	0.18	0.30	0.40	0.60	0.70	0.80	0.55		
	IE	PRO	0.71	1	1.33	1.71	1.67	3.40	1.92	23.64	0.054
		TECH	0.58	1.61	2.33	2.65	2.67	3.57	2.51		
Group 4 (General) Professional vs Students	Variables	Treatment 25 participants	SD	Min	25 <sup>th</sup>	MD	75 <sup>th</sup>	Max	Avg.	% Diff	t-test
											p-value
	CIR	PRO	0.23	0.14	0.20	0.43	0.43	0.86	0.44	22.81	0.146
		TECH, GR and PG	0.24	0.43	0.33	1.29	0.67	1.80	0.57		
	IE	PRO	0.71	1	1.33	1.71	1.67	3.40	1.92	12.39	0.274
		TECH, GR and PG	0.53	1.31	1.86	2.30	2.17	3.04	2.19		

**Legend:** Standard Deviation (SD), Minimum (Min), First Quartile (25<sup>th</sup>), Median (MD), Third Quartile (75<sup>th</sup>), Maximum (Max), Average (Avg.), Percentage Difference (% Diff), Correct integration rate (CIR), Integration effort (IE), Technician (TECH), Graduate (GR), Postgraduate (PG) and Professional (PRO).

professionals and students, shows that professionals spent on average 1.92 min. to integrate feature models, i.e., 12.39% less effort to integrate feature models in relation to students. Therefore, professionals tend to invest less effort to produce  $FM_{AB}$  using manual integration techniques.

**Testing hypotheses.** We also performed statistical tests to evaluate whether the measures of  $Effort(FM_A, FM_B)$ ,  $eff(FM_A, FM_B)$ ,  $diff(FM_{CM}, FM_{AB})$ , and  $iff(FM_{CM})$  are statistically significant. We hypothesize that professionals in relation to students tend to require less effort than their counterparts. According to the hypothesis test previously described  $H_{1-0}$ , the t-test failed to reject the null hypothesis, with the p-value is 0.95. Therefore, there is no significant difference between the applied effort to integrate features between professionals and students.

### B. Correctness and integration techniques

**Descriptive statistics.** This section analyzes the collected data regarding the impact of integration techniques on the correctness rate (CIR). For this, we also calculated descriptive statistics to understand the distribution of the data, see Table 2. It was possible to verify in Group 1 that the CIR (rate of correctness) is higher for GR and PR (undergraduate and graduate students), which reached an average of 0.59 of correct answers, i.e., 6.78% more successful than technical students when integrating the FMs. In Group 2, professionals obtained a higher rate of correct answers in relation to the graduated and postgraduates students, i.e., an average of 0.59 correct answers, representing that university students were 26.17% less precise to integrate FMs. In Group 3, the correctness rate (CIR) is superior for the technical students, who reached an average of 0.55

correct answers, i.e., professionals were 20.91% less precise when integrating the FMs. Finally, in Group 4 we can verify that the rate of correct answers is superior for the students, who reached an average of 0.57 correct answers, i.e., professionals were 22.81% less precise to integrating feature models.

**Testing hypotheses.** It evaluates the experience level in relation to the CIR (rate of correctly integrated models). The rows identified with CIR shows the statistic p-values for comparisons between groups. The results show that t-test (T) rejects the null hypothesis  $H_{2-0}$ , with the p-value equal to 0.146. This means the level of experience does not have correlation with the rate of correctly integrated features. Specifically, the hypothesis test failed to reject the null hypothesis.

### C. Additional observations

All participants in this research have previously submitted to a small training with 15 minutes to explain what a feature is, how it behaves, what existing relationships, and ultimately examples of integration.

Academics tend to be more prepared than professionals with respect to the application of modeling techniques. As understood in the company surveyed professionals perform short meetings with the tasks to be developed. They do not follow models, only requirements described in their documentation. This way, we can extend the interpretations if they are going to undergo changes, since the documentation is not usually updated.

Considering our results, the average effort applied is two minutes per question, which implies in 20 minutes running the 10 questions. However, the degree of difficulty proposed for this research for integration between the FMs (syntactic and

semantic) is small, considering what is applied in the industry. This demonstrates the need for automation of integration techniques, as well as the possibility of working collaboratively between analysts and developers. To facilitate its visualization and the possible set of updates that is necessary. Another revealed fact refers to the corrected rate, which we believe can be improved with the application of a semiautomatic technique. In this way, indicating when any inconsistency occurs, so that the developer can make the decision that suits him best applied the Wilcoxon test and the t-test to check the H2-0.

#### IV. RELATED WORKS

The integration of feature models is a research field of interest in academia. The integration of features is important for composing software product lines [3]. Recently, the research initiatives focused on proposing techniques for features integration. However, there is a lack of experimental studies. In [3], several composition operators can be defined, depending on the combination strategies and semantic framework for developers and researchers to plan and carry out qualitative and quantitative research, as well as to reproduce and reproduce empirical studies. In [9], authors demonstrate how FMs can be reduced to propositional formulas or constraint satisfaction problems. Benefits are tools that propagate constraints (so that incorrect specifications can be detected automatically). Finally, this expected in Feature Models.

#### V. THREATS TO VALIDITY

**Statistical validity.** The independent and dependent variables were submitted to suitable statistical methods We minimized this threat by checking whether. We test all hypotheses considering the significance level at 0.05 level ( $p \leq 0.05$ ). **Construct validity.** The measures applied in this study, i.e., the effort and the correctness are widely applied on controlled experiments on software engineering [1][8]. **Internal validity.** The dependent variables varied appropriately according to corresponding independent variables. **External validity.** Some aspects must be followed to reproduce the results of this study such as: participants must have the familiarity with feature integration models must have similar sizes, and the same variables must be collected.

#### VI. CONCLUSIONS AND FUTURE WORKS

This article reported on a controlled experiment that explored three points: application effort of integration techniques of feature models by professionals and students, and detection and resolution effort of inconsistencies found in output-integrated models.

Both hypothesis tests failed rejecting the null hypothesis. This means that has no significant difference on performance on both groups on integration of Features Models. However, overall results on descriptive statistics show that professionals

tend to invest less effort to integrate feature models, but they produce integration with more errors than students. There are few studies that evaluate the effort required to use model integration techniques. Further empirical studies are still required to better understand whether these findings are confirmed or not in other contexts, considering other FMs, encompassing different evolution scenarios, and evaluating other integration techniques. Finally, we hope that the issues outlined throughout the paper encourage other researchers to replicate our study in the future under different circumstances and that this work represents a first step in a more ambitious agenda on better supporting feature model integration tasks.

#### ACKNOWLEDGMENT

Thank you to UNISINOS for the teaching and research environment in which they provided to support this research.

#### REFERENCES

- [1] K. Farias, A. Garcia, J. Whittle, C. v. F. G. Chavez and C. Lucena, "Evaluating the effort of composing design models: a controlled experiment," *Software & Systems Modeling*, vol. 14, pp. 1349-1365, 2015.
- [2] P. M. Uesbeck, A. Stefik, S. Hanenberg, J. Pedersen, and P. Daleiden. "An empirical study on the impact of C++ lambdas and programmer experience". In *Proceedings of the 38th International Conference on Software Engineering (ICSE '16)*. ACM, New York, NY, USA, pp. 760-771, 2016.
- [3] M. Acher, P. Collet, P. Lahire and R. France, "Composing feature models," em *International Conference on Software Language Engineering*, 2009.
- [4] M. M. Alam, A. I. Khan and A. Zafar, "A Comprehensive Study of Software Product Line Frameworks," *International Journal of Computer Applications*, vol. 151, 2016.
- [5] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical software engineering*, vol. 14, p. 131, 2009.
- [6] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. "Experimentation in software engineering". Springer Science & Business Media. 2012.
- [7] B. W. Yap and C. H. Sim, "Comparisons of various types of normality tests," *Journal of Statistical Computation and Simulation*, vol. 81, pp. 2141-2155, 2011.
- [8] A. Oliveira, V. Bischoff, L. J. Gonçalves, K. Farias and M. Segalotto, "BRCode: An interpretive model-driven engineering approach for enterprise applications," *Computers in Industry*, vol. 96, pp. 86-97, 2018
- [9] D. Batory, D. Benavides e A. Ruiz-Cortes, "Automated analysis of feature models: challenges ahead," *Communications of the ACM*, vol. 49, pp. 45-47, 2006.
- [10] K. Farias, A. Garcia, C. Lucena, "Evaluating the Impact of Aspects on Inconsistency Detection Effort: A Controlled Experiment," In: 5th Int. Conf. on Model-Driven Eng. Languages and Systems, Vol. 7590, pages 219-234, 2012.
- [11] K. Farias, L. Gonçalves, M. Scholl, T.C. Oliveira, M. Veronez.. "Toward an Architecture for Model Composition Techniques," In *27th Int. Conf. on Software Engineering and Knowledge Engineering (SEKE'15)*, pp. 656-659, 2015.



# Modeling of software process families with automated generation of variants

Andrea Delgado<sup>1</sup> and Daniel Calegari<sup>2</sup>  
Instituto de Computación  
Universidad de la República  
11300 Montevideo, Uruguay  
<sup>1</sup>adelgado@fing.edu.uy, <sup>2</sup>dcalegar@fing.edu.uy

Félix García  
Escuela Superior de Informática  
Universidad de Castilla-La Mancha  
13071 Ciudad Real, España  
felix.garcia@uclm.es

**Abstract**—Modeling variability in software systems or processes promotes reuse of core assets. In particular, variability in software development processes allows customization of activities, artifacts, roles and other elements to specific projects, in what is called a process variant. Each process variant is derived from a base common process, called process tailoring, which is usually a tedious and error prone manual process. In the last years, there has been a growing interest in supporting the process variability approach, being v-SPEM a proposal that integrates and advanced view of variability into the SPEM standard. In this paper we present an extension of v-SPEM support for final users, with automated generation of variants, to help during process tailoring. The use of automatic mechanisms reduces errors and simplifies the tailoring process by hiding the details to the end users.

**Keywords:** software development processes, SPEM, process variability, Software Process Lines, Model Driven Engineering.

## I. INTRODUCTION

A Software Process is a specific type of business process that focuses on the software development domain, as other domains have their own typical processes such as banking, health, or education. The Software Process Engineering Meta-modeling language (SPEM 2.0 [1]), is commonly used as a domain specific modeling language in this context.

Software processes may accept variants based on specific requirements of an organization or project, leading to the definition of a process family. A Software Process Line (SPrL) [2], [3] aims to provide the techniques and mechanisms to model the common (known as *base process*) and variable parts (known as *variation points*) of a process family, as well as to support the customization of each process variant (known as *process tailoring*). The advantages of using the SPrL paradigm to model software process families are well-known, but although there are many proposals and some tools to support them [4], the tailoring process is in general carried out by final users with the aid of technical users, following a not well defined, manual, error prone and complex process. What is more, existing tools such as Eclipse Process Framework (EPF) Composer<sup>1</sup> which implements SPEM, are mostly targeted to process modelers with process families modeling skills, not to final users.

With the aim of improving the automation of SPrL by reducing technical complexity to final users, the Model-Driven Engineering (MDE) [5] paradigm can be considered as a very promising approach. Based on this, we identify the following research question for our work: *How the automated tailoring of specific variants from a process family can be supported using the MDE approach?*

In this paper, we improve the v-SPEM [3] approach, a proposal of our own that integrates an advanced view of variability into the SPEM standard. The base approach does not formalize the variability process, its tool support is not user-friendly (vEPF<sup>2</sup>) for non-technical users, and the tailored process is hard-coded. In this context, we extend v-SPEM by defining a process composed by roles, activities and tool support for MDE-based process tailoring. Tailoring is performed by means of an ATL [6] model transformation, based on user's selection. The main contributions of our work are: i) a web tool targeted for final users in which a process family model can be imported and tailored to generate variants; ii) guidance on how to perform such tailoring; and iii) a set of ATL transformations which automatically generate the selected variant.

The rest of this paper is structured as follows. In Section II we briefly present the SPEM standard and v-SPEM extension. In Section III we present related work regarding existing process variability proposals. In Section IV we describe our proposal to support process tailoring within v-SPEM, including transformations for the automated generation of variants, and in Section V we provide an example of application. Finally in Section VI we present conclusions and future work.

## II. SPEM AND v-SPEM

The SPEM [1] standard defines specific elements for software process modeling, such as: tasks, activities, roles, work product, and processes. It also provides a way for modeling variability, defining several variability types between two related elements, such as: "replaces" which states that the origin element substitutes the target element under some assumptions, and "extend" which states that the origin element expands the target element definition, probably by overriding attributes and associations.

DOI reference number: 10.18293/SEKE2018-019

<sup>1</sup>EPF Composer, <https://www.eclipse.org/epf/>

<sup>2</sup>v-EPF, [https://alarcos.esi.uclm.es/vepf/variant\\_rich\\_process\\_paradigm/vepf.html](https://alarcos.esi.uclm.es/vepf/variant_rich_process_paradigm/vepf.html)













	Activity	WorkProductUse	RoleUse	TaskUse
Base Element				
VarPoint				
Variant				

Fig. 1. Main v-SPEM concepts [3]

However, it has some limitations from the perspective of a SP<sub>r</sub>L, e.g.: it is not clear how common and variable parts interact, and it is difficult to visualize the base process with the common elements [7]. In this context, the variability extension for SPEM (v-SPEM) [3], [7] provides means for a direct specification of the variability in a process. Variability is modeled in two ways: single (or on-point) variations for expressing variability with respect to specific elements of the process model (e.g.: tasks), and crosscutting variations for expressing variability with respect to several elements at once. Both kinds allow full modeling of software process families. In this work we focus on single variations.

A process engineer defines the variation points over the common process, specifying which variants can occupy these points (i.e.: a "replaces" relation), and during process tailoring, each variation point is substituted with exactly one variant. v-SPEM extends the SPEM metamodel and defines a new notation to represent variants and variation points for SPEM elements, as shown in Figure 1, in which a base SPEM element, e.g.: *Activity*, can be defined as a variation point (*VPActivity*) or as a variant (*VActivity*).

It also provides tool support as an extension of the EPF Composer, as mentioned, the v-EPF The tool allows modeling a process family and the Java-based generation of process variants. However, it is devised exclusively for technical users, and hard-codes the generation of variants.

### III. RELATED WORK

Some proposals are also based on SPEM 2.0 models. SmartySPEM [8] adds variability over SPEM by adding stereotypes to the metamodel. It provides guidelines for identifying and representing process variability, and process tailoring by variability resolution. Tool support is using any UML editor, which are as EPF, targeted to technical users. It was defined after v-SPEM and provides similar elements, and no generation of variants. In CASPER [9] the authors define two input models: the base process with variability, and a context model for a specific project, specifying things such as: project size, schedule, team size, among others. The approach provides automated support for process tailoring based on a ATL [6] model transformation, which takes the context model for the inference of variants. Then, variants are deduced from the values entered by the final user, leading to a variant they

generate based on previous knowledge. In this approach, the final user is not able to choose between variants for the variation points.

Other works are based on feature models (FMs), which is the common approach in software product lines. In [10] the authors use FMs for expressing variability with respect to roles, tasks and work products. One drawback is that features and relations of different types cannot be distinguished. Also, the control flow of the process is lost. In [11] the authors use a SPEM model for identifying variation points, and an Orthogonal Variability Model associates each variation point with the defined variants (as with FMs).

The Common Variability Language (CVL)<sup>3</sup> [12] is an approach for language independent variability modeling. Besides that automated generation of variants is supported (but not provided), models are hard to specify and maintain.

As a summary, all approaches provide support for variability almost over the same type of elements. Most of them also provide supporting tools, but not all are available. Also, all approaches provide guides for process tailoring. Besides CASPER, the generation of process variants in the other approaches is carried out manually. Unlike CASPER, we let the user choose each specific variant for each variation point.

### IV. v-SPEM TAILORING SUPPORT

In what follows we describe the roles, activities, process tailoring and tool support we propose for v-SPEM.

#### A. Roles

When managing process families, there are different stakeholders involved, who participate in different activities and are interested in different artifacts or work products, as in the software development process itself. We have defined two main roles within our proposal, the process engineer (technical user) and the final user (non-technical user).

The Process engineer role, is in charge of modeling the process family, thus it needs advanced software engineering skills such as knowing about SPEM, v-SPEM, SP<sub>r</sub>L and process families, being familiar with process modeling and using tools such as v-EPF for doing it.

The Final user can be a software project manager who has knowledge of software engineering and software development processes but, not necessarily about SP<sub>r</sub>L and process families. However, this user is the one who needs to tailor base processes into a specific variant for each project he will lead. Thus, for this kind of user, it is very valuable to have extra support to carry out the tailoring process.

Based on these definitions we defined the main activities each one must perform within the tailoring process with the Use Cases shown in Figure 2. As it can be seen, the process engineer is mostly involved with the technical activities to define the base process, variability points and possible variants to occupy them. The final user is the focus of the tailoring process, for which we extended the v-SPEM existing support. Also, we add the possibility of working with a central

<sup>3</sup>CVL. <http://www.omgwiki.org/variability/doku.php>

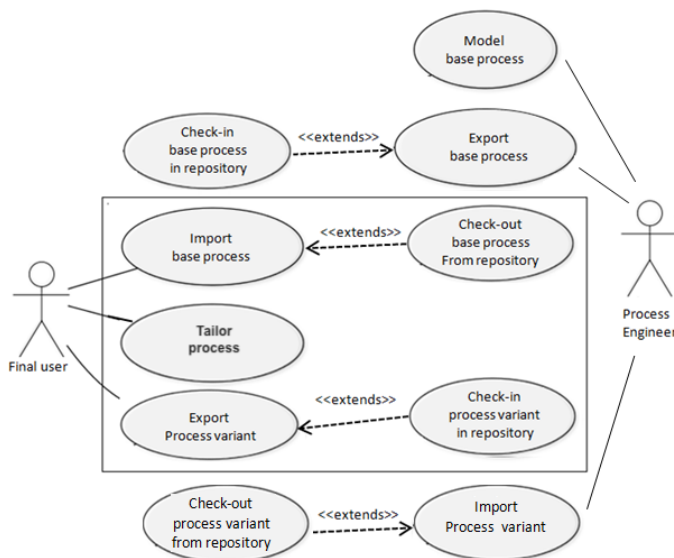


Fig. 2. Use cases for the tailoring process

repository for models, both the base process and the generated variants, such as GitHub<sup>4</sup>. The advantage of this is that the organization can manage their models in a centralized and consistent way, promoting the reuse of elements.

### B. Process tailoring

The result of the process tailoring is a process variant, defining a complete software development process, with no variability, for the specific project we are customizing it to. The process tailoring is part of the more general managing process variability process, that is shown in Figure 3 modeled with BPMN 2.0 as a business process.

As depicted in Figure 3, *Model base process* and *Tailor process* are defined as sub-processes, meaning that several tasks need to be carried out in order to obtain the base process and then, from it, a process variant. We do not explicit the *Model base process* sub-process since it is not the focus of this paper, but we detail the *Tailor process* sub-process.

After obtaining the base process of the family (locally or from the central repository) the final user has to derive the specific variant from it, tailoring the base process. For doing this, the final user has to select each variation point and, from the possible variants, select the one that best fits the project at hand. After this, an automated activity is performed to check the restrictions associated with the selected variation point and corresponding variant. An example of this can be if there was a previous variation point that should have been filled with a specific variant say A, to be able to select the variant B in the following variation point. If the restrictions hold the final user can continue selecting variation points and variants for it, until there are no more variations point. When the restrictions do not hold, the variant cannot be selected.

<sup>4</sup>GitHub. <https://github.com/open-source>

We enclose the import-export tasks in groups, to explicitly show the interactions between the process engineer and the final user via the base process and process variant generated. This supports the use of a centralized repository, where all base process and all generated variants can be stored and reused within the organization. These tasks could have not been shown in the process, since their corresponding work products, i.e.: the process models, can also be seen as input and output data from other activities. We explicitly model them as tasks, since they show user tasks that the defined roles have to carry out within the process.

Finally, after process tailoring is manually defined, we execute an ATL [6] model transformations, automatically generating the corresponding process variant. This transformation takes as an input the base process and the configuration model with the information of the variants that the user selected to occupy each variation point. For now, this variant has to be imported in the EPF so the process engineer is able to publish it as a web site using the facilities provided by the framework. However, we are working on integrating these facilities in the web site, so the generated process variant can be also published by the final user.

### C. Automated generation of variants

The automated generation of variants provides support for a cleaner and repeatable tailoring process, with knowledge reuse and less human errors. The ATL transformation takes as input the base process which was modeled by the process engineer, and the configuration model containing the information of the variants selection made by the final user, when tailoring the process. The transformation contains the knowledge regarding which key elements of the variability modeling extension in the vSPeM metamodel can be mapped to which key elements of the SPeM metamodel, used to specify the generated process variant, based on the information in the configuration model.

Since the base process of the process family also contains elements that are common to all process variants of the family, these elements with no variability have to be maintained in the transformation, as they must be included in all process variants. For this reason, we have separated the generation in two main blocks: i) elements with no variability that have to be copied as they are into the output process variant, and ii) variation points (i.e.: elements with variability) whose selected variant is defined in the configuration model, thus replacing the variation point in the base process with the selected variant.

In Listing 1 we present an excerpt of an adaptation rule to generate a variant from an `Activity` variation point. The transformation takes a model conforming to the v-SPeM metamodel (`Vuma.ecore`) and the configuration model (`Configuration.ecore`) and produces another v-SPeM model (the process variant). The rule takes an activity variation point (`vpActivity`), selects the variant (by id in the configuration model using and auxiliary function) that must be used (`VarActivity`), and generates as output an activity with the information provided by the selected variant.

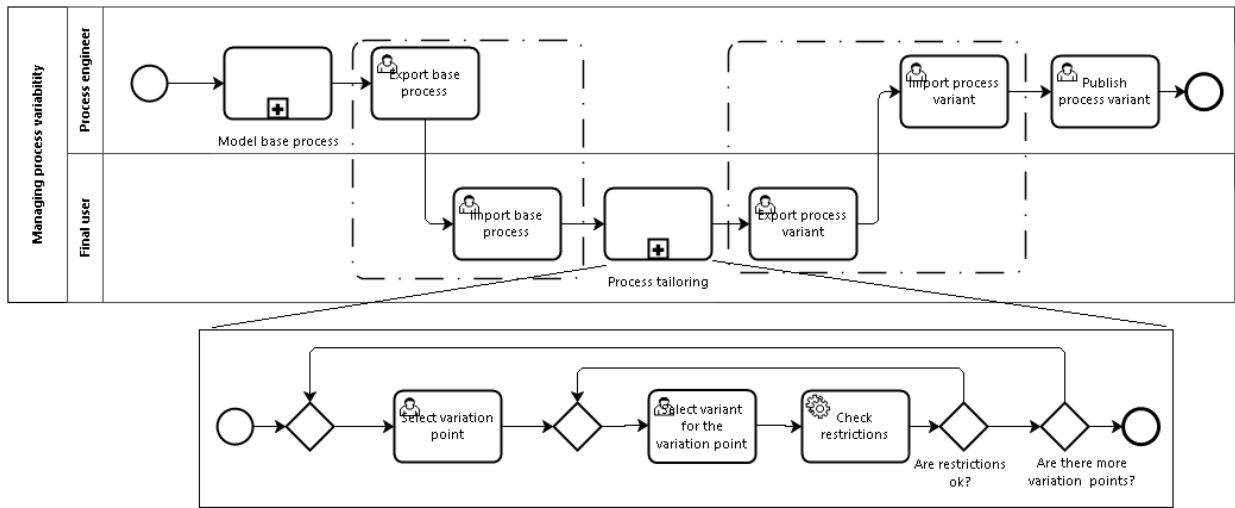


Fig. 3. Managing process variability process

Listing 1. Example of a variant generation rule

```

-- @path MM=/Vuma.ecore
-- @path MM1=/Configuration.ecore
-- @path MM2=/Vuma.ecore

module vSpemYconf;
create OUT: MM2 from IN: MM, IN1: MM1;
.....

//rule to adapt an Activity
rule AdaptActivity {
  from
    input_name: MM!vpActivity (
      input_name->vpProcessElementVa()
    )
  using {
    variant: MM!VarActivity = MM!VarActivity
      -> allInstances() -> any(vt | vt.guid
        = thisModule ->
          getSelectedVariantOfVarPoint
            (input_name.guid).id);
  }
  to
    output_name: MM2!Activity (
      name <- 'AdaptedActivity:' +
        variant.name,
      guid <- input_name.guid,
      presentation <- variant.presentation
    )
}

```

#### D. Tool support

As mentioned before, we use as basis the v-EPF tool that is an extension of EPF to support the v-SPEM proposal. It adds a new folder named Process Lines in which Capability pattern and Delivery process with variability are defined. SPRLs in this folder are structured as shown in Figure 4. Folders *Var Points* and *Variants* contain the variation points and variants defined, also holding the dependence relations

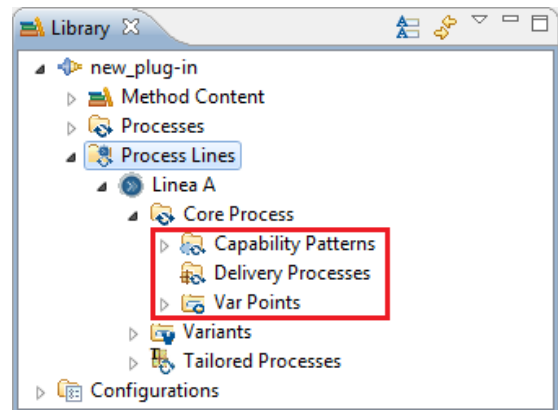


Fig. 4. vEPF tool supporting the vSPEM extension

between variants and variation points. In the Tailored process folder, the process variants that can be derived are specified. Although v-EPF already provides support for the process tailoring, it is aimed at technical users with broadly knowledge of EPF and SPem. Also, process variability does not allow nested variability, and publishing the process in the web is not allowed for processes defined below the process line folder.

Based on the definitions we have presented in previous sections, we also extended the v-EPF tool to support process tailoring with focus on the non-technical final user. For this, we have developed a web application shown in Figure 5.

The web application we have implemented can be used by any final user in the organization and works together with the central repository and the v-EPF used by the process engineer. Figure 5 shows at the top the three main activities (tasks and sub-process) to be performed by the final user as defined in the BPMN 2.0 process shown in Figure 3: (1) import the base process from the repository, (2) perform process tailoring by resolving every variation point (once it is done, the application runs the model transformation), and (3) export process variant.

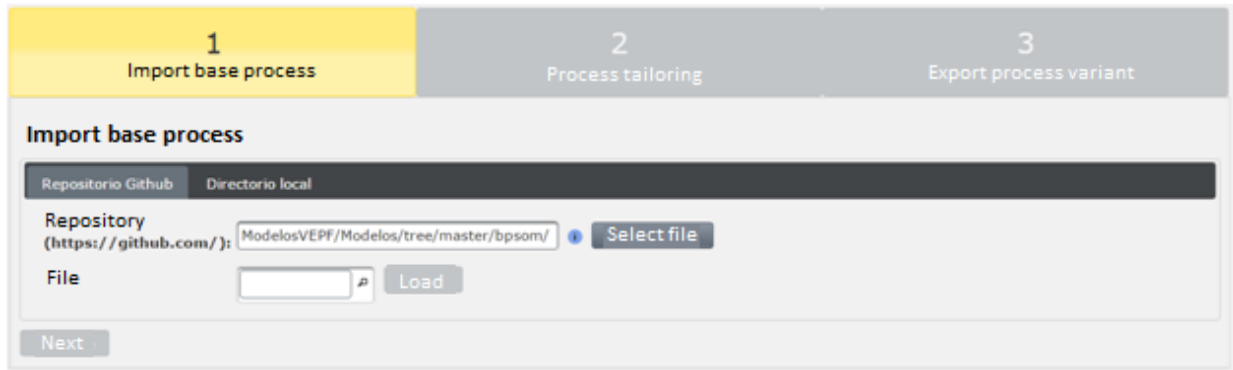


Fig. 5. Screen-shot of the v-SPeM web tool

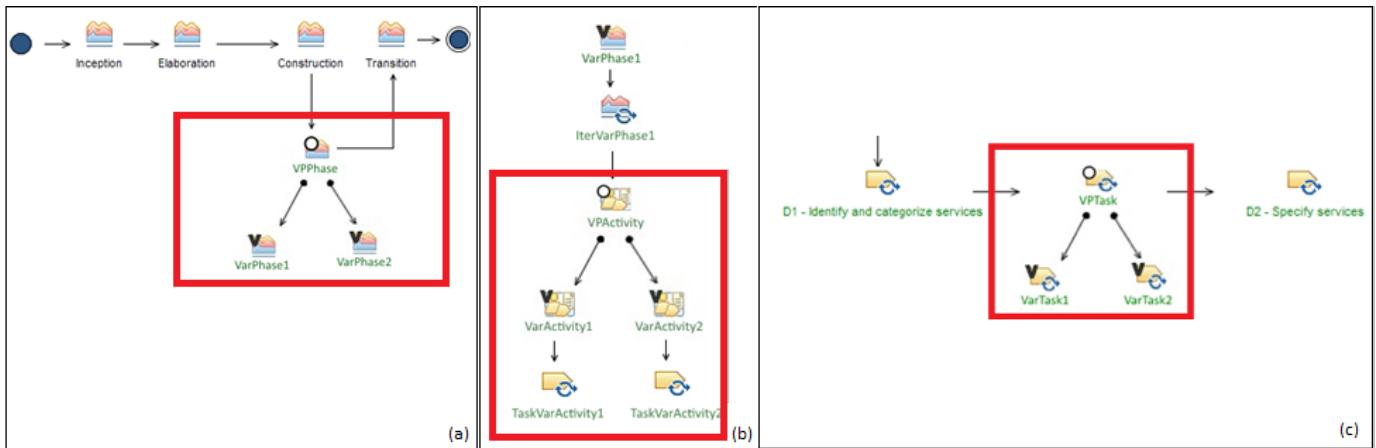


Fig. 6. BPSOM variability with v-SPeM: (a) phase, (b) activity, (c) task

## V. EXAMPLE OF APPLICATION

In this section we present an example of application of our proposal based on the BPSOM<sup>5</sup> software development process we have defined for implementing service-oriented applications from business processes [13]. The process was specified using EPF. It is based on the Unified Process and customized with specific roles, disciplines, activities, work products and a delivery process consisting of four phases.

The process engineer uses v-EPF, under the process line category, to model the base process adding variability in the elements that v-SPeM provides support for: phases, iterations, activities, tasks and roles, as shown in Figure 6. In (a) a new phase between the construction and transaction phases was added, with variants that also have variability for one activity and for one iteration (b); in (c) variability for a task was added, which also has role variability in the corresponding variants. Then, the process engineer exported the base process with variability to a GitHub repository.

The web application allows a final user to perform process tailoring in three steps. First, the base process is imported from the repository. Second, process tailoring is performed as shown in Figure 7. The process is graphically depicted, in this

case: the phases of the BPSOM base process with the phase variation point we added. The variation point has a link that, when pressed, shows the defined variants that can be selected to occupy its place. After selecting the `varPhase1` variant for the `VPPhase` variation point, the variant corresponding to the nested `VPAActivity` variation point must be also defined. In this case, the `VarActivity2` variant is selected, which includes the task `TaskVar2`. This variant also requires to select the role that will perform the `TaskVar2` task, thus the `VarRole1` is selected. Third, the generated variant can be exported to EPF and published as a web site, which facilitates its use as the software development process for the selected project within the organization.

## VI. CONCLUSIONS AND FUTURE WORK

In this paper we presented a proposal that provides support for process tailoring in SPeL based on v-SPeM. Although the approach already provides support for managing variability in SPeL, it does not provide guidance for final users (there were no explicit process) and its tool support for carrying out the process tailoring was hard-coded and within a technical user environment. With this motivation, we specified the activities to be performed for the defined roles and extended the tool support with a new web application focused on process

<sup>5</sup><http://alarcos.esi.uclm.es/minerva/bpsom/published/>

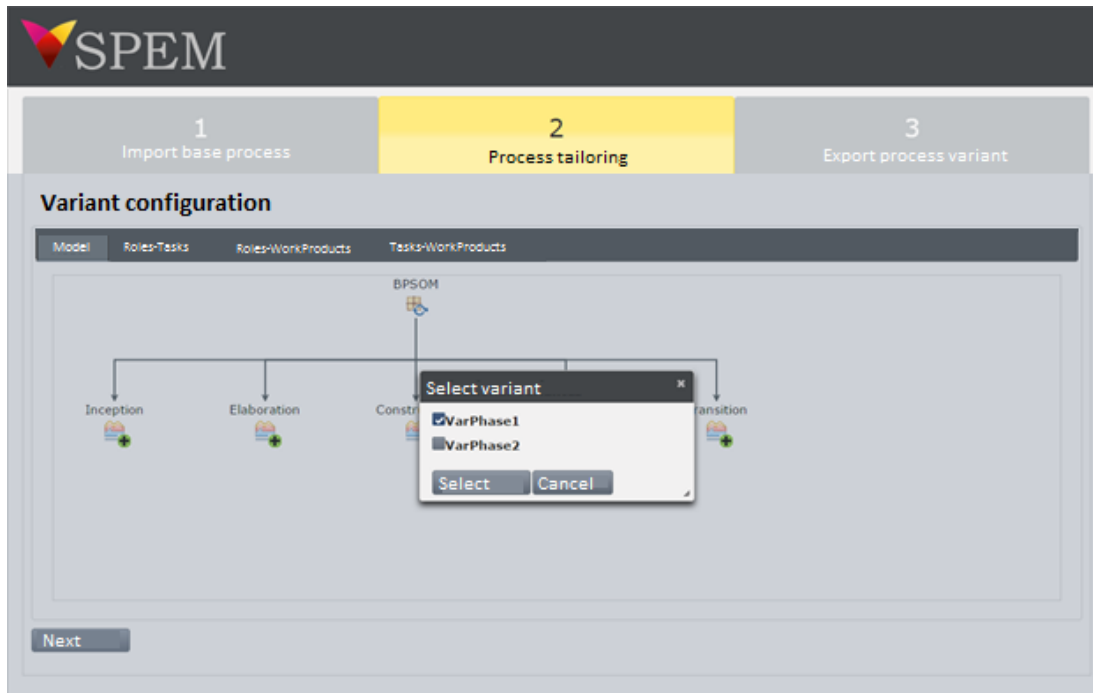


Fig. 7. Web process tailoring: selection of a variant for a variation point

tailoring for final users. The tool provides guides and support for each activity in the tailoring process. Besides it seems that tool support was improved, since the web application is friendlier for non-technical users than the Eclipse environment, and also technical complexity is completely hidden for them, we require further empirical studies with final users in order to conclude. In particular, we expect to evaluate and improve the approach by carry out a case study in a real organization.

The tool also provides organizations with a central repository for SPRL, laying the foundation for a systematic governance of their processes. Moreover, the automated generation of variants provides support for a clearer and less error prone process with respect to the traditional manual tailoring process.

We expect to improve the tool, as for example, publishing generated process variants directly from the web application without the round trip to v-EPF. Moreover, we expect to improve our work for supporting crosscutting variations, as defined in v-SPEM. Finally, although the paper focuses on software processes, the generation of variants approach is quite generic. In fact, we are currently working on applying the approach in the context of BPMN 2.0 business processes [14].

#### ACKNOWLEDGEMENT

This work was partially funded by Comisión Sectorial de Investigación Científica (CSIC), Uruguay. We would like to thank undergraduate students Fabiana Roldán and Marcela Viera, and CSIC grantees Emiliano Alonzo and Martín Prino who worked with the tool and transformation.

#### REFERENCES

[1] OMG, "Software and Systems Process Engineering Metamodel (SPEM) v2.0," Object Management Group (OMG), Tech. Rep., 2008.

[2] D. Rombach, "Integrated software process and product lines," in *Unifying the Software Process Spectrum: Intl. SW Process Works. 2005, Revised Sel. Papers.* Springer, 2006, pp. 83–90.

[3] T. Martínez-Ruiz, F. García, and M. Piattini, "Towards a spem v2.0 extension to define process lines variability mechanisms," in *SW Eng. Research, Mgmt and Applications (SERA).* Springer, 2008, pp. 115–130.

[4] T. Martínez-Ruiz, J. Münch, F. García, and M. Piattini, "Requirements and constructors for tailoring software processes: a systematic literature review," *Software Quality Journal*, vol. 20, no. 1, pp. 229–260, 2012.

[5] S. Kent, "Model driven engineering," in *Integrated Formal Methods, Third Intl. Conf., IFM 2002, Proceedings*, ser. Lecture Notes in Computer Science, vol. 2335. Springer, 2002, pp. 286–298.

[6] F. Jouault, F. Allilaire, J. Bezivin, and I. Kurtev, "Atl: A model transformation tool," *Science of Computer Programming*, vol. 72, no. 1–2, pp. 31 – 39, 2008.

[7] T. Martínez-Ruiz, F. García, M. Piattini, and J. Munch, "Modelling software process variability: an empirical study," *IET Software*, vol. 5, no. 2, pp. 172–187, 2011.

[8] E. Oliveira, M. Pazin, I. Gimenes, U. Kulesza, and F. Aleixo, *SMartySPEM: A SPEM-Based Approach for Variability Management in Software Process Lines.* Springer, 2013, pp. 169–183.

[9] J. Hurtado and M. Bastarrica, "Building software process lines with casper," in *Intl. Conf. on Software and System Process.* IEEE Press, 2012, pp. 170–179.

[10] K. Kang, S. Cohen, J. Hess, W. Novak, and A. Peterson, "Feature-oriented domain analysis (foda) feasibility study," Carnegie Mellon University, Tech. Rep. CMU/SEI-90-TR-021, 1990.

[11] K. Pohl, G. Böckle, and F. van der Linden, *Software Product Line Engineering.* Springer, 2005.

[12] E. Rouille, B. Combemale, O. Barais, D. Touzet, and J. Jezequel, "Leveraging cvl to manage variability in software process lines," in *2012 19th Asia-Pacific Software Engineering Conference*, 2012, pp. 148–157.

[13] A. Delgado, F. Ruiz, I. García, and M. Piattini, "Business process so methodology (BPSOM) with service generation in soaml," in *Advanced Inf. Systems Eng. - 23rd Intl. Conf., CAiSE 2011*, 2011, pp. 672–680.

[14] A. Delgado and D. Calegari, "BPMN 2.0 based modeling and customization of variants in business process families," in *XLIII Latin American Computer Conference, CLEI 2017.* IEEE, 2017, pp. 1–9.

# Tailored Quality Modeling and Analysis of Software-intensive Systems

Robert Heinrich

Karlsruhe Institute of Technology

Karlsruhe, Germany

robert.heinrich@kit.edu

**Abstract**—While developing and operating software-intensive systems various concerns must be considered. Concerns like quality properties, domain, and lifecycle phase may differ from one project to another. Currently different languages and tools are required for modeling and analyzing these concerns. This results in enormous effort for model creation and maintenance. In this paper, we present a vision of tailored quality modeling and analysis by clearly separating several concerns using modular metamodels and tooling.

**Index Terms**—domain-specific modeling language, metamodel, simulation, analysis, quality, reference architecture

## I. INTRODUCTION

Mobility, energy, and infrastructure strongly depend on software which is not always of high quality. Critical performance or security issues may arise from bad software quality. During development and operations of a software-intensive system, various concerns must be considered. Quality properties are examples of concerns which may differ from one project to another. For instance, performance is highly relevant for a web shop whereas for a storage service security may be more relevant. In the context of cloud computing and data-intensive systems, quality properties like privacy can only be assessed reasonably during operations which makes development analysis widely unrewarding. Further, software interacts with other domains (e.g., in Industry 4.0 scenarios), like a business process or mechanics and electronics of an automated production system, where only a subset of domains may be relevant for a certain software project.

For representing a software-intensive system in form of a model, a modeling language is required. Modeling languages are often defined through a metamodel. A metamodel is a model which defines the structure and characteristics of other models. If a model conforms to a metamodel, the model is considered an instance of the metamodel. Thus, a metamodel is similar to a grammar, as it defines a language.

Currently, developers and operators apply different modeling and analysis tools for each of the concerns. Each tool requires specific input models of different languages. Hence, the input models are not integrated and require enormous manual effort for creation and maintenance. Quality is mostly not represented in a domain-specific modeling language (DSML). The commonly agreed DSML in software engineering, the Unified Modeling Language (UML) [20], does not consider quality

properties. Even its extensions, for example MARTE [19] or UMLSec [16], are restricted only to single quality properties. Languages and tools that comprise all possible concerns would be very large, unhandy, and hard to maintain.

The vision proposed in this paper targets more flexibility in Model-Driven Engineering (MDE) by using MDE adaptation capabilities to tailor DSMLs and related tooling to specific concerns. The paper addresses the concerns quality property, domain, and lifecycle phase. We aim for improving efficiency, scalability and reuse of modeling and analysis approaches by clear separation of concerns, focusing the modeling effort only on the relevant concerns, and enabling easy tool customization. Sec. II presents a motivating example before the state of the art is discussed in Sec. III. Our vision of tailored modeling and analysis of various concerns is proposed in Sec. IV and applied prototypically in Sec. V. The paper concludes in Sec. VI.

## II. MOTIVATING EXAMPLE

We introduce the Common Component Modeling Example (CoCoME) as a typical software system to exemplify different configurations that may result from several concerns. We also introduce a DSML to demonstrate limitations in current modeling approaches for various concerns. CoCoME is a community case study for software architecture modeling [12], [8]. It resembles a trading system of a supermarket chain. A software system like CoCoME has to consider several concerns. Several quality properties must be satisfied such as performance, reliability, maintainability, security and privacy. CoCoME interacts with other domains – a business process and an automated production system. Software systems are typically involved in one or more business processes to satisfy business goals. In an Industry 4.0 scenario, software systems interact with automated production systems to enable customized production. An automated production system consists of software, mechanical, and electrical components. Another concern that must be considered is the system's lifecycle phases, i.e. development and operations [7]. Fig. 1 depicts three different configurations of concerns for CoCoME.

For representing CoCoME and the associated concerns in form of models we need a metamodel of a DSML. A prominent example is the UML metamodel. Another historically grown metamodel is the Palladio Component Model (PCM) [23] which we choose for modeling and analyzing CoCoME.

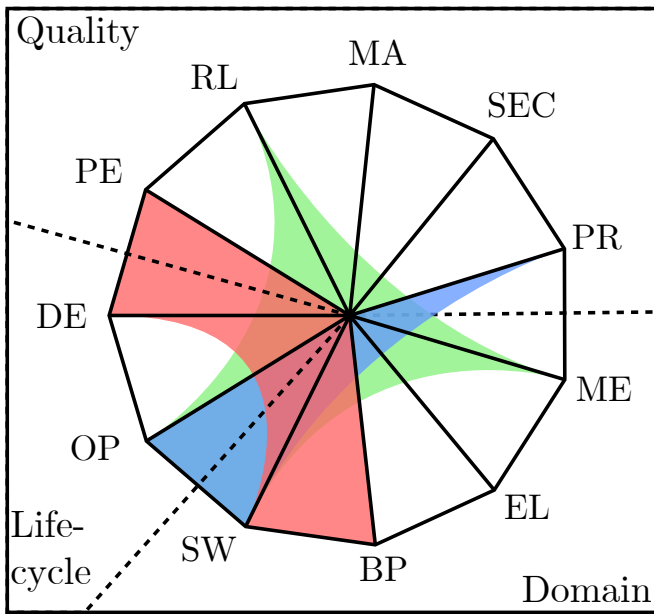


Fig. 1. Colors Represent Three Different Configurations of Concerns for Software-intensive Systems. Concerns Depicted in the Figure are Performance (PE), Reliability (RL), Maintainability (MA), Security (SEC), Privacy (PR), Mechanics (ME), Electronics (EL), Business Processes (BP), Software (SW), Operations (OP), and Development (DE)

In contrast to UML, the PCM comprises elements for reflecting analysis configurations and results. However, the PCM is limited to modeling and analyzing software systems. So we cannot model business processes or automated production systems with the original Palladio approach. Extensions of the PCM to model business processes have been proposed in [9]. Extensions for modeling mechanics and electronics are not yet planned.

Analyzing the various quality properties related to Co-COME currently requires several tools. They all need specific input models in different languages. For example, Queuing Petri Nets may be applied for performance analysis. Markov chains are used for analyzing reliability. Logic programming languages like Prolog may be used for privacy constraint checking. Consequently, high effort is required to create the different models while large parts of the models represent the same structure and behavior in different languages. Moreover, the single models are hard to compare which results in inconsistencies. The example shows existing modeling capabilities are not sufficient for various concerns. Putting all concerns into a large metamodel is just as little a solution. To prevent large and blown metamodels and enable tailoring the models and tools to specific concerns a more modular specification and composition of metamodels and related tooling is required.

### III. STATE OF THE ART

Our work has intersections with three research areas – metamodel modularization, quality modeling, and runtime modeling.

Existing approaches to *metamodel modularization* do not consider architectural characteristics for modularization and do not support the interrelation between metamodel modules and tooling. The underlying model in [1] captures all concerns into orthogonal dimensions which are accessed through views. CORE [26] specifies flexible software modules for model-based and concern-oriented software reuse. Melange [5] and MontiCore [18] reuse and customize legacy modules for creating new languages. An open topic is to provide modularization concepts for DSMLs in direct interplay with related tooling. First attempts came up for modular transformations [22] and generators [15].

*Quality modeling* currently considers only single quality properties in metamodels and lack configurability. Literature reviews give an overview of quality modeling approaches (e.g., [17], [4]) for software systems. Quality properties are also addressed in other domains like business processes (e.g., [3]) and automated production systems (e.g., [30]). In our previous work, we developed an approach for modeling and analyzing maintainability [24]. This approach was first restricted to software systems. The approach has been extended for modeling and analyzing maintainability of business processes [25] and automated production systems [30]. Finally, the approach has been generalized by providing a methodology for domain-spanning maintainability analysis [11]. The promising potential of a generic approach for modeling and analyzing quality has already been recognized in earlier publications [21]. First approaches for generic quality modeling came up decades ago (e.g., [6]). They lack predictive analysis as formal specifications are missing. Formal specifications of quality have been proposed e.g. in [31]. Context-independent modeling of quality is described in [14]. An open topic is using commonalities between quality properties to make quality-related annotations in metamodels configurable.

*Runtime modeling* distinguishes approaches for reusing development models as foundation for reflecting systems during operations and approaches for model extraction from scratch using observations [10]. A comprehensive review of runtime modeling approaches is given in [29]. An open topic is reconfiguring models to changing concerns during operations while keeping all development decisions.

Further, configuration and reuse are central to *software product lines and eco systems*. While this research is limited to the instance level, our work refers to the metamodel level.

### IV. A VISION OF TAILORED QUALITY MODELING AND ANALYSIS

Our vision is to provide a reference architecture for metamodels that enables clear separation of several concerns in quality modeling and analysis. Beyond considering different quality properties the reference architecture comprises multiple domains related to software (i.e., business processes, mechanics, and electronics), and life-cycle phases (i.e., development and operations). Modelers will be enabled to customize their tooling by composing modular specifications for each of the concerns as desired for their project and run



transformations to create model editors and solvers. Prior publications (cf. Sec. III) already raised the need for a generic approach to quality modeling and analysis. A reference architecture for metamodels for quality modeling and analysis is an ambitious goal. In the past it was assumed to be unattainable as quality properties were considered to be too different in their relation to system's architecture and context. Consequently, a generic approach was assumed to be too abstract for adequate predictive analysis. Recent advances, however, lead to the conclusion that the challenges can now be overcome as: (i) Research on formalization of single qualities (e.g., [2]) resulted in much deeper understanding of similarities which we can use for the reference architecture. (ii) Research on mutual quality impact between different domains (e.g., [9]) provides starting points for specification of quality-dependent inter-domain relations. (iii) A first reference architecture for single qualities in a DSML for software systems [27] is starting point to more generic investigation for different concerns. (iv) Research on simulation-based quality analysis (e.g., [23]) is foundation for a general approach to generic tooling. The vision requires innovation in several areas as detailed hereafter.

*Metamodel Modularization and Composition:* Foundation to the reference architecture is a concept for composition of modular metamodels. If modular metamodels are not already available, we first need to identify a set of dimensions and elaborate criteria along which metamodels can be modularized. One way is to specify dimensions based on characteristics of metamodels. As we focus on metamodels for quality modeling and analysis in various domains, dimensions like paradigm, domain, quality specification, and quality analysis appear natural. Prior work on dimensions is given in [27]. Then, existing model extension mechanisms (e.g., inheritance, profiles, stereotypes, and aspects) for composing metamodel modules created along the dimensions can be investigated and mapped to composition operators. The operators are used to build composable tooling by running transformations. Finally, we can construct the reference architecture by exploiting the concepts for metamodel modularization and composition.

*Extensible DSML:* The reference architecture then enables the structured extension of existing DSMLs (e.g., PCM [23]) by comprehensive specifications of quality properties. In a software DSML, systems are typically described as compositions of components by connectors made explicit through interfaces. These generic modeling concepts can be extended to support the specification of various quality properties and their context dependencies. Using modularization and composition capabilities eases the extension of a software DSML to the domains business process and automated production system by composing the corresponding modular metamodels. Further investigation is necessary on how the fundamental concepts of a DSML (i.e., composition and connectors) known from software modeling can be applied to other domains.

*Reconfiguration for Operational Concerns:* While building upon the extensible DSML we can reconfigure development models to changing concerns during operations while keeping all development decisions. Reconfiguration cannot be limited

to DSMLs and tools but also affects monitoring probes required to observe the quality properties in the changed focus when running the system.

*Modular Tooling:* For providing tools tailored to specific concerns, the notion of modular metamodels for several concerns must be expanded to modular construction of modeling and analysis tools. Aforementioned concepts for metamodel modularization and composition are foundation for modular tooling. Explicitly specified dependencies between quality properties of several domains mark the points where different analytical and simulative solvers need to interact. Composable tools for modeling and analysis can be built using the composition operators. Foundations already exist for coupling simulative solvers [9].

## V. PROTOTYPICAL APPLICATION

This section gives concrete examples for metamodel modularization and composition to demonstrate the applicability of the reference architecture and tool modularization.

*Application of the Reference Architecture:* The PCM in its current form is focused on single quality properties yet does not reflect the various concerns related to our motivating example (cf. Fig. 1). For each concern we must provide specific metamodels as extensions to the PCM. For tailoring the DSML to specific concerns, the PCM and its extensions are divided into four dimensions – paradigm, domain, quality, and analysis – along which the metamodels can be modularized. Previous work limited to metamodels of software systems is given in [27] and [10].

The four dimensions have been chosen as (i) they represent generic characteristics of a DSML for quality modeling and analysis. (ii) Their hierarchical nature eases the composition of metamodels assigned to the single dimensions. In the reference architecture the modularization dimensions are represented as layers ordered hierarchically with respect to their dependencies. Metamodel modules of one layer may only depend on modules of the same or more foundational layers.

The modularization of the PCM according to the reference architecture is depicted prototypically for the domains software system and business process as well as for the quality properties maintainability and performance in Fig. 2. The figure shows a very simplified representation of the modular PCM for visualization in this paper. Each rectangle reflects a modular metamodel that may extend another metamodel and can itself be extended. Arrows depict general relationships between metamodel modules. Within a certain module the relationship is implemented by composition operators between the meta-classes of one and another module.

The paradigm layer (II) defines the foundational concepts without any semantics, e.g. componentization. Here the component module and the activity module specify the core entities to describe structure and behavior. Composition by connectors is specified for both in the component composition and activity composition module respectively. The data module provides foundational concepts for specifying data flows such as source and sink.

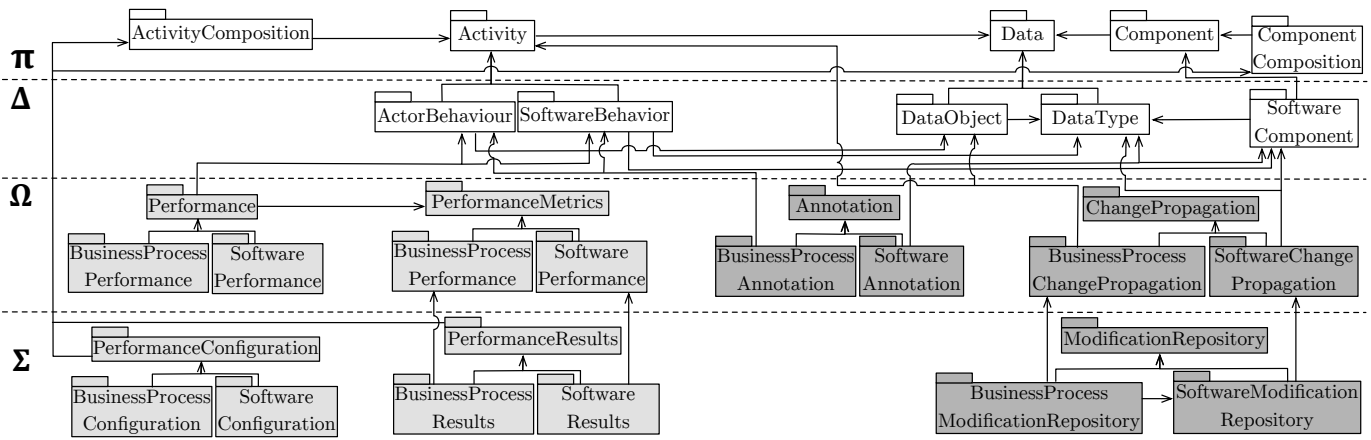


Fig. 2. Prototypical Modularization of the PCM for Different Domains and Quality Properties

The domain layer ( $\Delta$ ) extends  $\Pi$  by domain-specific semantics. Here the software components and their interfaces are specified within the software component module. The software behavior module extends the software component module by a behavior abstraction for software similar to flow charts. A business process on an atomic level comprises activities either conducted by human actors or machines (here the software system) [9]. To specify a business process, the software behavior module and the actor behavior module extend the activity module. Thus, the software behavior module serves as the connecting link between business process modeling and software modeling. Moreover, the data object and data type modules extend the generic data module by specifications of data used in business processes and software services. The modules on  $\Delta$  layer are used in subsequent layers for performance and maintainability modeling.

The quality layer ( $\Omega$ ) defines quality properties, primarily in form of second class entities which enrich the first class entities of  $\Delta$ . Quality properties on  $\Omega$  are not derived by analysis [27]. Here,  $\Omega$  comprises modular metamodels to reflect performance (light grey) and maintainability (dark grey) properties. The performance module comprises modeled performance properties which are specialized for software and business processes in the extending modules. For example, the software performance module specifies resource demands of a service while the business process performance module reflects execution time of a human activity [9]. The same structure applies to the performance metrics module which comprises abstract metrics to be specialized for software and business processes.

The annotation module contains abstract specifications of artifacts annotated to first class entities on  $\Delta$ . The extending modules specialize these artifacts, e.g. test cases for software components [24] or training material for human activities [25]. Maintainability analysis in a PCM instance is conducted as change propagation analysis using the KAMP approach [24]. Once a component or activity changes the test cases or training material may change, too. Starting with a seed modification

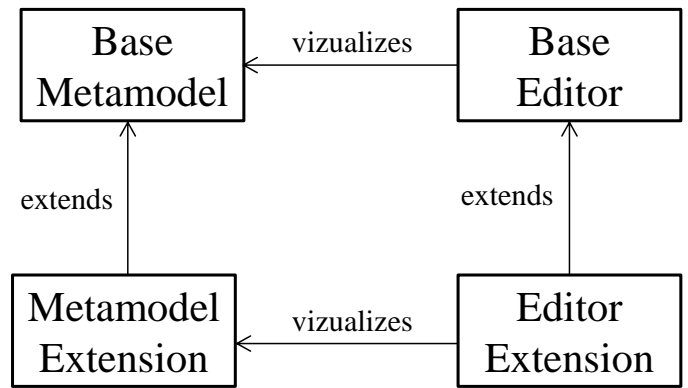


Fig. 3. Modular Base Editor and its Extension

changes propagate through instances of the entities on  $\Delta$  which may change instances of the entities on  $\Omega$ . Elements to specify this change propagation are contained in the change propagation module and specialized by the extending modules.

The analysis layer ( $\Sigma$ ) is required if models are used for analysis or simulation. It comprises metamodels for derived performance properties in the results module and the simulation configuration in the configuration module. The modification repository module reflects the origin of a change and the result of a change propagation analysis.

Modularizing the PCM according to the reference architecture is key for modularizing the related tooling as discussed hereafter. Again we use the example of software systems and business processes for demonstration.

*Application of Modular Tooling:* The modularized PCM and all its extensions form a directed acyclic graph. The tooling and all its extensions must mirror this structure. Next, we discuss how model editors, simulative and analytical solvers – that are closely related to metamodels – can be implemented for modular metamodels.

Model editors allow for visualizing and modifying model elements and thus are obviously related to metamodels. For exemplifying the implementation of *modular editors* we build

upon the eclipse modeling framework and the Sirius editor framework. Details on the implementation of the editors for the Palladio tooling are given in [28]. The implementation of modular editors is depicted schematically in Fig. 3. Rectangles represent modular metamodels and modular tools. Arrows reflect relationships between the modules or instances of the modules. The relationships are labeled to further specify the sort of relationship. Sirius offers the possibility to extend a diagram representation by further layers without altering the implementation of the base diagram intrusively. Analogously to a modular base metamodel and its extensions, editor extensions are packaged in their own Eclipse plugins. Applying the reference architecture, modelers can customize their tooling by selecting plug-ins for the specific metamodel modules and corresponding editors. Base metamodel in our example is those for software systems which is composed of metamodel modules on the layers  $\Pi$  to  $\Delta$  of the reference architecture. The base editor comprises all features required for visualizing and modifying elements of software systems. The base metamodel is extended by metamodel modules on the layers  $\Pi$  to  $\Delta$  for representing business processes. Furthermore, metamodel modules on  $\Omega$  layer extend the base metamodel by performance and maintainability properties. Therefore, modular editors for business processes and the quality properties performance and maintainability extend the base editor.

More sophisticated solutions are required for modular simulations. For *modular simulations* we sketch an online co-simulation of a software system and a business process in Fig. 4. We conduct a simulation of performance properties as common in Palladio [23]. A discussion on benefits and limitations of online co-simulation is given in our previous work [9]. We have instances of two metamodels – one for the software system and one for the business process – that are composed of metamodel modules on the layers  $\Pi$  to  $\Delta$ . Additionally, the metamodels contain modeled performance properties and metrics on  $\Omega$ . Each metamodel has its specific modular tooling – the simulative solvers – which are interlinked by a coordinator. The coordinator is responsible for time management and model synchronization to coherently integrate the modular simulations. Simulation configuration and results are specified on  $\Sigma$ . The simulative solvers assure technical interoperability by providing an interoperability layer for enabling the coordinator to interact with the simulations. There already exists approaches to couple simulations based on a common runtime infrastructure, e.g. High-Level Architecture [13], which can be applied to build a coordinator.

Coupling *modular maintainability analyzes* for software systems and business processes again requires modular metamodels and analysis tools. In contrast to simulation coupling, there is no coordinator needed for synchronization. Consistent notion of time is not required in modular maintainability analyzes. Fig. 5 shows a metamodel for software systems composed of modules on layers  $\Pi$  to  $\Delta$ . The metamodel also contains modules for artifact annotations and change propagation on  $\Omega$  as well as for analysis results [24] on  $\Sigma$ . Instances of

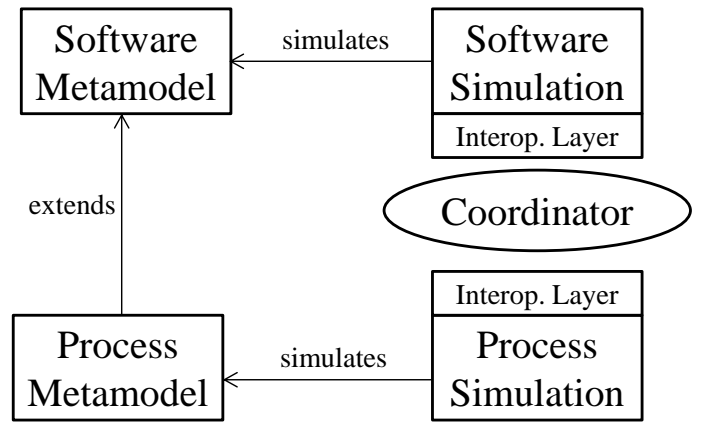


Fig. 4. Modular Performance Simulation of Software System and Business Processes

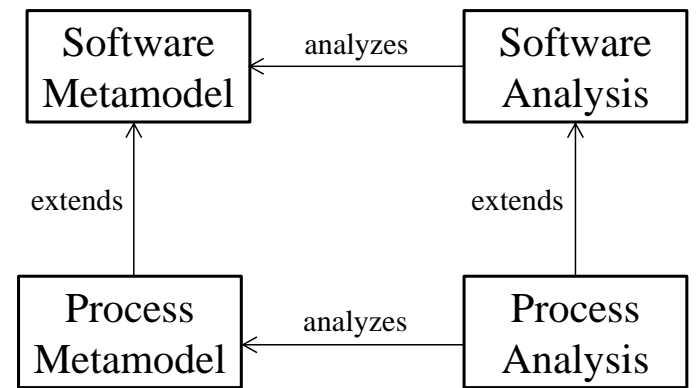


Fig. 5. Modular Maintainability Analyzes of Software Systems and Business Processes

this metamodel are analyzed by the modular software analysis tool. The metamodel for software systems may be extended by another metamodel to reflect business processes on  $\Pi$  to  $\Delta$ . This again contains modules for annotations on  $\Omega$  and analysis results on  $\Sigma$ . Instances of the business process metamodel are analyzed by the modular business process analysis tool which extends the software analysis tool. Due to the modular nature we can decide whether to include the metamodel modules specific to business processes or not. This allows for tailoring metamodels and tooling to specific purposes.

*Reconfiguring* development models to changing concerns during operations is also part of our research vision. With the iObserve approach [7] we provide first attempts for updating development models by operational observations to construct runtime models. iObserve bridges the divergent levels of abstraction in architectural models used in development and operations. An architectural model is combined with monitoring probes and used for source code generation. Monitoring data related to source code artifacts is gathered during system operation and is associated with architectural runtime model elements. Thus, iObserve allows for reusing development models during operation phase while preserving design decisions.

The iObserve approach is described in detail in [7]. Metamodel modularization and composition in this context is sketched in [10].

*Findings:* The prototypical application of our modularization and composition concepts for metamodels and tooling revealed several findings as summarized hereafter. Based on the PCM it is easy to see that a modularized metamodel according to our reference architecture provides many benefits compared to a metamodel without modular structure. Modularization offers a well specified base for extension as demonstrated for business processes and quality properties. Due to the hierarchical structure, modelers merely need to understand the modules they directly or indirectly extend. Side effects of changes to modules are minimized. Thus, clear separation of concerns allows modelers to focus modeling effort only on concerns relevant for a specific project. The greatest advantage for our research is that metamodel modules can be reused in different contexts. This allows for tailoring the metamodels to specific concerns. Tools can be customized easily for specific metamodels as exemplified for model editors, simulative and analytical solvers. Thus, the concepts proposed in this paper contribute to more efficient, scalable and reusable modeling and analysis of various concerns.

## VI. CONCLUSION

In order to enable clear separation of several concerns we proposed a reference architecture for metamodels used for quality modeling and analysis. We sketched modular tooling for model editors, analytical and simulative solvers. We demonstrated the applicability of the reference architecture and modular tooling to a historically grown metamodel for software systems. Our prototypical application comprised the quality properties performance and maintainability for the two domains. Advantages of modular metamodels and tooling are a well structured base for metamodel extension, minimization of side effects in case of modification, focus only on relevant concerns, reuse of modular metamodels and tools as well as project-specific configuration of metamodels and tools.

In the future, we will continue the modularization of existing metamodels and related tooling in several case studies to evaluate, expand and sharpen our approach. This includes further investigation of technologies for metamodel composition and simulation coupling.

## ACKNOWLEDGEMENT

This work was supported by the MWK (Ministry of Science, Research and the Arts Baden-Württemberg, Germany) in the funding line Research Seed Capital (RiSC). The author thanks Kiana Busch, Misha Strittmatter and Sandro Koch for valuable discussion and support for this paper.

## REFERENCES

[1] C. Atkinson *et al.*, “Orthographic software modeling: A practical approach to view-based development,” in *Evaluation of Novel Approaches to Software Engineering*, vol. 69. Springer, 2010, pp. 206–219.  
 [2] S. Becker *et al.*, “Towards a methodology driven by dependencies of quality attributes for QoS-based analysis,” in *ICPE*. ACM, 2013, pp. 311–314.

[3] J. Cardoso *et al.*, “Modeling quality of service for workflows and web service processes,” *Journal of Web Semantics*, vol. 1, pp. 281–308, 2002.  
 [4] L. Dai and K. Cooper, “A survey of modeling and analysis approaches for architecting secure software systems,” *Journal of Network Security*, vol. 5, pp. 187–198, 2007.  
 [5] T. Degueule *et al.*, “Melange: A meta-language for modular and reusable development of DSLs,” in *SLE*. ACM, 2015, pp. 25–36.  
 [6] S. Frolund and J. Koistinen, “QML: A language for quality of service specification,” Hewlett-Packard Software Technology Laboratory, Tech. Rep. HPL-98-10, 1998.  
 [7] R. Heinrich, “Architectural run-time models for performance and privacy analysis in dynamic cloud applications,” *Perform. Eval. Rev.*, vol. 43, no. 4, pp. 13–22, 2016.  
 [8] R. Heinrich *et al.*, “A platform for empirical research on information system evolution,” in *SEKE*, 2015, pp. 415–420.  
 [9] —, “Integrating business process simulation and information system simulation for performance prediction,” *Software & Systems Modeling*, vol. 16, no. 1, pp. 257–277, 2017.  
 [10] —, *Software Architecture for Big Data and the Cloud*. Elsevier, 2017, ch. An Architectural Model-Based Approach to Quality-aware DevOps in Cloud Applications.  
 [11] —, “A methodology for domain-spanning change impact analysis,” in *SEAA*. IEEE, 2018.  
 [12] S. Herold *et al.*, “CoCoME – the common component modeling example,” in *The Common Component Modeling Example*. Springer, 2008, pp. 16–53.  
 [13] IEEE, “Standard for modeling and simulation High Level Architecture,” 2000.  
 [14] K. Jezek *et al.*, “Towards context independent extra-functional properties descriptor for components,” *Electron. Notes Theor. Comput. Sci.*, vol. 264, no. 1, pp. 55–71, 2010.  
 [15] R. Jung *et al.*, “GECO: A generator composition approach for aspect-oriented DSLs,” in *ICMT*. Springer, 2016, pp. 141–156.  
 [16] J. Jürjens, “UMLsec: Extending uml for secure systems development,” in *UML*. Springer, 2002, pp. 412–425.  
 [17] H. Koziolok, “Performance evaluation of component-based software systems: A survey,” *Perform. Eval.*, vol. 67, no. 8, pp. 634–658, 2010.  
 [18] H. Krahn *et al.*, “MontiCore: Modular development of textual domain specific languages,” in *TOOLS EUROPE 2008*. Springer, 2008, pp. 297–315.  
 [19] Object Management Group, *UML Profile for MARTE*, Object Management Group Std., 2011.  
 [20] —, *Unified Modeling Language, Version 2.4.1*, Object Management Group Std., Rev. 2.4.1, 2011.  
 [21] D. C. Petriu, “Challenges in integrating the analysis of multiple non-functional properties in model-driven software engineering,” in *WOSP-C*. ACM, 2015, pp. 41–46.  
 [22] A. Rentschler, *Model Transformation Languages with Modular Information Hiding*. KIT Scientific Publishing, 2015.  
 [23] R. H. Reussner *et al.*, *Modeling and Simulating Software Architectures – The Palladio Approach*. MIT Press, 2016.  
 [24] K. Rostami *et al.*, “Architecture-based assessment and planning of change requests,” in *QoSA*. ACM, 2015, pp. 21–30.  
 [25] —, “Architecture-based change impact analysis in information systems and business processes,” in *ICSA*. IEEE, 2017, pp. 179–188.  
 [26] M. Schöttle *et al.*, “On the modularization provided by concern-oriented reuse,” in *Modularity*. ACM, 2016, pp. 184–189.  
 [27] M. Strittmatter *et al.*, “A modular reference structure for component-based architecture description languages,” in *ModComp*. CEUR, 2015, pp. 36–41.  
 [28] —, “Extensible graphical editors for palladio,” in *7th Symposium on Software Performance*, 2016.  
 [29] M. Szvetits and U. Zdun, “Systematic literature review of the objectives, techniques, kinds, and architectures of models at runtime,” *Software & Systems Modeling*, vol. 15, no. 1, pp. 31–69, 2016.  
 [30] B. Vogel-Heuser *et al.*, “Maintenance effort estimation with KAMP4aPS for cross-disciplinary automated PLC-based production systems - a collaborative approach,” in *20th IFAC World Congress, IFAC-PapersOnLine*, vol. 50, 2017, pp. 4360–4367.  
 [31] S. Zschaler, “Formal specification of non-functional properties of component-based software systems,” *Software & Systems Modeling*, vol. 9, no. 2, pp. 161–201, 2010.

# Modeling and Verifying Leader Election Algorithm in CSP

Yucheng Fang   Huibiao Zhu\*   Huiwen Wang  
Shanghai Key Laboratory of Trustworthy Computing,

School of Computer Science and Software Engineering, East China Normal University, China

**Abstract**—Leader election is a fundamental problem in distributed systems and has a variety of applications in wireless networks, such as key distribution, routing coordination, and general control. The main statement of the leader election problem is to eventually elect a unique leader from a fixed set of nodes. As the wireless network is becoming more and more important in daily life, leader election algorithm plays a vital important role in wireless network, which makes the correctness and robustness of such algorithms become evermore important and challenging to establish. In this paper, firstly, we study an election algorithm LE for MANETs (Mobile Ad Hoc Network) designed by Vasudevan et al. Then we present a formal model for LE based on process algebra CSP (Communicating Sequential Process). Modeling algorithm like LE sometimes pose non-trivial challenges, time, geometry, communication delays and failures, mobility and bi-directionality can interact in unforeseen ways that are hard to model and analyze by automatic formal methods, but we will take on these challenges. On that basis, we use the model checker FDR (Failures Divergence Refinement) to automatically simulate the developed model and verify whether the model is consistent with the specification and exhibits relevant secure properties. Our results show the correctness and safety of LE in this respect.

**Index Terms**—leader election, formal methods, CSP, FDR

## I. INTRODUCTION

In a network, leader election algorithm is to select a unique leader of each node in a network. It is a fundamental control problem in distributed systems and has a variety of applications in wireless networks, such as key distribution [7], routing coordination [2] and sensor coordination [8].

The purpose of electing a leader in an interconnected network is to permit the control of the network by a unique node in order to perform a specific action or activity with the other members of the network. Several algorithms have been proposed to solve this problem such as [5], [6]. Only a few of the proposed algorithms can be applied to MANETs. In this paper, we focus on the algorithm, which is called LE, designed by Vasudevan et al. in [11]. It aims at electing the most-valued node according to some measure, e.g., the amount of remaining battery life in a network. In LE, several spanning trees were established. Then, these spanning trees were reduced to a unique spanning tree and the root to decide which is the leader in the network. In this paper, we mainly model LE in a context of static topology, under the assumption that nodes with its neighbors are fixed the nodes number of the network would not increase when LE is running. We give a

detailed explanation in next section. Our intention is to prove that an abstraction of the LE works correctly.

There actually exist many research on leader election algorithm [1], [13]. Most of them use model checking based on a specific network such as ring [13] but seldom of them consider to verify LE. In this paper we formalize LE based on process algebra CSP and we concentrate on the status changing of a node, modeling the operations of each node. Our work build a baseline for verifying LE in CSP way. Based on the formalized model, we use FDR to automatically simulate the achieved model and verify whether it caters for some significant properties such as deadlock freedom, divergence-free and unique leader scheme.

CSP is a formal language for describing patterns of interaction in concurrent systems [9] and it has been practically applied in industry as a tool for specifying and verifying the concurrent aspects of a variety of different systems, such as [12]. There are many model checkers for CSP, such as FDR [3], Process Analysis Toolkit (PAT) [10] and so on. FDR has the best performance among them because it includes a parallel refinement-checking engine that achieves a linear speed-up as the number of cores increase. It is able to check processes with billions of states, and is able to make efficient use of on-disk storage to complement memory.

The remainder of this paper is organized as follows. In Section 2, we give a brief introduction to LE, and the process algebra CSP. In Section 3, we model LE in CSP and In Section 4, we give three basic properties and verify that the LE model respects them. In Section 5, we conclude and outline the future work.

## II. BACKGROUND

In this section, we give a brief introduction to LE algorithm and give an example to illustrate. Further, we also present the relevant introduction on CSP.

### A. Brief Introduction to LE

When an election is triggered at a node, the node broadcasts an *election* message to its immediate neighbors (one hop neighbors). A node that receives an *election* message for the first time, records the sender of the message as its *parent* in the spanning tree under construction, and multicasts an *election* message to its other immediate neighbors. When a node receives an *election* message from a node that is not its parent, it immediately responds with an *ack* message. When

\* Corresponding Author. Email: hbzhu@sei.ecnu.edu.cn

a node has received *ack* messages from all of its children, it sends an *ack* message to its parent. Each such *ack* message to a parent includes the identity and value of the most-valued node in the subtree rooted at the sender. Therefore, when the source node has received an *ack* message from all of its children, it can determine the most-valued node in the entire spanning tree. The source node then broadcasts a *leader* message to all of its immediate neighbors to announce a new leader. When a node receives a *leader* message, it updates its own leader and broadcasts it to its immediate neighbors.

Fig. 1 shows a run of LE under a static topology of five nodes, with node 1 being the source and node 5 being the most-valued node. In this figure, thin arrows indicate the direction of flow of messages and thick arrows indicate parent pointers. These parent pointers together represent the constructed spanning tree. Node 1 starts its diffusing computation by sending out *election* messages to its immediate neighbors 2 and 3, shown in Fig. 1(a). As indicated in Fig. 1(b), nodes 2 and 3 set its parent pointer to point node 1 and in turn propagate an *election* message to all their neighbors except their parent nodes. Hence 2 and 3 send *election* to each other, as we explain before, they will send *ack* to each other immediately but not taken the other as its parent. In Fig. 1(c), a complete spanning tree is built. In Fig. 1(d), nodes 4 and 5 send its value to its parent nodes, since they are the leaves of the tree. Eventually, the source 1 hears pending acknowledgments from both 2 and 3 in Fig. 1(e) and then broadcasts the identity of the leader, 5, via *leader* message shown in Fig. 1(f).

Multiple nodes can concurrently initiate multiple elections; in this case, only one election should “survive”. This is done by associating to each election a *priority*, so that a node already in an election ignores incoming elections with lower priority, but participates in an election with higher priority. In some cases, a node maybe *fail* and will not send *ack* to its parent. To handle the case like this, every node sets a expire time  $T$ , when the time exceeds  $T$ , the node will be removed from its parent’s waiting list. The report [11] gives a detailed pseudo-code specification of LE.

### B. Brief Introduction to CSP

The CSP method, abbreviation for Communicating Sequential Processes, is first proposed by C.A.R Hoare [4]. It is a process algebra designed mainly for analyzing the behaviors of concurrent processes. In CSP, a synchronous communication mechanism holds when processes communicate with each other to coordinate the parallel executions. The syntax of a subset of the CSP language is given as follows.

$$P, Q = \text{Skip} \mid \text{Stop} \mid a \rightarrow P \mid c?x \rightarrow P \mid c!x \rightarrow P \mid P \square Q$$

$$P \parallel Q \mid P \parallel\parallel Q \mid Q \setminus M \mid P; Q \mid \text{if } b \text{ then } P \text{ else } Q$$

where:

- *Stop* represents that the process does nothing and its state is deadlock.

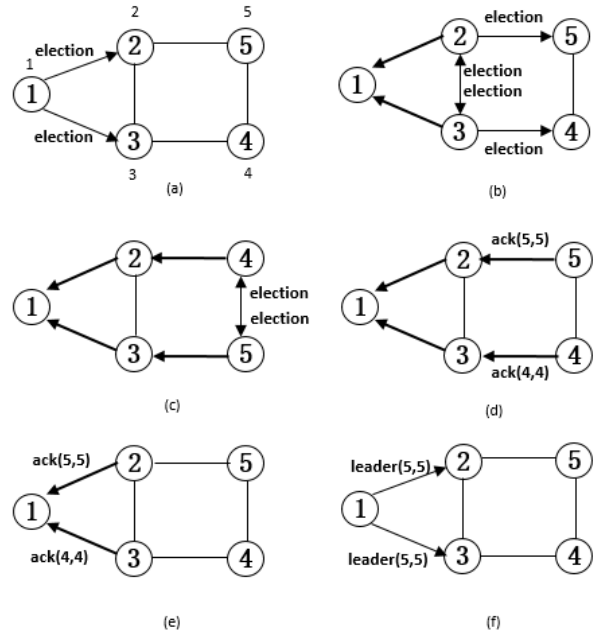


Fig. 1. An LE run in a static topology

- *Skip* stands for a process which terminates successfully.
- $a \rightarrow P$  first performs the event  $a$  then behaves like  $P$ .
- $c!v \rightarrow P$  sends message  $v$  through channel  $c$ , then performs like  $P$ .
- $c?x \rightarrow P$  receives a message through channel  $c$  and assigns it to a variable  $x$ , then does the subsequent behaviors like  $P$ .
- $P \square Q$  acts like either  $P$  or  $Q$  and lets the environment decide the selection.
- $P \parallel Q$  shows the parallel composition between  $P$  and  $Q$ .
- $Q \setminus M$  acts like  $Q$ , except all events from the set  $M$  are hidden.
- $P \parallel\parallel Q$  indicates the process chooses to perform actions in  $P$  and  $Q$  randomly.
- $P; Q$  executes  $P$  and  $Q$  sequentially.
- $\text{if } b \text{ then } P \text{ else } Q$  denotes the conditional choice. If the value of  $b$  is true then it behaves like  $P$  else like  $Q$ .

### III. MODELING LE

In this section, we present a CSP model of LE. The formalization is carried out based on the introduction to LE which has been described in Section 2. Firstly, we give the whole structure of our model and then we model each process respectively.

#### A. Parameters in Model

To clarify the whole system, we give the channels and messages used in LE. They are described as follows.

- *election*: message for a node to pass an election message
- *ack*: to acknowledge receipt of an *election* message
- *leader*: to announce the new leader
- *probe*: to determine if a node is still alive

- *reply*: sent in response to a *probe* message

In our model, every process has some variables to help it make a decision when an action occurs. They are list as follows.

- $d_i$ : a binary variable indicating if  $i$  is currently in an election or not
- $p_i$ :  $i$ 's parent node in the spanning tree
- $D_i$ : a binary variable indicating if  $i$  has sent an *ack* to  $p_i$  or not
- $lid_i$ :  $i$ 's leader
- $N_i$ :  $i$ 's current neighbors
- $S_i$ : set of nodes from which  $i$  is yet to hear an *ack* from
- $src_i$ :  $i$ 's priority
- $max_i$ : the most-valued node  $i$  has record so far

In our model the status of each node can be in one of eight statuses: *BeginElection*, *WaitFor*, *SendElection*, *AwaitAck*, *AwaitLeader*, *SendInfo*, *SendLeader* and *Running*. Their state transition diagram is shown in Fig. 2. Node  $i$  can start an election by sending message *election*. After sending message to all of its immediate neighbors, it enters into state *AwaitAck*. In this state,  $i$  sets a expire time  $T$  for each of node in its waiting list. If a node has no answer in  $T$  units,  $i$  removes it from its waiting list. After receiving all of the *ack* messages, node  $i$  sends message *ack* to its parent and enters into state *SendInfo*. In state *SendInfo*, if the election is start by its own, it will enter into state *SendLeader* to send the *leader* message, otherwise it will enter into state *AwaitLeader* to wait a *leader* message from its parent. When a node finishes sending *leader* message or receives a *leader* message, it will enter into state *Running*. We will discuss each process respectively.

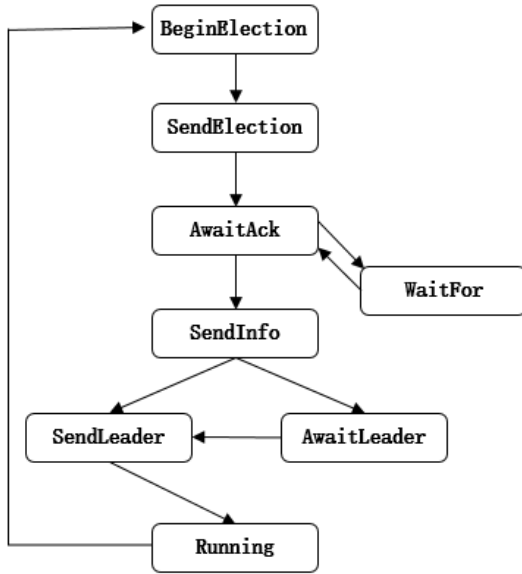


Fig. 2. State Transition Diagram

## B. Modeling each Process

1) *BeginElection*: In process *BeginElection*,  $n$  denotes its id number and  $N$  represents its neighbors.

$$BeginElection(n, N) \hat{=}$$

$$SendElection(n, True, n, false, -1, N, N, n, n, N)$$

*BeginElection* starts an election by sending *election* messages to its one-hop neighbors through process *SendElection*.

2) *SendElection*: Process *SendElection* is shown in Fig. 3. In process *SendElection*,  $n$  is the id number of process *SendElection*. The eight variables between  $n$  and  $N'$  have been explained in previous section.  $N'$  is a set that records all nodes which have not been sent *election* message by the current node. Action  $election.n!i : N'!src$  represents that node  $n$  sends *election* message to its neighbor node  $i$  with election priority  $src$ . When  $N'$  is empty, it enters into process *AwaitAck*. During its sending action, it may receive another *election* message from its neighbor and it will compare the priority of these two elections (i.e.,  $s.p > src.p$  in Fig. 4) and decide which election to take part in. If it receives an *ack* message from its neighbor, it removes it from its waiting list  $S$ . The last three statements, i.e., *probe*, *fail* and *tock*, describe three basic actions of each process. A node can be tested whether it is alive or not by its parent via channel *probe*. A node may fail and enters into a state *Fail* or it just does nothing but wait the time pass by via channel *tock*.

3) *AwaitAck*: Process *AwaitAck*, its main job is to wait *ack* message from all neighbors in  $S$ . The process is shown in Fig. 3. When it receives an *ack* message from its neighbor, it removes the neighbor from its waiting list. *AwaitAck* tests whether a node is still alive or not by sending message *probe* and sets a expire time  $T$  then it enters into process *WaitFor*. If it receives an *election* message, like process *SendElection*, it compares the priority of these two elections and decides which election to take part in. In our model, since every node can start an election, so there may exist a local leader and therefore when a node receives a *leader* message, it will compare the priority of these two elections and decide to take part in which election.

4) *WaitFor*: Process *WaitFor* will wait the *reply* message from  $Node_{pid}$  until the time exceeds  $T$  or it receives a *reply* message.

$$WaitFor(n, d, p, D, lid, N, S, src, max, pid, T) \hat{=}$$

if  $T == 0$

then  $AwaitAck(n, d, p, D, lid, N, S \setminus \{pid\}, src, max)$

else  $tock \rightarrow WaitFor(n, d, p, D, lid, N, S, src, max, pid, T - 1)$

$\square reply.n.pid \rightarrow AwaitAck(n, d, p, D, lid, N, S, src, max)$

5) *SendInfo*: It judges whether the election is start by its own. If is, it enters into *SendLeader*. Otherwise, it sends *ack*

```

SendElection( $n, d, p, D, lid, N, S, src, max, N'$ )  $\hat{=}$ 
  if empty( $N'$ ) then AwaitAck( $n, d, p, D, lid, N, S, src, max$ )
  else election.n!i :  $N'!src \rightarrow SendElection(n, d, p, D, lid, N, S, src, max, N' \setminus \{i\})$ 
 $\square$ election? $c : N!n?s \rightarrow$  (if  $s.p > src.p$  then  $SendElection(n, d, c, False, -1, N, N \setminus \{c\}, s, max, N \setminus \{c\})$ )
  else  $ack!n!c!max \rightarrow SendElection(n, d, p, D, lid, N, S, src, max, N')$ 
 $\square$ ack? $c : S!n?v \rightarrow SendElection(n, d, p, D, lid, N, S \setminus \{c\}, src, Max(max, v), N')$ 
 $\square$ leader? $c : N!n?s \rightarrow$  (if  $s.p > src.p$  then  $SendElection(n, d, c, False, -1, N, N \setminus \{c\}, s, max, N \setminus \{c\})$ )
  else  $SendElection(n, d, p, D, lid, N, S, src, max, N')$ 
 $\square$ probe? $c \in N!n \rightarrow reply!c!n \rightarrow SendElection(n, d, p, D, lid, N, S, src, max, N')$ 
 $\square$ fail.n  $\rightarrow Faild(n, N)$ 
 $\square$ tock  $\rightarrow SendElection(n, d, p, D, lid, N, S, src, max, N')$ 

```

Fig. 3. Process SendElection

```

AwaitAck( $n, d, p, D, lid, N, S, src, max$ )  $\hat{=}$ 
  if empty( $S$ ) then SendInfo( $n, d, p, D, lid, N, S, src, max$ )
  else tock  $\rightarrow AwaitAck(n, d, p, D, lid, N, S, src, max)$ 
 $\square$ ack? $c : S!n?v \rightarrow AwaitAck(n, d, p, D, lid, N, S \setminus \{c\}, src, Max(max, v))$ 
 $\square$ probe? $c : N!n \rightarrow reply!c!n \rightarrow AwaitAck(n, d, p, D, lid, N, S, src, max)$ 
 $\square$ probe! $n?j : S \rightarrow WaitFor(n, d, p, D, lid, N, S, src, max, j, T)$ 
 $\square$ tock  $\rightarrow AwaitAck(n, d, p, D, lid, N, S, src, max)$ 
 $\square$ fail.n  $\rightarrow Faild(n, N)$ 
 $\square$ election? $c : N!n?s \rightarrow$  (if  $s.p > src.p$ 
  then  $SendElection(n, d, c, False, -1, N, N \setminus \{c\}, s, max, N \setminus \{c\})$ 
  else (if  $s == src$  then  $ack!n!c!max \rightarrow AwaitAck(n, d, p, D, lid, N, N \setminus \{c\}, src, max)$ 
  else  $AwaitAck(n, d, p, D, lid, N, S, src, max)$ ))
 $\square$ leader? $c : N!n?s \rightarrow$  (if  $s.p > src.p$ 
  then  $SendElection(n, d, c, False, -1, N, N \setminus \{c\}, s, max, N \setminus \{c\})$ 
  else  $election.n!c!src \rightarrow AwaitAck(n, d, p, D, lid, N, S, src, max)$ )

```

Fig. 4. Process AwaitAck

message to its parent and enters into process *AwaitLeader*.

```

SendInfo( $n, d, p, D, lid, N, S, src, max$ )  $\hat{=}$ 
  if  $src == n$ 
  then  $SendLeader(n, d, p, D, max, N, S, src, max, N)$ 
  else  $ack!n!p!max \rightarrow$ 
     $AwaitLeader(n, d, p, True, max, N, S, src, max)$ 
 $\square$ tock  $\rightarrow SendInfo(n, d, p, D, lid, N, S, src, max)$ 
 $\square$ fail.n  $\rightarrow Faild(n, N)$ 
 $\square$ probe.p.n  $\rightarrow reply.p.n \rightarrow$ 
   $SendInfo(n, d, p, D, lid, N, S, src, max)$ 
 $\square$ election? $c : N!n?s \rightarrow$  (if  $s.p > src.p$  then
   $SendElection(n, d, c, False, -1, N,$ 
   $N \setminus \{c\}, s, max, N \setminus \{c\})$ 
  else  $SendInfo(n, d, p, D, lid, N, S, src, max)$ )

```

6) *Fail*: At any time, one node may enter into state *Fail*. In this process, it absorbs all messages. It is shown as follows.

```

Faild( $n, N$ )  $\hat{=}$  tock  $\rightarrow Faild'(n, N)$ 
 $\square$ probe? $c : N!n \rightarrow Faild(n, N)$ 
 $\square$ election? $c : N!n?v \rightarrow Faild(n, N)$ 
 $\square$ ack? $c : N!n?v \rightarrow Faild(n, N)$ 
 $\square$ leader? $c : N!n?s \rightarrow Faild(n, N)$ 
Faild'( $n, N$ )  $\hat{=}$  tock  $\rightarrow Faild'(n, N)$ 
 $\square$ probe? $c : N!n \rightarrow Faild(n, N)$ 
 $\square$ election? $c : N!n?v \rightarrow Faild(n, N)$ 
 $\square$ ack? $c : N!n?v \rightarrow Faild(n, N)$ 
 $\square$ leader? $c : N!n?s \rightarrow Faild(n, N)$ 
 $\square$ revive.n  $\rightarrow BeginElection(n, N)$ 

```



A process enters into *Fail* at least one unit then it can enter into process *Fail'*. In this state, it also absorbs all the actions but it can revive as well.

7) *SendLeader*: Process *SendLeader* sends the *leader* message to all of its one-hop neighbors and then enters into process *Running*. If it receives an *election* message, it compares the priority of the two elections and decides whether to take part in the new election or to ignore this the message.

$$\begin{aligned} & \text{SendLeader}(n, d, p, D, lid, N, S, src, max, N') \hat{=} \\ & \text{if empty}(N') \\ & \text{then Running}(n, false, n, D, lid, N, S, src, max) \\ & \text{else leader.n!i : } N' \setminus max \rightarrow \\ & \quad \text{SendLeader}(n, d, p, D, lid, N, S, src, max, N' \setminus \{i\}) \\ \square \text{tock} & \rightarrow \text{SendLeader}(n, d, p, D, lid, N, S, src, max, N') \\ \square \text{fail.n} & \rightarrow \text{Faild}(n, N) \\ \square \text{election?c : } N \setminus n?s & \rightarrow \text{if } s.p > src.p \text{ then} \\ & \quad \text{SendElection}(n, d, c, False, -1, N, N \setminus \{c\}, s, max, N \setminus \{c\}) \\ & \quad \text{else SendLeader}(n, d, p, D, lid, N, S, src, max, N') \end{aligned}$$

8) *AwaitLeader*: Process *AwaitLeader* just waits for *leader* message from its parent. When it receives the leader information, it will enter into process *SendLeader* to pass this message to its neighbors.

$$\begin{aligned} & \text{AwaitLeader}(n, d, p, D, lid, N, S, src, max) \hat{=} \\ & \text{leader.p.n?v} \rightarrow \\ & \quad \text{SendLeader}(n, d, p, D, v, N, S, src, v, N \setminus \{p\}) \\ \square \text{tock} & \rightarrow \text{AwaitLeader}(n, d, p, True, lid, N, S, src, max) \\ \square \text{fail.n} & \rightarrow \text{Faild}(n, N) \\ \square \text{probe.p.n} & \rightarrow \text{reply.p.n} \rightarrow \\ & \quad \text{AwaitLeader}(n, d, p, D, lid, N, S, src, max) \\ \square \text{election?c : } N \setminus n?s & \rightarrow \text{if } s.p > src.p \text{ then} \\ & \quad \text{SendElection}(n, d, c, False, -1, N, N \setminus \{c\}, s, max, N \setminus \{c\}) \\ & \quad \text{else if } s == src \text{ then ack.n!c!max} \rightarrow \\ & \quad \quad \text{AwaitLeader}(n, d, p, D, lid, N, S, src, max) \\ & \quad \quad \text{else AwaitLeader}(n, d, p, D, lid, N, S, src, max) \end{aligned}$$

9) *Running*: Process *Running* represents a normal state of a node. The CSP code is shown as follows.

$$\begin{aligned} & \text{Running}(n, d, p, D, lid, N, S, src, max) \hat{=} \\ \square \text{election?c : } N \setminus n?v & \rightarrow (p(v) > p(lid) \text{ or } \neg d) \& \\ & \quad \text{SendElection}(n, d, c, False, -1, N, \\ & \quad \quad N \setminus \{c\}, v, max, N \setminus \{c, v\}) \\ \square \text{tock} & \rightarrow \text{Running}(n, d, p, D, lid, N, S, src, max) \\ \square \text{fail.n} & \rightarrow \text{Faild}(n, N) \\ \square \text{probe?c : } N \setminus n & \rightarrow \text{reply!c!n} \rightarrow \\ & \quad \text{Running}(n, d, p, D, lid, N, S, src, max) \end{aligned}$$

When it receives an *election* message, if it has no leader (it has not taken in any election) or the priority of the node which

starts this election is greater than its leader, it will take part in this election by entering into state *SendElection*.

#### IV. VERIFICATION

In this section, we implement CSP model and verify some important properties in FDR. Before we do the verification, we should construct our network. To illustrate our model is correct. We choose a topology Fig. 1 in Section 2 to do verification.

$$\begin{aligned} \text{Proc} & = \{1..5\} \\ \text{NEIGHBORS} & \hat{=} \{(1, 2), (1, 3), (2, 3), (2, 5), (3, 4), (4, 5)\} \\ \text{Neighbors}(x) & \hat{=} \{k \mid k \leftarrow \text{Proc}, \text{member}((x, k), \text{NEIGHBORS}) \\ & \quad \text{or member}((k, x), \text{NEIGHBORS})\} \end{aligned}$$

*Proc* is a set which contains five numbers. We stores all the edges in set *NEIGHBORS* and *Neighbors(x)* is a function which returns a set of all neighbors of *x*. We formalize a node as follows.

$$\text{Node}(x) \hat{=} \text{if } x == 1 \text{ then BeginElection}(x, \text{Neighbors}(x)) \text{ else Running}(x, false, x, false, x, \text{Neighbors}(x), \text{Neighbors}(x), x, x)$$

Therefore, we build our network as follows. *Alpha(n)* denotes the synchronized action set of node *n*. For instance  $\text{Alpha}(1) = \{\text{ack}.1.2, \text{ack}.2.1, \dots, \text{tock}\}$  where “...” indicates some channels like *reply*, *probe* and actions between 1 and 3.

$$\text{Network} \hat{=} \parallel n : \text{Proc} \bullet [\text{Alpha}(n)] \text{Node}(n)$$

##### (1) DeakLock Freedom

In LE, deadlock freedom means process network can move on at any time. In FDR, we use statement below to do that.

$$\text{assert Network} : [\text{deadlock free } [F]]$$

The verification result is shown in Fig. 5. The first statement in the *Assertions* block is for deadlock freedom and the green dot on the left side shows that this assertion is passed, which means our system is deadlock free. The block under the *Assertions* is the *Tasks* block, in which the actual checking steps lay here.

##### (2) Divergence Freedom

A divergence of a process is that any trace of the process has a point after which the process behaves chaotically. Divergence freedom assures our model is well-defined without ambiguousness. We use statement below to complete the verification.

$$\text{assert Network} : [\text{divergence free } [F]]$$

The checking result is shown in Fig. 5., the second statement in *Assertion* block.

##### (3) Unique Leader

The main goal of LE is that it will eventually select a unique leader, which is the most-valued-node among the nodes in the component. In our model, leader information is sent by *leader* message, and therefore we only concentrate on message *leader* and hide the other channels.

$$\begin{aligned} \text{network} & \hat{=} \text{Network} \setminus \{ \mid \text{probe}, \text{fail}, \text{ack}, \\ & \quad \text{election}, \text{tock}, \text{reply}, \text{revive} \} \end{aligned}$$

In the topology which we prepare to verify, we know that the most-valued-node is 5 and its value is 5 so we formalize the property as follows.

$$UniLeader \hat{=} leader?c : Proc?d : Proc!5 \rightarrow UniLeader$$

Process *UniLeader* means if there exists a *leader* message the *id* which it sends must be 5. Process *UniLeader* contains all the traces which elect node 5 as the leader. We use refinement to complete this check. If  $A [= B$  is true (where  $[=$  represents refinement) then the behaviours of *B* are contained within the behaviours of *A*. By showing  $UniLeader [T = network$ , we can conclude that our model satisfies the unique leader scheme. The verification result is shown in Fig. 5.

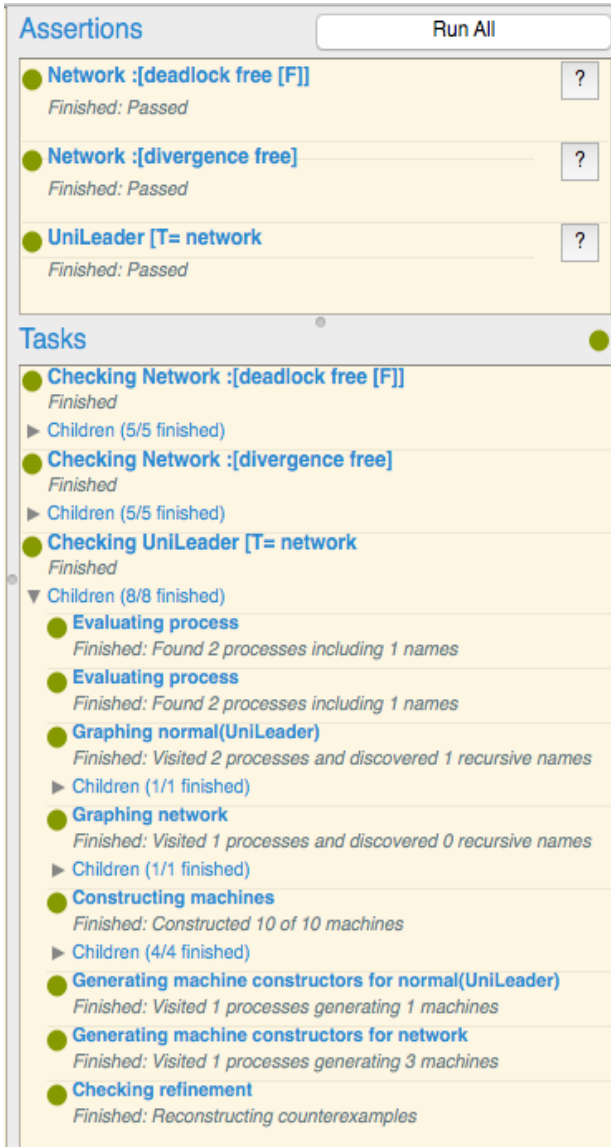


Fig. 5. Verification Results

## V. CONCLUSION

In this paper, we have constructed the formal models for LE, modeled the operations of each state of LE in CSP. We

also verified the LE model using FDR. We constructed the specific models based on three properties including deadlock freedom, divergence freedom and unique leader scheme. The results show that our model satisfies all those properties, indicating the LE get a strong robustness and consisting with the specification.

In mobile ad hoc network, node can transfer from one position to another which results that a node will connect to a new network and disconnect from the old one. Therefore, in the future, we will modify the model to fit the situation like this and based on the new model we do other checks to verify LE.

**Acknowledgement.** This work was supported by Shanghai Collaborative Innovation Center of Trustworthy Software for Internet of Things (No. ZF1213).

## REFERENCES

- [1] A. Ansari. Verification of Peterson's Algorithm for Leader Election in a Unidirectional Asynchronous Ring Using NuSMV. *CoRR*, abs/0808.0962, 2008.
- [2] S. Bhattacharya, J. Kulkarni, and V. S. Mirrokni. Coordination mechanisms for selfish routing over time on a tree. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 186–197, 2014.
- [3] T. Gibson-Robinson, P. Armstrong, A. Boulgakov, and A. Roscoe. FDR3 — A Modern Refinement Checker for CSP. In E. Abraham and K. Havelund, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 8413 of *Lecture Notes in Computer Science*, pages 187–201, 2014.
- [4] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.
- [5] S. Kim, V. Vasireddy, and K. Harfoush. Scalable coordination for sensor networks in challenging environments. In *Proceedings of the 2007 ACM Symposium on Applied Computing (SAC), Seoul, Korea, March 11-15, 2007*, pages 214–221, 2007.
- [6] A. Mazezev, A. Semenov, and A. Simonov. A Distributed Parallel Algorithm for Minimum Spanning Tree Problem. *CoRR*, abs/1610.04660, 2016.
- [7] A. Mehmood, M. M. Umar, and H. Song. ICMSD: secure inter-cluster multiple-key distribution scheme for wireless sensor networks. *Ad Hoc Networks*, 55:97–106, 2017.
- [8] Y. Nakamura, M. Louvel, and H. Nishi. Coordination middleware for secure wireless sensor networks. In *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society, Florence, Italy, October 23-26, 2016*, pages 6931–6936, 2016.
- [9] A. W. Roscoe. *Understanding Concurrent Systems*. Texts in Computer Science. Springer, 2010.
- [10] J. Sun, Y. Liu, and J. S. Dong. Model Checking CSP Revisited: Introducing a Process Analysis Toolkit. In *Leveraging Applications of Formal Methods, Verification and Validation, Third International Symposium, ISOFA 2008, Porto Sani, Greece, October 13-15, 2008. Proceedings*, pages 307–322, 2008.
- [11] S. Vasudevan, J. F. Kurose, and D. F. Towsley. Design and Analysis of a Leader Election Algorithm for Mobile Ad Hoc Networks. In *12th IEEE International Conference on Network Protocols (ICNP 2004), 5-8 October 2004, Berlin, Germany*, pages 350–360, 2004.
- [12] L. Wang, F. Sui, Y. Huang, and H. Zhu. Modeling and Verifying the Ballooning in Xen with CSP. In *16th IEEE International Symposium on High Assurance Systems Engineering, HASE 2015, Daytona Beach, FL, USA, January 8-10, 2015*, pages 18–25, 2015.
- [13] L. Xu and P. Jeavons. Simple algorithms for distributed leader election in anonymous synchronous rings and complete networks inspired by neural development in fruit flies. *Int. J. Neural Syst.*, 25(7), 2015.

# A Formal Approach for Distributed Computing of Maximal Cliques in Dynamic Networks

Faten Fakhfakh<sup>1</sup>, Mohamed Tounsi<sup>1</sup>, Mohamed Mosbah<sup>2</sup>, and Ahmed Hadj Kacem<sup>1</sup>

<sup>1</sup> ReDCAD Laboratory, University of Sfax, FSEGS, Tunisia

<sup>2</sup> LaBRI Laboratory, University of Bordeaux, France

faten.fakhfakh@redcad.org, mohamed.tounsi@redcad.org,

ahmed.hadjkacem@fsegs.rnu.tn, mohamed.mosbah@labri.fr

**Abstract**—The aim of this work is to propose a distributed algorithm, encoded by the local computations model, for computing maximal cliques in dynamic networks. This model provides an abstraction which simplifies the design and the proof of distributed algorithms. To guarantee the correctness of our algorithm, we use the Event-B formal method, which supports a refinement based incremental development using the RODIN platform.

**Keywords**—Distributed algorithm, Local computations, Dynamic networks, Maximal cliques, Event-B method.

## I. INTRODUCTION

### A. Motivation

Computing maximal cliques is a challenging problem in computer science which has found applications in several fields such as robotics, bioinformatics, etc. Many efforts have been dedicated to solve the problem of detecting maximal cliques. Most of the proposed approaches have used centralized algorithms and only few of them [6] [2] are based on distributed algorithms [7]. In the context of dynamic networks, the nodes are dynamically connected in an arbitrary manner without any established infrastructure or centralized administration. So, the topology of the network may change rapidly and unpredictably. Considering the complexity of distributed algorithms and the highly dynamic behavior, it is interesting to ensure the correctness of these algorithms to give us confidence that distributed systems perform as designed and do not behave harmfully.

According to our study, we notice that the majority of the existing works [2] [8] [6] for detecting maximal cliques rely on simulation to evaluate the performance of these solutions. Some works [2] [6] have proved the correctness of their algorithms based on formal proofs. Nevertheless, the proofs which have been presented are done manually. Also, these proofs are long and tedious specially in the case of complex algorithms and a minor error can have serious consequences on the system operation. To the best of our knowledge, only the solution of Xu et al. [8] dealt with dynamic networks. However, it has an exponential complexity.

### B. Contribution

We propose in this paper a distributed algorithm for computing maximal cliques in dynamic networks inspired by the work of Luo et al. [6] which has a linear complexity. Our algorithm is based on the local computations model [5] that

provides an abstraction for the design of distributed algorithms. To model dynamic networks, we use the evolving graph model [3] which consists in recording the evolution of the network topology as a discrete sequence of static graphs. Each static graph represents a snapshot of the dynamic network at a given date. In order to prove the correctness of the proposed algorithm, we use a formal method which offers a real help for expressing correctness with respect to safety properties in the design of distributed algorithms. The *correct-by-construction* approach [4] offers a simple way to specify and prove algorithms. It consists in developing distributed algorithms following a top/down approach controlled by the refinement of models. This process allows to simplify the proofs and validate the integration of requirements. The Event-B modeling language [1] can support this methodological proposal by suggesting proof based-guidelines.

### C. Organization of the paper

The remainder of this paper is structured as follows: Section II introduces our proposed algorithm. In Section III, we specify this algorithm with the Event-B method. Finally, the last section concludes and provides insights for future work.

## II. ALGORITHM FOR COMPUTING MAXIMAL CLIQUES IN DYNAMIC NETWORKS : CMCDN

A network can be modeled as a simple and undirected graph  $g=(V,E)$  where  $V$  is the set of nodes and  $E$  is the set of edges. In this work, we suppose that every node in the graph knows its neighbours. A “*clique*” is a fully connected (or complete) subgraph of the graph  $g$  and a “*maximal clique*” is a clique that is not a subset of any other clique in the same graph. The proposed algorithm may be encoded by the graph relabeling system  $R = (L, I', P)$ . For a given node  $x$ ,  $L = \{State, Clique, N, Event\}$ ,  $I' = \{Init, \emptyset, N(x), False\}$ , and  $P = \{R1, R2, R3, R4, R5, R6\}$ . *State* and *Clique* are two functions. The node  $x$  has the following four labels:

- **State(x)**  $\in \{Init, C, I, W, A\}$  is the state of the node  $x$ . It can take one of these labels: i) **Init** : the node  $x$  is in the initial state, ii) **C** : we call the node which detects a maximal clique “*the center*” of this clique, iii) **I** : the node  $x$  belongs to a maximal clique and it is different to the center of this clique, iv) **W** : the node  $x$  is in the waiting state. It has not been assigned to a maximal clique yet, v) **A** : the node  $x$  does not belong to any maximal clique. It is called “*isolated node*”.
- **N(x)** : It stores the node  $x$  and all its neighbours. The set  $N(x)$  will be updated when an incident edge is removed.

- **Clique(x)** : The node  $x$  belongs to the maximal clique “Clique(x)”. Initially, each node  $x$  of the graph has  $Clique(x) = \emptyset$ .

- **Event(x) ∈ {False, Alert}** : It determines whether the node  $x$  belongs to an edge that has undergone a topological change or not. Initially, all nodes are in the state “False”. If an edge  $u \mapsto x$  of a maximal clique has disappeared, then  $Event(u)$  and  $Event(x)$  take the state “Alert” to transmit a reconstruction order of the concerned clique.

$I'$  is the set of initial labels and ( $R1$ ,  $R2$ ,  $R3$ ,  $R4$ ,  $R5$  and  $R6$ ) are the relabeling rules. For each node  $x \in V$ , we define  $S(x) = \bigcap_{k \in N(x)} N(k)$  the set of nodes that stores

the intersection of all  $N(k)$  ( $k \in N(x)$ ). We note  $R(x)$  all the nodes of  $S(x)$  which have not been assigned to maximal cliques yet. Therefore, these nodes are labeled “Init” or “W”.

To handle the disappearance of an edge during the construction of maximal cliques in a graph, we distinguish two cases :

**Case 1 :** Deleting an edge of the graph (an edge that does not belong to any maximal clique) has no effect on the maximal cliques already detected. This case requires applying the rule  $R4$  (see Section II-D).

**Case 2 :** Deleting an edge belonging to a maximal clique can cause the destruction of this clique or the possibility of creating a new maximal clique.

#### A. The first rule: R1

The first rule aims to construct a maximal clique in the graph. Let  $x$  be a node in the initial state and  $R(x)$  including at least three nodes. As a result, the nodes of  $R(x)$  form a maximal clique having  $x$  as center. Formally, the rule  $R1$  is written as follows:

Precondition :

- $State(x) = Init$
- $card(R(x)) \geq 3$

Relabeling :

- $State(x) := C$  and  $Clique(x) := R(x)$
- $\forall a \cdot a \in R(x) \setminus \{x\} \implies State(a) := I$  and  $Clique(a) := R(x)$

#### B. The second rule: R2

The goal of the rule  $R2$  is to attribute to a node that is unable to form a maximal clique the waiting state. This rule requires the presence of a node  $x$  in the initial state and  $R(x)$  containing less than three nodes. Formally, the rule  $R2$  is written as follows:

Precondition :

- $State(x) = Init$
- $card(R(x)) < 3$
- $\exists k \cdot k \in N(x) \setminus \{x\}$  et  $State(k) = Init$

Relabeling :

- $State(x) := W$

#### C. The third rule: R3

The purpose of the rule  $R3$  is to identify an isolated node which does not belong to any maximal clique. Let  $x$  be a node in the initial or the waiting state and all the neighbouring nodes of  $x$  are not in the initial state. By applying  $R3$ , the label of the node  $x$  becomes “A” and  $Clique(x)$  contains only the node  $x$ . Formally, the rule  $R3$  is written as follows:

Precondition :

- $State(x) \in \{Init, W\}$
- $\forall a \cdot a \mapsto x \in g \implies State(a) \neq Init$

Relabeling :

- $State(x) := A$  and  $Clique(x) := \{x\}$

#### D. The fourth rule: R4

The purpose of this rule is to express the modification made when deleting an edge of the graph. It requires the presence of an edge ( $u \mapsto v$ ) that does not belong to any detected clique. The application of the rule  $R4$  causes the deletion of the edge  $u \mapsto v$  from the graph as well as the updating of the sets  $N(u)$  and  $N(v)$ . Formally, the rule  $R4$  is written as follows:

Precondition :

- $u \mapsto v \in g$
- $\neg ((Clique(u) = Clique(v))$  and  $(u \in Clique(u)$  and  $v \in Clique(u)))$
- $Event(v) = False$  and  $Event(u) = False$

Relabeling :

- $g := g \setminus \{u \mapsto v\}$
- $N(u) := N(u) \setminus \{v\}$  and  $N(v) := N(v) \setminus \{u\}$

#### E. The fifth rule: R5

The role of this rule is to reflect the preliminary influence of the deletion of an edge belonging to a maximal clique. Let  $u$  and  $v$  be two nodes of the same maximal clique. While applying the rule  $R5$ , the edge  $u \mapsto v$  disappears and  $Event(u)$  and  $Event(v)$  take the state “Alert”. In addition, the sets  $N(u)$  and  $N(v)$  are updated. Formally, this rule is written as follows:

Precondition :

- $u \mapsto v \in g$
- $Clique(v) = Clique(u)$
- $State(v) \in \{C, I\}$  and  $State(u) \in \{C, I\}$
- $Event(v) = False$  and  $Event(u) = False$

Relabeling :

- $g := g \setminus \{u \mapsto v\}$
- $N(u) := N(u) \setminus \{v\}$  and  $N(v) := N(v) \setminus \{u\}$
- $Event(v) := Alert$  and  $Event(u) := Alert$

## F. The sixth rule: R6

The rule *R6* is used to reset the nodes states of a maximal clique. It requires the presence of a node (noted  $u$ ) belonging to the maximal clique  $\{u, a, \dots, v\}$  and having “*Event*( $u$ ) = *Alert*”.

- If there is a neighbour node of  $u$  (noted  $h$ ) having the state  $A$ , then it resets the initial state. In fact, it can be a member of a maximal clique after resetting the states of nodes of the clique  $\{u, a, \dots, v\}$ .
- If a node (noted  $b$ ) which belongs to another maximal clique is the neighbour of the node  $u$  and can form a maximal clique (after destroying the clique  $\{u, a, \dots, v\}$ ) with the node  $u$  and other neighbours, then the node  $b$  takes the state *Alert*. Indeed, it must reset the states of the clique nodes by applying again the rule *R6*, to form a larger maximal clique. Formally, the rule *R6* is written as follows:

Precondition :

- $Clique(u) = \{u, a, \dots, v\}$  and  $card(Clique(u)) \geq 3$
- $Event(u) = Alert$  and  $(\forall k \cdot u \mapsto k \in g \text{ and } k \in Clique(u) \implies Event(k) = False)$

Relabeling :

- $\forall k \cdot k \in Clique(u) \implies State(k) := Init$
- $\forall k \cdot k \in Clique(u) \implies Clique(k) := \emptyset$
- $\forall k \cdot k \in Clique(u) \implies Event(k) := False$
- $\exists h \cdot \left( \begin{array}{l} u \mapsto h \in g \wedge State(h) = A \\ \wedge Clique(h) = \{h\} \end{array} \right) \implies State(h) := Init \wedge Clique(h) := \emptyset$
- $\exists b \cdot \left( \begin{array}{l} u \mapsto b \in g \wedge card(Clique(b)) \geq 3 \\ \wedge Clique(b) \neq Clique(u) \\ \wedge (\forall r \cdot r \in Clique(b) \implies u \mapsto r \in g) \end{array} \right) \implies Event(b) := Alert$

The propose rules of the algorithm CMCDN are applied asynchronously and non-deterministically until no rule is applicable. This means that many different runs are possible. In the final configuration, every node ( $x$ ) has  $State(x) \in \{C, I, A\}$ ,  $Clique(x) \neq \emptyset$  and  $Event(x) = False$ :

- If  $x$  is an isolated node, it will have  $State(x) = A$  and  $Clique(x) = \{x\}$ .
- If  $x$  belongs to a maximal clique, it will have  $State(x) \in \{C, I\}$  and  $card(Clique(x)) \geq 3$ .

## III. FORMAL SPECIFICATION OF THE ALGORITHM CMCDN USING THE EVENT-B METHOD

We introduce in this section our Event-B formal model of the algorithm CMCDN. It is based on four abstraction levels as shown in Fig. 1. The first machine **M0** abstractly specifies the goal of the proposed algorithm which aims to compute maximal cliques problem in dynamic networks. It utilizes properties defined in the context “Graph”. The second machine **M1** refines M0. It contains some events which specify how nodes are making a choice to detect maximal cliques. Moreover, it globally specifies the consequences of deleting an edge. The machine **M2** refines M1. It provides more details to detect the set of nodes of each maximal clique and maintain

the set of the detected cliques when deleting an edge. Finally, the machine **M3** refines M2 and sees the context “Labels”. It includes a set of events corresponding to the six relabeling rules.

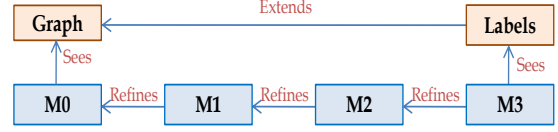


Fig. 1: The refinement strategy of the algorithm CMCDN

### A. Formal specification of the contexts

1) *The context “Graph”* : This context specifies static properties of the network. Formally, a graph is modeled by a set of nodes called  $V$ . In our work, we suppose that a dynamic graph is composed of stable nodes. So, we define  $V$  in the context as an abstract set. We specify that the number of nodes in the network is finite ( $axm1 : finite(V)$ ). Also, we introduce a constant, called “ $tn$ ”, that represents the final system date ( $axm2 : tn \in \mathbb{N}1$ ).

2) *The context “Labels”* : This context extends the context “Graph” by adding the labels of nodes to our model. Indeed, we introduce two sets called “*labels*” and “*event\_labels*”. “*labels*” contains the labels : *Init*, *C*, *I*, *W* and *A* ( $axm1$  and  $axm2$ ). The label allows nodes to perform an elementary step of computation according to some relabeling rules.

$axm1 : partition(labels, \{Init\}, \{C\}, \{I\}, \{W\}, \{A\})$   
“*event\_labels*” includes the labels *Alert* and *False*.

$axm2 : partition(event\_labels, Alert, False)$

### B. Formal specification of the machines

1) *The first level* : The first machine M0 specifies the goal of the distributed algorithm, without describing the process of computing the solution (see Listing 1).

Listing 1: Machine M0 invariants

```

inv1 :  $g \in 0 \dots t \rightarrow \mathbb{P}(V \times V)$ 
inv2 :  $\forall ti \cdot ti \in dom(g) \Rightarrow g(ti) = (g(ti))^{-1}$ 
inv3 :  $\forall ti \cdot ti \in dom(g) \Rightarrow (V \triangleleft id) \cap (g(ti)) = \emptyset$ 
inv4 :  $t \in \mathbb{N} \wedge t \leq tn$ 
inv5 :  $change \in \{0, 1\}$ 
inv6 :  $t \geq 1 \wedge change = 0 \Rightarrow g(t) = g(t-1)$ 
inv7 :  $cliques = \{V1, g1 \cdot V1 \subseteq V \wedge finite(V1) \wedge card(V1) \geq 3 \wedge g1 \subseteq g(t) \wedge g1 = (V1 \times V1) \setminus (V \triangleleft id) \wedge g1 = g1^{-1} | V1\}$ 
inv8 :  $combination \subseteq \mathbb{P}(cliques)$ 
inv9 :  $combination = \{X, x1, x2 \cdot X \subseteq cliques \wedge x1 \in X \wedge x2 \in X \wedge x1 \neq x2 \wedge x1 \cap x2 = \emptyset \wedge (\forall Y, a \cdot Y \in X \wedge a \in cliques \Rightarrow Y \cap a = \emptyset) | X\}$ 
inv10 :  $solution \in 0 \dots t \rightarrow \mathbb{P}(V)$ 
inv11 :  $final \in \{0, 1\}$ 
  
```

A network can be modeled as a simple and undirected graph  $g$ . It is defined as a function to be the graph at the current date  $t$  ( $inv1$  and  $inv4$ ). An undirected graph means that there is no distinction between two nodes associated with each edge ( $inv2$ ). A graph is simple if it has zero or one edge between any two nodes and no edge starts and ends at the same node ( $inv3$ ). Moreover, we introduce a variable called “*change*” ( $inv5$ ). If one topological event has been produced, “*change*” is equal to “1”, otherwise “*change*” is equal to “0”. In the invariant  $inv6$ , we indicate that if the date  $t$  is strictly greater than “0” and  $change$  is equal to “0”, the graph does not undergo any topological event. We define “*cliques*” as the set of all possible cliques in the graph  $g$  ( $inv7$ ). Any pair of nodes can never form a clique, then all cliques contain

at least three nodes. We add *inv8* and *inv9* to specify all possible combinations of maximal and disjoint cliques in the graph *g* called “*combination*”. Also, we introduce the variable “*solution*” which contains the result of the algorithm at the date *t* (*inv10*). The variable “*final*” allows to check if the *Oneshot* event (will be explained later) has been triggered. The machine M0 includes three events :

- **Event “Oneshot”** (see Listing 2) : It specifies the result of the algorithm in one step. The analogy of someone closing and opening their eyes. In a given graph, there are many combinations of maximal cliques. The event *Oneshot* assigns in a non-deterministic way an element from “*combination*” to the variable “*solution*” at the date *t*.

Listing 2: Event *Oneshot*, in M0

EVENT	<i>Oneshot</i>
ANY	<i>c</i>
WHERE	
	<i>grd1</i> : $solution(t) = \emptyset$
	<i>grd2</i> : $c \in combination$
	<i>grd3</i> : $final = 0 \wedge t \neq tn$
THEN	
	<i>act1</i> : $solution(t) := c$
	<i>act2</i> : $final := 1$
END	

- **Event “Remove\_Edge”** (see Listing 3) : An edge has been removed at the date *t* if it is present at the date “*t*” (*grd1*) and “*t*” is different from the final date “*tn*” (*grd2*). Then, we update the graph *g(t)* and the sets “*cliques*” and “*combination*”. Also, the variable “*change*” takes the value “1” (*act2*).

Listing 3: Event *Remove\_Edge*, in M0

EVENT	<i>Remove_Edge</i>
ANY	<i>x, y</i>
WHERE	
	<i>grd1</i> : $x \mapsto y \in g(t)$
	<i>grd2</i> : $final = 0 \wedge t \neq tn$
THEN	
	<i>act1</i> : $g(t) := g(t) \setminus \{x \mapsto y, y \mapsto x\}$
	<i>act2</i> : $change := 1$
	<i>act3</i> : $cliques, combination :  cliques'  = \{V1, g1 \cdot V1 \subseteq V \wedge finite(V1) \wedge card(V1) \geq 3 \wedge g1 \subseteq (g(t) \setminus \{x \mapsto y, y \mapsto x\}) \wedge g1 = (V1 \times V1) \setminus (V \triangleleft id) \wedge g1 = g1^{-1}   V1\}$
	$\wedge combination' = \{X, x1, x2 \cdot X \subseteq cliques' \wedge x1 \in X \wedge x2 \in X \wedge x1 \neq x2 \wedge x1 \cap x2 = \emptyset \wedge (\forall Y, a \cdot Y \in X \wedge a \in cliques' \Rightarrow Y \cap a = \emptyset)   X\}$
END	

- **Event “Increment\_Time”** (see Listing 4) : This event can be activated if the current date *t* is strictly lower than the final system date *tn* (*grd1*), the result of the algorithm is verified (*grd3*) and at least one topological event is performed in the network (*grd2*). In the action component of this event, we increment the time to *t + 1* (*act1*) and we set the graph at the date *t + 1* to the graph *g(t)* (*act2*). In addition, we reset the variables *change*, *final* and *solution*. Therefore, we have no topological change at the date *t + 1*.

Listing 4: Event *Increment\_Time*, in M0

EVENT	<i>Increment_Time</i>
WHERE	
	<i>grd1</i> : $t < tn$
	<i>grd2</i> : $change = 1$
	<i>grd3</i> : $final = 1$
THEN	
	<i>act1</i> : $t := t + 1$
	<i>act2</i> : $g(t + 1) := g(t)$
	<i>act3</i> : $solution(t + 1) := \emptyset$
	<i>act4</i> : $change := 0$
	<i>act5</i> : $final := 0$
END	

2) *The second level* : In the machine M1, we start by introducing details to calculate globally the maximal cliques. To do so, we add some variables to the invariant component as shown in Listing 5:

- “*cliques\_in*” contains the nodes which belong to the detected cliques (*inv1*).
  - “*cliques\_out*” defines the set of nodes which do not belong to these cliques (*inv2*).
  - “*cliques\_new*” is the set of nodes of the detected maximal and disjoint cliques (*inv5* and *inv6*).
  - “*adjacents*” gives the set of adjacent nodes to each node at the current date *t* (*inv9* and *inv10*).
  - “*nodes\_alert*” contains the nodes of the clique edge that has undergone a topological event (*inv11*).
- Initially, *cliques\_out* contains all the graph nodes “*V*”, whereas *cliques\_in* and *cliques\_new* are empty.

Listing 5: Machine M1 invariants

<i>inv1</i>	: $cliques\_in \subseteq V$
<i>inv2</i>	: $cliques\_out \subseteq V$
<i>inv3</i>	: $cliques\_in \cap cliques\_out = \emptyset$
<i>inv4</i>	: $cliques\_in \cup cliques\_out = V$
<i>inv5</i>	: $cliques\_new \subseteq \mathbb{P}(cliques\_in)$
<i>inv6</i>	: $\forall a, b \cdot a \in cliques\_new \wedge b \in cliques\_new \wedge a \neq b \Rightarrow a \cap b = \emptyset$
<i>inv7</i>	: $\forall x \cdot x \in cliques\_new \wedge finite(x) \Rightarrow card(x) \geq 3$
<i>inv8</i>	: $\forall x \cdot x \in cliques\_new \Rightarrow x \in cliques$
<i>inv9</i>	: $adjacents \in V \times (0..t) \rightarrow \mathbb{P}(V)$
<i>inv10</i>	: $\forall ti, x \cdot ti \in (0..t) \wedge x \in V \Rightarrow adjacents(x \mapsto ti) = \{y \cdot x \mapsto y \in g(ti) \vee y \mapsto x \in g(ti)   y\}$
<i>inv11</i>	: $nodes\_alert \subseteq V$
<i>inv12</i>	: $\forall x \cdot x \in cliques\_out \Rightarrow x \notin nodes\_alert$

The definition of these variables requires the addition of new properties:

- (*inv3*) The nodes of *cliques\_in* are different from those of *cliques\_out*.
- (*inv4*) The total of these nodes is equal to the set of nodes *V*.
- (*inv7*) All the detected cliques contain at least three nodes.
- (*inv8*) The set of the detected cliques is a subset of “*cliques*”.
- (*inv11*) The nodes of *cliques\_out* can not belong to the set *nodes\_alert*.

At this level, we refine the events defined in M0 and we add some new events:

- **Event “Clique+”** : This event aims to compute maximal cliques in the graph *g* (see Listing 6). It can be activated if we have a ball “*B*” including at least three nodes (*grd3* and *grd6*). We note “*x*” the center of the ball *B* (*grd2* and *grd3*). All the nodes of *B* are connected (*grd4*) and belong to *cliques\_out* (*grd1*). Using the *grd5*, we express that the ball *B* can not be extended by one or more adjacent nodes. At every computation step, the nodes of the detected ball are eventually added to the set *cliques\_in* (*act2*) and removed from *cliques\_out* (*act1*). Moreover, we add the set of ball nodes to *cliques\_new* (*act3*).

Listing 6: Event *Clique+*, in M1

EVENT	<i>Clique+</i>
ANY	<i>B, x</i>
WHERE	
	<i>grd1</i> : $B \subseteq cliques\_out$
	<i>grd2</i> : $x \in B$
	<i>grd3</i> : $B \subseteq (adjacents(x \mapsto t) \cup \{x\})$
	<i>grd4</i> : $\forall y, z \cdot y \in B \wedge z \in B \wedge y \neq z \Rightarrow y \mapsto z \in g(t)$
	<i>grd5</i> : $(\forall r \cdot r \in cliques\_out \wedge r \in adjacents(x \mapsto t) \wedge \{r\} \times (B \setminus \{r\}) \subseteq g(t) \Rightarrow r \in B)$
	<i>grd6</i> : $card(B) \geq 3$
	<i>grd7</i> : $final = 0 \wedge t \neq tn$
THEN	
	<i>act1</i> : $cliques\_out := cliques\_out \setminus B$
	<i>act2</i> : $cliques\_in := cliques\_in \cup B$
	<i>act3</i> : $cliques\_new := cliques\_new \cup \{a \cdot a \in B   a\}$
END	

• **Event “Clique-”** : The purpose of this event is to detect a node (noted “ $x$ ”) which can not belong to any maximal clique. If this node belongs to a connected ball  $B1$ ,  $B1$  should contain less than three nodes. Also, all the neighbours of “ $x$ ”, which have not been affected to a maximal clique yet, have a ball including at most two nodes.

• **Event “Oneshot”** : This event refines the *Oneshot* presented in M0 to verify that the final value of *cliques\_new* represents the result of the algorithm (see Listing 7). To do so, we reinforce the guard component by specifying that all maximal cliques in the graph  $g$  have been detected (*grd2*). By means of the theorem *Th4*, we verify that the set *cliques\_new* represents the detected maximal and disjoint cliques. The abstract parameter “ $c$ ”, defined in M0, is replaced with a concrete value (*cliques\_new*) by means of a witness. A witness designates a simple equality predicate involving the abstract parameters.

Listing 7: Event *Oneshot*, in M1

<b>EVENT</b>	<i>Oneshot</i>
<b>REFINES</b>	<i>Oneshot</i>
<b>WHERE</b>	
<i>grd1</i>	: $solution(t) = \emptyset$
<i>grd2</i>	: $\forall y. y \subseteq g(t) \wedge y \notin cliques\_new \implies y \notin cliques$
<i>grd3</i>	: $final = 0 \wedge t \neq tn$
<i>Th4</i>	: $cliques\_new \in combination$
<i>grd4</i>	: $\forall x. x \in V \implies x \notin nodes\_alert$
<b>WITH</b>	$c : c = cliques\_new$
<b>THEN</b>	
<i>act1</i>	: $solution(t) := cliques\_new$
<b>END</b>	

At this level, we refine the event *Remove\_Edge* in two events:

• **Event “Remove\_Graph\_Edge”** : It specifies the case of deletion of a graph edge. To do so, we add a condition (*grd4*) to indicate that the edge  $x \mapsto y$  does not belong to a maximal clique (noted  $n$ ):

$$\forall n. n \in cliques\_new \implies \neg(x \in n \wedge y \in n)$$

• **Event “Remove\_Clique\_Edge”** : This event, depicted in Listing 8, specifies the preliminary influence of the deletion of an edge belonging to a maximal clique. Formally, we reinforce the guard component to express that the edge  $x \mapsto y$  belongs to a maximal clique noted “ $k$ ” (*grd3*). Also, no topological event affects the clique (*grd4*). In the clause “THEN”, we introduce two actions to update the sets of adjacent nodes of  $x$  and  $y$  (*act4*). In addition, we add  $x$  and  $y$  to the set *nodes\_alert* (*act5*).

Listing 8: Event *Remove\_Clique\_Edge*, in M1

<b>EVENT</b>	<i>Remove_Clique_Edge</i>
<b>REFINES</b>	<i>Remove_Edge</i>
<b>ANY</b>	$x, y, k$
<b>WHERE</b>	
<i>grd3</i>	: $k \in cliques\_new \wedge x \in k \wedge y \in k$
<i>grd4</i>	: $x \notin nodes\_alert \wedge y \notin nodes\_alert$
<b>THEN</b>	
<i>act4</i>	: $adjacents := adjacents \Leftarrow \{(x \mapsto t) \mapsto adjacents(x \mapsto t) \setminus \{y\}, (y \mapsto t) \mapsto adjacents(y \mapsto t) \setminus \{x\}\}$
<i>act5</i>	: $nodes\_alert := nodes\_alert \cup \{x, y\}$
<b>END</b>	

To specify the different situations of reinitialization of the neighbours (of the concerned maximal clique) of a node (noted  $u$ ) belonging to the set *nodes\_alert*, we distinguish four cases. Because of space limitation, we only detail the informal description of these cases:

**Case 1:** It specifies the simplest case which consists in resetting the nodes states of  $u$  and its neighbours of the maximal clique. This case is specified by a new event called *Initialize1*.

**Case 2:** If there is a neighbour node of  $u$  (noted  $h$ ) which can not belong to any maximal clique, we also reset the state of  $h$ . Indeed, the node  $h$  can be a member of a maximal clique after resetting the nodes states of the clique containing  $u$ . This situation is specified using a new event *Initialize2*.

**Case 3:** If a neighbour node of  $u$  (noted  $b$ ) which belongs to another detected clique can form a maximal clique with the node  $b$  and other neighbours of the clique, then we introduce  $b$  to the set *nodes\_alert*. We specify this case by means of an event called *Initialize3*.

**Case 4:** The presence of the cases (2) and (3) requires the application of an event called *Initialize4*.

3) *The third level* : The refinement of M1 called M2 introduces more details about the algorithm CMCDN. In fact, we introduce the variable “*Clique*” as a function which assigns to each node the set of nodes of its maximal clique (*inv1* :  $Clique \in V \times (0..t) \rightarrow \mathbb{P}(V)$ ). Initially, the *Clique* of each node is empty. To link the states between the machines M1 and M2, we define some gluing invariants:

◦ Each node of *cliques\_in* belongs to a detected maximal clique.

(*inv1*)  $\forall x, ti. x \in cliques\_in \implies Clique(x \mapsto ti) \neq \emptyset \wedge x \in Clique(x \mapsto ti)$

◦ Each node of *cliques\_out* has not computed its maximal clique yet.

(*inv3*)  $\forall x, ti. x \in cliques\_out \implies Clique(x \mapsto ti) = \emptyset$

◦ Each node which has not been assigned to a maximal clique yet belongs to the set *out\_cliques*.

(*inv4*)  $\forall x, ti. Clique(x \mapsto ti) = \emptyset \implies x \in cliques\_out$

◦ A node “ $x$ ”, which does not belong to a maximal clique, is not part of the set *nodes\_alert*.

(*inv5*)  $\forall x, ti. Clique(x \mapsto ti) = \emptyset \vee Clique(x \mapsto ti) = \{x\} \implies x \notin nodes\_alert$

At this refinement level, the events of the previous level still exist but they become more concrete. We restrict to detail in what follows some events :

• **Event “Remove\_Graph\_Edge”** : We refine the event *Remove\_Graph\_Edge* by reinforcing the guard component. In fact, we replace the guard *grd4* using the variable *Clique* that the removed edge does not belong to any maximal clique: *grd4* :  $\neg(Clique(x \mapsto t) = Clique(y \mapsto t) \wedge card(Clique(x \mapsto t)) \geq 3)$

• **Event “Clique+”**: The goal of this event is to detect maximal and disjoint cliques and assign to each node its corresponding clique (see Listing 9).

Listing 9: Event *Clique+*, in M2

<b>EVENT</b>	<i>Clique+</i>
<b>REFINES</b>	<i>Clique+</i>
<b>ANY</b>	$B, x$
<b>WHERE</b>	
<i>grd1</i>	: $Clique(x \mapsto t) = \emptyset \wedge B \cap \{k. Clique(k \mapsto t) \neq \emptyset   k\} = \emptyset$
<i>grd5</i>	: $\forall r. Clique(r \mapsto t) = \emptyset \wedge r \in adjacents(x \mapsto t) \wedge \{r\} \times (B \setminus \{r\}) \subseteq g(t) \implies r \in B$
<i>grd6</i>	: $card(inter(\{a. a \in B   adjacents(a \mapsto t) \cup \{a\}\} \setminus \{k. Clique(k \mapsto t) \neq \emptyset   k\})) \geq 3 \wedge finite(inter(\{a. a \in B   adjacents(a \mapsto t) \cup \{a\}\} \setminus \{k. Clique(k \mapsto t) \neq \emptyset   k\}))$
<i>grd7</i>	: $final = 0 \wedge t \neq tn$
<b>THEN</b>	
<i>act1</i>	: $Clique := Clique \Leftarrow \{a. a \in B   (a \mapsto t) \mapsto B\}$
<b>END</b>	

In fact, we reinforce the guard component by using the new variable *Clique*. The guard (*grd1*) specifies that the center  $x$

of the ball  $B$  and its neighbours from  $B$  do not belong to any detected clique. The *grd5* states that  $B$  can not be extended by other nodes which have not been assigned to maximal cliques yet. We indicate in the guard *grd6* that the intersection of all the elements of  $N(a)$ , which do not belong to maximal cliques, contains at least three nodes. We note “ $a$ ” as each node of the ball  $B$ . In the action component, we set the maximal clique of each node of the ball to “ $B$ ” (*act1*).

• **Event “Oneshot”:** This event refines the “Oneshot” presented in the machine M1. It uses the concrete variable *Clique* to check that each node of the graph has computed the maximal clique to which it belongs. In fact, the result of the algorithm represents the set containing the maximal clique of each node  $x$  (if  $Clique(x \mapsto t) \neq \{x\}$ ).

4) *The fourth level* : Once the machine of the third level has been specified and proven, it can be refined for describing the local label modification and encoding the relabeling rules proposed in Section II. In order to reach this goal, we introduce a new variable “*State*” (*inv1* :  $State \in V \times (0..t) \rightarrow labels$ ) which assigns to each node a label from the set “*labels*” that encodes the state of a process. Initially, all the nodes are labeled “*Init*”. The addition of the variable “*State*” involves adding new properties which link the abstract state variables to the concrete ones. We have formalized these properties in the form of Event-B invariants:

◦ A node  $x$ , which has  $Clique(x \mapsto ti)$  not empty, belongs to a maximal clique or it is an isolated node at the date  $ti$ .

(*inv2*)  $\forall x, ti. Clique(x \mapsto ti) \neq \emptyset \Rightarrow$

$x \in State^{-1}[\{C, I, A\}]$

◦ A node which has not been assigned to a maximal clique yet is in the initial or the waiting state.

(*inv3*)  $\forall x, ti. Clique(x \mapsto ti) = \emptyset \Rightarrow$

$x \in State^{-1}[\{Init, W\}]$

◦ If a node is labeled  $C$ , its maximal clique contains itself and a set of its neighbours.

(*inv4*)  $\forall x, ti. State(x \mapsto ti) = C \Rightarrow Clique(x \mapsto ti) \subseteq \{x\} \cup adjacents(x \mapsto ti) \wedge x \in Clique(x \mapsto ti)$

◦ Each maximal clique contains one center node labeled  $C$  and the other nodes are labeled  $I$ .

(*inv5*)  $\forall x, y, ti. y \in Clique(x \mapsto ti) \setminus \{x\} \wedge State(x \mapsto ti) = C \Rightarrow State(y \mapsto ti) = I \wedge Clique(y \mapsto ti) = Clique(x \mapsto ti)$

◦ If a node  $y$  is labeled  $I$ , it has a neighbouring node which belongs to the same maximal clique and it is the center of this clique.

(*inv6*)  $\forall y, ti. State(y \mapsto ti) = I \Rightarrow (\exists x. x \in adjacents(y \mapsto ti) \wedge State(x \mapsto ti) = C \wedge Clique(y \mapsto ti) = Clique(x \mapsto ti))$

◦ An isolated node does not belong to any maximal clique, then its maximal clique is the identity.

(*inv7*)  $\forall x, ti. State(x \mapsto ti) = A \Rightarrow Clique(x \mapsto ti) = \{x\}$

◦ Each node  $x$  labeled  $C$  or  $I$  belongs to a maximal clique, then the set of elements of its clique is not empty.

(*inv8*)  $\forall x, ti. State(x \mapsto ti) \in \{C, I\} \Rightarrow Clique(x \mapsto ti) \neq \emptyset$

◦ Each node in the initial or the waiting state has not been assigned to a maximal clique yet.

(*inv9*)  $\forall x, ti. State(x \mapsto ti) \in \{W, Init\} \Rightarrow Clique(x \mapsto ti) = \emptyset$

At this level, we refine the event “Oneshot” and we specify the six relabeling rules of our algorithm:

◦ The event “*Rule1*” refines the event “*Clique +*” defined

in M2 to specify the rule *R1*.

◦ We introduce a new event called “*Rule2*” to specify the rule *R2*.

◦ The event “*Rule3*” specifies the rule *R3* and refines the event *Clique*– of the machine M2.

◦ The event “*Rule4*” refines the event *Remove\_Graph\_Edge* to express the modification made when deleting an edge of the graph.

◦ The event “*Rule5*” refines the event *Remove\_Clique\_Edge* to specify the preliminary influence of the removal of an edge belonging to a maximal clique.

◦ The events (*Rule6*, *Rule6'*, *Rule6''*, *Rule6'''*) refine respectively the events (*Initialize1*, ..., *Initialize4*) to express the different cases of the rule *R6*.

◦ The event “*Oneshot*” verifies that, at the end of the algorithm execution, no node is in the initial (labeled “*Init*”) or the waiting state (labeled “*W*”).

### C. Proof statistics

To prove the correctness of our formal model, a number of proof obligations (POs) generated by the Rodin platform should be discharged. The algorithm development results in 405 POs, in which 226 (56%) POs are proved automatically and 179 (44%) are proved interactively using the RODIN prover. An Event-B model is correct when all POs have been discharged. Formal definitions of all POs are given in [1].

## IV. CONCLUSION

We have presented in this paper a new distributed algorithm for enumerating maximal cliques in dynamic networks. Our algorithm combines local computations model and refinement to prove its correctness. The proposed algorithm is based on six relabeling rules which allow to detect maximal cliques and react correctly in case of edge deletion.

## REFERENCES

- [1] J.-R. Abrial, *Modeling in Event-B - System and Software Engineering*. Cambridge University Press, 2010.
- [2] A. Conte, R. D. Virgilio, A. Maccioni, M. Patrignani, and R. Torlone, “Finding all maximal cliques in very large social networks,” in *the International conference Extending Database Technology (EDBT)*, 2016, pp. 173–184.
- [3] A. Ferreira, “On models and algorithms for dynamic communication networks: The case for evolving graphs,” in *the conference ALGOTEL*, 2002.
- [4] G. T. Leavens, J.-R. Abrial, D. Batory, M. Butler, A. Coglio, K. Fisler, E. Hehner, C. Jones, D. Miller, S. Peyton-Jones, M. Sitaraman, D. R. Smith, and A. Stump, “Roadmap for enhanced languages and methods to aid verification,” in *the International Conference Generative Programming and Component Engineering (GPCE)*. ACM, 2006, pp. 221–236.
- [5] I. Litovsky, Y. Métivier, and E. Sopena, “Handbook of graph grammars and computing by graph transformation.” World Scientific, 1999, ch. Graph Relabelling Systems and Distributed Algorithms, pp. 1–56.
- [6] C. Luo, J. Yu, D. Yu, and X. Cheng, “Distributed algorithms for maximum clique in wireless networks,” in *the International conference Mobile Ad-hoc and Sensor Networks (MSN)*. IEEE, 2015, pp. 222–226.
- [7] N. A. Lynch, *Distributed algorithms*. Elsevier, 1996.
- [8] Y. Xu, J. Cheng, and A. W.-C. Fu, “Distributed maximal clique computation and management,” *IEEE Transactions on Services Computing*, vol. 9, no. 1, pp. 110–122, 2016.



# DCCD: An Efficient and Scalable Distributed Code Clone Detection Technique for Big Code

Junaid Akram\*(Member, IEEE), Zhendong Shi\*, Majid Mumtaz\* and Luo Ping\*

\*State Key Laboratory of Information Security, School of Software Engineering, Tsinghua University China.

Email: [znd15, szd15, maji16]@mails.tsinghua.edu.cn

Email: luop@mail.tsinghua.edu.cn

**Abstract**—Code clone detection is a very hot topic in the field of software maintenance, reuseability and security. There is still a lack of techniques to detect near-miss clones at different level of granularities, especially in big code. This paper presents Distributed Code Clone Detection (DCCD) technique, which detects clones from big code bases based on feature extraction. We performed preprocessing, indexing and clone detection for almost 27 TB of source code (324 billion LOC), DCCD is quite faster and efficient as compared to existing distributed indexing and clone detection techniques, i.e. 36 times faster than Benjamin technique, which is 86 times faster than CCFinder. These two techniques are also distributed and just detect Type-1 and Type-2 clones, but our technique DCCD even detects Type-3 clones, efficiently. Our approach is faster, flexible, scalable and provides 87% accurate results with authenticity, ease of accessibility, upgradeability and maintainability.

**keyword** Clone detection, Software maintenance, Software reuse, Big code, Similarity/Plagiarism detection

## I. INTRODUCTION

When a programmer copies code fragments and tries to reuse them by pasting in other code sections with or without making minor modifications, this type of code reuse approach is called code cloning, and the pasted code fragment called a clone of the original. It is a very adapting process in software development activities. However, during detection of clones, it is hard to say that which code fragment is original and which code fragment is copied. Code clones bring troubles in software security and maintenance and they lead to bug propagation. Roy describes that a very significant range (7% - 23%) of code is cloned in large scale systems [1]. Code clone detection techniques are very helpful for code maintainability, code plagiarism detection [2] [3], code verification, copyright detection, security flaws detection, detection of bugs and malicious software detection.

Our developed platform provides an index-based hybrid solution (semantic approach) by combining different clone detection techniques for large scale systems which is distributed, scalable and incrementable. It detects Type-1, Type-2, Type-3 code clones in real time environment on the basis of big code. Our system is based on Hadoop environment, which extends a practical applicability of index based clone detection for very large code bases and it demonstrates the response time sufficiently fast. Our technique has been developed and tested to detect code clones in 15 different programming languages i.e. Java, JavaScript, C, C++, C#, Xml, Python, Php, Sql, Vb,

Cobol, Text, Ruby, Ada, Matlab. In this paper, we will discuss and display results of 3 programming languages on large scale level, which are Java, JavaScript and C/C++. The downloaded source code was about 40 TB (20 TB C & C++, 10 TB Java, 10 TB JavaScript), but the experiment of preprocessing, indexing and feature extraction was performed on almost 27 TB (C & C++: 16 TB, JavaScript: 11 TB, Java: 1 TB) source code. The main purpose of our research was to deal with the big code on a large-scale level and detect near-miss clones accurately, which is not only challenging but computationally expensive. We preprocess source code using different mining techniques/filters, extracting main features and store the index information into a database for further inspection process of clone detection. Preprocessing, normalization, feature extraction and indexing process were performed in a pipeline, so index creation is actually fully depended on the output of preprocessing of source code. The top level view of whole system has been shown in Fig 1. Index based [4] and CCFinder [5] are two token-based distributed clone detection techniques but they just only detect Type-1 & Type-2 clones, and even not support big code indexing and detection process. Our approach detects Type-1, Type-2, Type-3 clones with high accuracy rate, meanwhile it's incremental, scalable and fast.

## II. DCCD ARCHITECTURE

In this section, we briefly describe all the steps and phases of the proposed DCCD clone detection architecture. The proposed work is the hybrid technique of clone detection for large-scale systems, in which we have applied many screening filters to retrieve exact clone files from big code bases. Clone detection process comprises of many phases, where each next phase depends on the previous phase and builds on the outcome of the previous phase. There are four main phases in our clone detection approach as shown in Fig 1. Fig 2 shows the technical view of preprocessing and normalization process. The left side of Fig 2, correspondence between an original code fragment and the preprocessed and normalized code fragment is visualized. On the right side of Fig 2, the indexing entities and chunk properties have been shown, where MD5 are the hash values of the prefix and suffix sequence of statements, of the source code file. The reason we used a sequence of statement during indexing instead of individual statement is that the sequence of statements are more unique and identical. FNV values are the 10 hash values per division

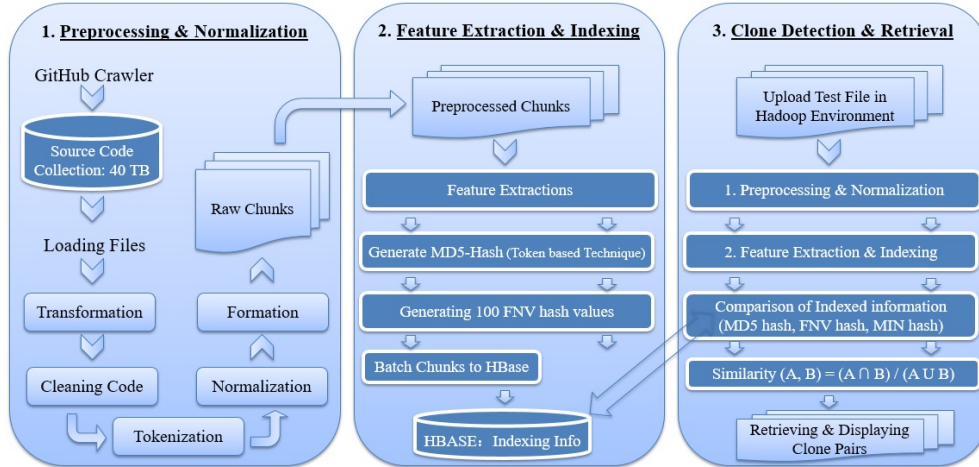


Fig. 1: DCCD System top level view

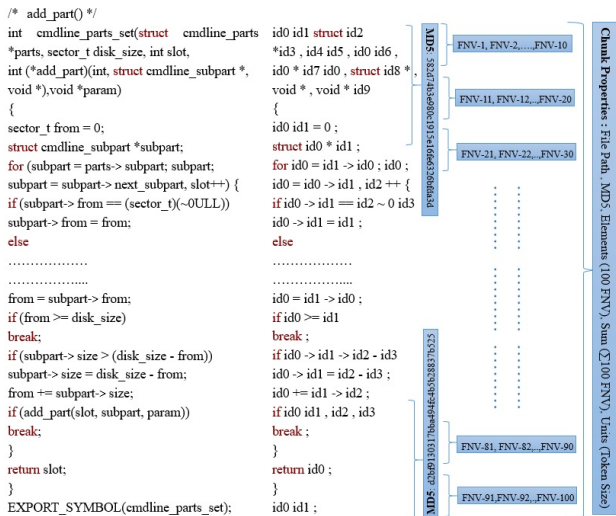


Fig. 2: The original source code (left), its normalization (middle) and indexing info (right)

of a file. The Chunk properties are the feature, which we use for comparison of two files.

### A. DCCD Preprocessing and Normalization

This is the first phase, which removes uninteresting and unwanted pieces of codes from source files. There are three major processes of this phase. (a) It reads source code files from disk and splits the code into tokens. (b) Remove all uninteresting pieces of code from the source code files. (c) Normalization is performed on these tokens. These tasks have been explained below in detail.

**Loading:** It reads the selected project files from the hard disk and load into RAM for further processing.

**Transformation:** In this task, we select the language type of project and consider the related source code files of that project and ignore all unwanted files.

**Tokenization:** After full transformation of source code into a

byte stream, we start tokenizing the code sequence. The main purpose of this process is to organize source code in tokenized form line by line. Then lexical analyzer uses each traversal at the same time to tokenize. Finally, the corresponding token sequences are generated and stored in memory.

**Normalization:** During normalization, we delete the unwanted tokens, comments, spaces, import libraries and the tokens which do not have any effect on the source code.

**Chunk Formation:** After token normalization, we initialize an empty entity set (chunk), each attribute in the entity set initialized as null. We use this formation to make chunk ready to store Elements (feature), meanwhile, we identify the size of the chunk. After performing this task, we able to have preprocessed code, in which identifiers replaced with ID plus numbers (starts from 0); a string replaced by an empty string; fixed string with some identified characters; floating types with 0, and boolean values into true as shown in Fig 2.

### B. Feature Extraction

This phase is the core part of our clone detection technique because these extracted features help us to detect code clones. Feature extraction basically involves reducing the amount of code to describe a big code base by extracting features from it. This phase consists of many steps, some of them performed in parallel and some of them performed in pipeline. The left side of Fig 3 shows the flow of each process into 5 steps, the right side displays the HBase entities, which we extract and store into HBase, i.e. Row\_key, Origin\_id, Elements, Units and All.

**Step 1:** This step actually gets the source code project files from code repository in pipeline. The source code of these files further converted into bytecode by using preprocessing and normalization phase.

**Step 2:** In this step, we use MD5 hashing algorithm as a feature extraction from the source file to differentiate the uniqueness of every file. MD5 encrypt the prefix 15 token statements and suffix 15 token statements of every source code

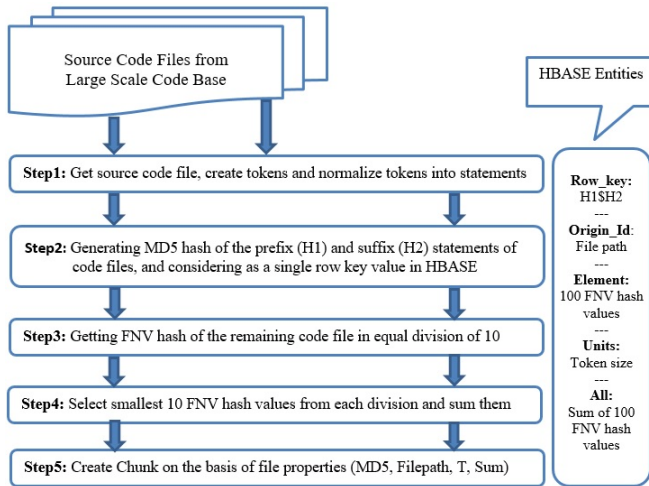


Fig. 3: Feature extraction from source code files

file. The reason to use 15 tokens of prefix and suffix is to define a unique signature set of every source file.

**Step 3:** In this step, we extract another feature from the source code files by using FNV (Fowler-Noll-Vo)<sup>1</sup> hashing algorithm. The reason we use FNV is that it quickly hashes the large amounts of data with a small conflict. The FNV hash has been generated for every token of the source file.

**Step 4:** In this case, we collect 100 hash values in total then add all of these values into one single value (long integer). These FNV hash values actually represent the overall characteristics of a file.

**Step 5:** The final step of feature extraction is to creating chunks on the basis of extracted features in previous steps.

### C. Feature-Based Index Creation

Indexing allow to find all clones against a single file or for an entire system. Meanwhile it allows to update index info, when files are removed, modified, or added. Indexing based on feature extraction is the main data structure in our code clone detection technique. The index creation process is very flexible, fast, accurate, easy to maintain and upgradeable. We can update, delete or edit index information from HBase of any file at any time. Meanwhile, we can keep track and retrieve the index information of any file in less than a microsecond. The indexing data consists of following a list of labels, which describes the entities of HBase.

**Row\_key:** It is MD5 hash value of the prefix (H1) and suffix (H2) statements of the source code files.

**Origin\_id:** It is the file path or location of a file.

**Element:** It is 100 FNV hash values of a file.

**Units:** It is the total size of tokens inside the concerned file.

**All:** It is the sum of 100 FNV hash values.

To build an index of big code, we used Hadoop distributed framework. There are 7 systems in the cluster, one Master (Intel i7, 32GB) and six Slaves (Intel i5, 16GB). To perform experiment, we built an index of almost 27 TB of source

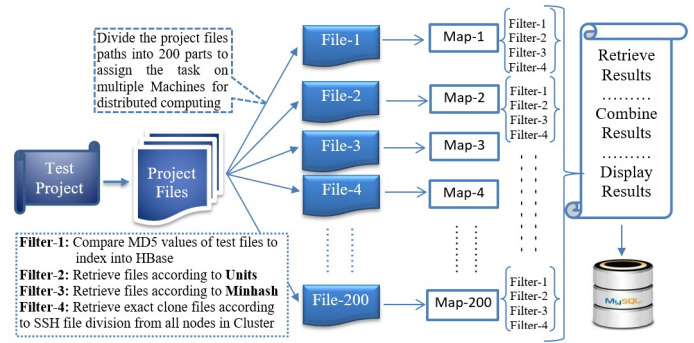


Fig. 4: MapReduce view for clone retrievals

code (16 TB C/C++, 10 TB JavaScript and 1TB of Java). This collection of source code for indexing was consisting of 1,039,260 projects, 885 million files and 324 billion of LOC.

### D. Code Clone Detection and Retrieval

This is the final phase, in which we retrieve and display the similarity between systems at different level of granularities, i.e. method level, chunk level, file level and project level. During detection, we filter, extract and retrieve all cloning objects, which meets the defined cloning filter conditions (Filter-1,2,3,4) as shown in Fig 4. The mapper (map reduce) retrieves all code clones and calculates the fraction of all files in the index, which at least contained one clone. We extract all clones against every single file of the test project.

In the first filter, the detector detects all the cloning files from big code repository by comparing indexes, which have the same `Row_key` (M1 hash \$ M2 hash) values, where M1 is the hash value of prefix token statements and M2 is the hash value of suffix token statements. Then, we apply another filter to get more abstract cloning files from large scale code. In this process, we compare the sum of 100 FNV hash values (All) in the index database, which were stored in HBase during the building of index of big code. We continue applying filters to get abstract scale code base from big code. In next filter, we compare the 100 FNV hash values (Elements) of the test file in the index and retrieve the resulted clone files. This Element entity of index consists of the 100 FNV hash values. Finally, after getting the abstract cloning files, we further evaluate them for *exact clones* by using Minhash algorithm. According to the idea of Minhash algorithm, the similarity can be calculated by the following formula.  $Similarity(A, B) = (A \cap B / A \cup B)$ . Where  $A \cap B$  is the number of code fragments, which are same in file A and in file B,  $A \cup B$  represents the total number of different code fragments in file A and in file B. In last step of detection and retrieval of code clone files. The detector uses SSH (Secure Shell) protocol to send, *to be tested files* in distributed environment to every attached node in the cluster. These files divided into chunks, generating the md5 hash values of defined chunks size (adjustable) and perform the further comparisons to detect exact clones. After getting same code fragments the detector evaluates the results, combined them and display to the users.

<sup>1</sup><http://www.isthe.com/chongo/tech/comp/fnv/index.html>

TABLE I: Building Index Results

Language	Size	Time (hours)	No of Projects	No of Files (millions)	LOC (billions)	Query Response per file (sec)
C & C++	16 TB	135	548,150	493	207	1.30
Java	1 TB	6	17,822	14	4	0.75
JavaScript	10 TB	65	473,228	378	113	0.95

**Configurable Threshold Value:** The threshold value is fully configurable by the user. It is also possible to detect clone fragments at different level of granularities, i.e. chunk level, method level, file level and project level. We note that we found the precision and recall to be optimum at 70% threshold. If we set threshold value high than 70%, the retrieved results will be more of Type-3, but it will be very less effect on Type-1 and Type-2 clones.

### III. CASE STUDIES AND RESULTS

In this section, we summarize the implementation and results of our proposed technique (DCCD) for batch clone detection and distributed clone detection in big code. Here are different case studies and RQs (research questions) on collection of big code, indexing, response time, detection of clones and their results comparison. Here are the main RQs, we formulated in our study:

- RQ1: How is the speed of index creation and how much storage space does it occupy in memory?
- RQ2: In what semantics DCCD produce better results as compare to previous state-of-the-art methods?
- RQ3: Does DCCD support partial level and full level similarities detection?
- RQ4: Is the DCCD technique scalable and efficient as compared to other clone detection techniques?

#### A. Collection of Source Code

To perform the assessment test on DCCD, the big code was required for our source repository. We collected almost 40 TB (C & C++: 20TB, Java: 10 TB, JavaScript: 10 TB) of source code. Our big code collection was consisting on thousands of projects, millions of project files and billions of LOC.

#### B. RQ1: Indexing Speed and Storage

Hadoop distributed environment is used to build an index of large scale system in a fast and efficient way. There were 7 machines (1: Intel i7, 32GB and 6: Intel i5, 16GB) in Hadoop environment. But only 5 Machines used for performing pre-processing and building index simultaneously. Table I shows the indexing results of 27 TB source code. The total indexed information size is almost 3 TB of 27 TB source code, which is quite small in size as compare to Benjamin [4] technique (3 times of original system).

#### C. RQ2: DCCD vs State-of-the-art Methods

In a comparison of old preprocessing, indexing and clone detection techniques, our approach is much faster as shown in Table II. In Benjamin [4] technique, they used 100 Google machines to build 73 million LOC source code in 3 hours on

Intel Xeon processor with 3GB RAM, which was 86 times faster than CCFinder [5] technique. In our approach, we just spend about 45 minutes to build the same amount of source code (73 million LOC) using 5 distributed systems (Intel i5, 16GB). CCFinder processed 400 MLOC on 80 machines (Pentium-4:3Ghz, 1GB RAM) in 51 hours, in our case we processed 400 MLOC in almost 5 hours on a single machine (Intel i5, 16GB). CCFinder [5] and Benjamin [4] just detected Type-1 and Type-2 clones. The most important achievement is that while they only detect Type-1, Type-2 clones. We detect all three types of clones in less time with big code, even though it was very challenging.

TABLE II: Build Index Speed Comparison

Technique	Machines	LOC(Million)	Time(Hours)	Clone Types
CCFinder	80	400	51	T-1, T-2
Benjamin	10	73	3	T-1, T-2
DCCD	5	3300	6	T-1, T-2, T-3

#### D. Batch Clone Detection

This case study shows that our feature extraction approach in the case of batch clone detection produces very good results. We used two open source projects of C and Java language as test projects. In the comparison of our technique (DCCD) with others techniques, i.e. Suffix-tree and Benjamin, the execution time of our approach is quite fast as shown in Table III. For each of the testing systems, DCCD is fast and meanwhile, it detects 3 types of clones, which is not possible in other 2 techniques.

TABLE III: Clone Detection Execution Time Comparison

Techniques	Jabref	Linux-Kernel	Clone Types
Suffix-tree	7.3 sec	166 min 13 sec	T-1, T-2
Benjamin	6.7 sec	47 min 29 sec	T-1, T-2
DCCD	5.2 sec	20 min 40 sec	T-1, T-2, T-3

#### E. DCCD (Distributed Code Clone Detection)

For the scalability and performance evaluation of our platform & algorithm to a large code base. Clone detection was performed through MapReduce, which retrieves all clones and calculates clone coverage for all project files in the index. In addition, to evaluate the scalability for ultra large code scale systems, we selected some subject systems including, e.g. Linux 2.6.33, Harvey, Cinder, PostgreSQL, OpenCV and Arduino. The detection processed 16.5 MLOC of C /C++ code of 37,398 files. In our experiment, we applied different filters of code clone detection method on every test project, the graphical representation have been shown in Fig 5.

TABLE IV: Detection Results Against their Files, LOC, Time and Clone Types

Test Projects	Total Files	LOC	Detection Time	Type-1 Clones	Type-2 Clones	Type-3 Clones
Linux 2.6.33	25,717	11,267,973	3 hours 20 min	1867	1443	3031
Harvey	3,761	1,307,197	22 min 37 sec	1460	546	917
Cinder	2,955	1,180,935	14 min 25 sec	1127	157	307
PostgreSQL	1,906	1,332,103	17 min 15 sec	889	356	373
OpenCV	2,379	1,141,501	13 min 48 sec	273	63	88
Arduino	680	277,710	9 min 33 sec	62	25	69

TABLE V: Retrieved Results Against Each Filter

C&C++ repository info: (Code: 16 TB, Files: 493 Million, LOC:207 billion)

Test Projects	Filter-1	Filter-2	Filter-3	Filter-4
Linux 2.6.33	99,031	95,471	78,570	66,034
Harvey	82,013	51,492	35,301	31,101
Cinder	57,788	37,031	25,701	13,876
PostgreSQL	40,371	27,096	19,811	14,503
OpenCV	17,359	10,782	8,703	5,609
Arduino	10,093	7,182	3,935	1,808

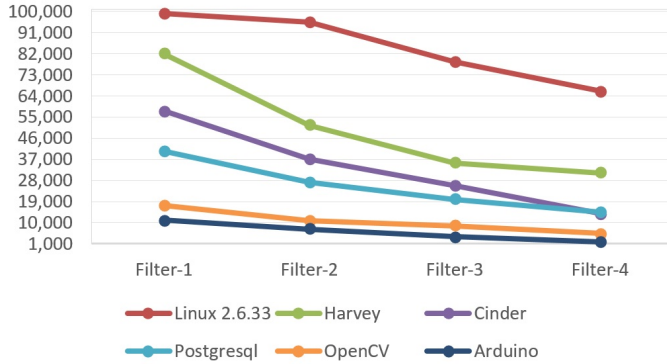


Fig. 5: Graphical representation of Table V

Table IV shows the test projects against their total number of project files, LOC, detection time and a number of clones of Type-1, Type-2, Type-3 in each testing project.

#### F. RQ3: Detection of Full Level and Partial Level Similarities

During full application similarity detection, we detected almost all cloning files from our big code base repository against our subject system entitled *Linux-Kernel* as the results have been shown Table IV. In the case of partial similarity detection, our approach successfully found the systems from code base, which are sharing some source files or part of their source codes. Both full and partial level of similarity detection basically requires finding the similar code fragments in source code.

#### G. RQ4: System Scalability and Efficiency

The execution time actually scales with the size of processed code (LOC) by a tool. As we used MapReduce model for parallel and distributing processing in a cluster, which increase the scalability and efficiency of our approach, meanwhile it is very cost effective and affordable solution. The DCCD

approach is able to scale 324 billion LOC, as it has been tested. The DCCD execution time of indexing and detection is quite faster than other techniques [6]. It is the only technique which has detected Type-1, Type-2 and Type-3 clones from big code source repositories of 27 TB. As the cluster was also used for other purposes, so we measured the time based on its overall load. The results from these case studies show that our proposed approach is very capable of supporting distributed code clone detection and batch clone detection in real time environment for big code bases.

## IV. RELATED WORK

Recently, many clone detection approaches have been developed for small scale and large scale systems [7] [6] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18], which detects clones on different level of granularities. Bundle of code clone detection techniques even for Java Bytecode [19] already have been proposed and implemented [20] [21] [22]. Their results and method have been employed by code clone management tools [23]. Some methods were implemented and embedded in programming platforms, for example, SimEclipse (clone detector plugin), Visual Studio and so on [24]. NICAD [25] by Roy is also considered as hybrid approach. NICAD technique uses *Longest Common Subsequence* algorithm to compare lines of source code [26]. The lexical approaches include CCFinder [5] by Kamiya, CP-Miner by Zhenmin, Boreas by Yong and Yao, FRISC by Murakami, and CDSW by Murakami [20] [27]. The token matching suffix tree based algorithm was used by CCFinder to find out all similar token sequences. Recently there is a tool SourcererCC [24], which performs code clone detection to big code but data set is not big as compared to DCCD, and it is not distributed. Meanwhile we explored different syntactical approaches, i.e CloneDr et al., Wahler et al., Koschke et al. [27], Jiang et al. Hotta et al. Mayrand et al, Kontogiannis, Kodhai, et al. Abdul-El-Hafiz, Kanika et al. [20] [2]. Metric-based [28] need parser to obtain values of metrics, and even it is possible that two code fragments having same metric values maybe not similar code fragments. CONQAT [29] [30] considered as hybrid, clones are detected in main three phases. During our review study in this field, we explored some important semantic techniques, i.e. Komondor and Horwitz (PDGs using CodeSurfer), Duplix (PDGs) by Krinke et al., GPLAG (PDGs using CodeSurfer) by Liu et al, Higo and Kusumoto (PDGs using CodeSurfer), ConQAT (Suffix-tree-based, Token) [4], Funaro (AST) and Agrawal (Tokens) [20]. PDG based techniques are not scalable for large

scale systems [23] [29], because it needs a PDG generator and graph matching, which is little bit expensive.

## V. CONCLUSION AND FUTURE WORK

In this paper, we proposed a Distributed Code Clone Detection (DCCD) technique for token-based code clone detection, which retrieves clones in an efficient way. It exploits an index of source code to achieve the scalability and maintenance of large scale project repositories for big code. The index of 27 TB source code (C, C++, Java, JavaScript) has been built in less than a month. To the best of our knowledge, this is the first approach which has been implemented for large scale systems, which is have been experimented on 27 TB of source code, meanwhile it detects all 3 types of clones very efficiently, especially Type-3 which was very challenging in large scale system. DCCD has been achieved a high accuracy (87%) rate in clone detection for large scale system. DCCD can be adopt at industry level for the detection of clones in big code, which is easily extendible and cost effective. For future concerns, our next target is to extend the current work and continue building a big index for other programming languages i.e. Python, Php, Xml, C#, Vb, Cobol, Text, Sql, Matlab, Ruby, Ada. Meanwhile we are considering to add additional functionalities related to vulnerabilities detection in systems.

## VI. ACKNOWLEDGMENT

This research was supported by Beijing National Research Center for Information Science and Technology (BNRist), and National Natural Science Foundation of China under Grant Nos. 90818021, 9071803.

## REFERENCES

- [1] C. K. Roy and J. R. Cordy, "An empirical study of function clones in open source software," in *Reverse Engineering, 2008. WCRE'08. 15th Working Conference on*. IEEE, 2008, pp. 81–90.
- [2] E. Choi, N. Yoshida, T. Ishio, K. Inoue, and T. Sano, "Extracting code clones for refactoring using combinations of clone metrics," in *Proceedings of the 5th International Workshop on Software Clones*. ACM, 2011, pp. 7–13.
- [3] J. Svajlenko, J. F. Islam, I. Keivanloo, C. K. Roy, and M. M. Mia, "Towards a big data curated benchmark of inter-project code clones," in *Software Maintenance and Evolution (ICSME), 2014 IEEE International Conference on*. IEEE, 2014, pp. 476–480.
- [4] B. Hummel, E. Juergens, L. Heinemann, and M. Conradt, "Index-based code clone detection: incremental, distributed, scalable," in *Software Maintenance (ICSM), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1–9.
- [5] T. Kamiya, S. Kusumoto, and K. Inoue, "Ccfinder: a multilinguistic token-based code clone detection system for large scale source code," *IEEE Transactions on Software Engineering*, vol. 28, no. 7, pp. 654–670, 2002.
- [6] H. Sajjani, V. Saini, J. Svajlenko, C. K. Roy, and C. V. Lopes, "Sourcerercc: Scaling code clone detection to big-code," in *Software Engineering (ICSE), 2016 IEEE/ACM 38th International Conference on*. IEEE, 2016, pp. 1157–1168.
- [7] J. Svajlenko and C. K. Roy, "A machine learning based approach for evaluating clone detection tools for a generalized and accurate precision," *International Journal of Software Engineering and Knowledge Engineering*, vol. 26, no. 09n10, pp. 1399–1429, 2016.
- [8] S. Charalampidou, A. Ampatzoglou, A. Chatzigeorgiou, and P. Avgeriou, "Assessing code smell interest probability: A case study," 2017.
- [9] J. Svajlenko and C. K. Roy, "Fast and flexible large-scale clone detection with cloneworks," in *Proceedings of the 39th International Conference on Software Engineering Companion*. IEEE Press, 2017, pp. 27–30.
- [10] T. Hatano and A. Matsuo, "Removing code clones from industrial systems using compiler directives," in *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)*, May 2017, pp. 336–345.
- [11] S. Alam, Z. Qu, R. Riley, Y. Chen, and V. Rastogi, "Droidnative: Automating and optimizing detection of android native code malware variants," *computers & security*, vol. 65, pp. 230–246, 2017.
- [12] Y. Yuki, Y. Higo, and S. Kusumoto, "A technique to detect multi-grained code clones," in *Software Clones (IWSC), 2017 IEEE 11th International Workshop on*. IEEE, 2017, pp. 1–7.
- [13] S. Kim, S. Woo, H. Lee, and H. Oh, "Vuddy: A scalable approach for vulnerable code clone discovery," in *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 2017.
- [14] F. Lyu, Y. Lin, J. Yang, and J. Zhou, "Stuidroid: An efficient hardening-resilient approach to android app clone detection," in *Trust-com/BigDataSE/I SPA, 2016 IEEE*. IEEE, 2016, pp. 511–518.
- [15] J. Svajlenko and C. K. Roy, "Evaluating clone detection tools with bigclonebench," in *Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on*. IEEE, 2015, pp. 131–140.
- [16] Y. Dang, D. Zhang, S. Ge, R. Huang, C. Chu, and T. Xie, "Transferring code-clone detection and analysis to practice," in *Proceedings of the 39th International Conference on Software Engineering: Software Engineering in Practice Track*. IEEE Press, 2017, pp. 53–62.
- [17] Y. Hu, Y. Zhang, J. Li, and D. Gu, "Binary code clone detection across architectures and compiling configurations," in *Proceedings of the 25th International Conference on Program Comprehension*. IEEE Press, 2017, pp. 88–98.
- [18] C. Ragkhitwetsagul and J. Krinke, "Using compilation/decompilation to enhance clone detection," in *Software Clones (IWSC), 2017 IEEE 11th International Workshop on*. IEEE, 2017, pp. 1–7.
- [19] D. Yu, J. Wang, Q. Wu, J. Yang, J. Wang, W. Yang, and W. Yan, "Detecting java code clones with multi-granularities based on bytecode," in *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, vol. 01, July 2017, pp. 317–326.
- [20] A. Sheneamer and J. Kalita, "A survey of software clone detection techniques," *International Journal of Computer Applications*, pp. 0975–8887, 2016.
- [21] T. Zhang and M. Kim, "Automated transplantation and differential testing for clones," in *Proceedings of the 39th International Conference on Software Engineering*. IEEE Press, 2017, pp. 665–676.
- [22] H. Sajjani, V. Saini, and C. Lopes, "A parallel and efficient approach to large scale clone detection," *Journal of Software: Evolution and Process*, vol. 27, no. 6, pp. 402–429, 2015.
- [23] X. Cheng, H. Zhong, Y. Chen, Z. Hu, and J. Zhao, "Rule-directed code clone synchronization," in *Program Comprehension (ICPC), 2016 IEEE 24th International Conference on*. IEEE, 2016, pp. 1–10.
- [24] V. Saini, H. Sajjani, J. Kim, and C. Lopes, "Sourcerercc and sourcerercc-i: tools to detect clones in batch mode and during software development," in *Proceedings of the 38th International Conference on Software Engineering Companion*. ACM, 2016, pp. 597–600.
- [25] C. K. Roy and J. R. Cordy, "Nicad: Accurate detection of near-miss intentional clones using flexible pretty-printing and code normalization," in *Program Comprehension, 2008. ICPC 2008. The 16th IEEE International Conference on*. IEEE, 2008, pp. 172–181.
- [26] J. R. Cordy and C. K. Roy, "The nicad clone detector," in *Program Comprehension (ICPC), 2011 IEEE 19th International Conference on*. IEEE, 2011, pp. 219–220.
- [27] A. Walenstein, R. Koschke, and E. Merlo, "Duplication, redundancy, and similarity in software: Summary of dagstuhl seminar 06301," *Dagstuhl, Germany, Dagstuhl*, 2006.
- [28] M. S. Aktas and M. Kapdan, "Structural code clone detection methodology using software metrics," *International Journal of Software Engineering and Knowledge Engineering*, vol. 26, no. 02, pp. 307–332, 2016.
- [29] Y. Ueda, T. Kamiya, S. Kusumoto, and K. Inoue, "Gemini: Maintenance support environment based on code clone analysis," in *Software Metrics, 2002. Proceedings. Eighth IEEE Symposium on*. IEEE, 2002, pp. 67–76.
- [30] M. S. Uddin, V. Gaur, C. Gutwin, and C. K. Roy, "On the comprehension of code clone visualizations: A controlled study using eye tracking," in *Source Code Analysis and Manipulation (SCAM), 2015 IEEE 15th International Working Conference on*. IEEE, 2015, pp. 161–170.

# A Hybrid System for Detection of Implied Scenarios in Distributed Software Systems

Anja Slama

*Department of Electrical and  
Computer Engineering  
University of Calgary  
Calgary, Canada  
anja.slama@ucalgary.ca*

Fatemeh Hendijani Fard

*Department of Computer Science  
University of Calgary  
Calgary, Canada  
fhendija@ucalgary.ca*

Behrouz Far

*Department of Electrical and  
Computer Engineering  
University of Calgary  
Calgary, Canada  
far@ucalgary.ca*

**Abstract**—Distributed software systems (DSS) are usually open-ended systems used in different domains such as robotics, energy, health, etc. Multi-agent system (MAS) are a sub-class of DSS. In DSS, maintaining consistency between the system iterations is a complex and expensive task that requires coping with requirements changes and systems upgrading. The interactions, complexity and decentralized communication between components of the DSS may emerge an unwanted behavior. An unwanted behavior, known as Emergent Behavior (EB) or Implied Scenario (IS), could lead to irreversible damages. Thus, detecting IS at an early stage of the system development is needed to decrease the cost of maintaining the system. This work focuses on verification of DSS that its requirements modeled using Message Sequence Chart (MSC). The system verification focuses on the detection of IS using two already proposed different approaches. This article presents the combination of the two approaches by improving the usability of the tool presented in the first approach and the catalogue presented in the second approach. This combination allows the detection of new implied scenarios not detected using the cited approaches separately.

**Index Terms**—Distributed Systems, Multi-agent systems, Implied Scenarios, Message Sequence Chart

## I. INTRODUCTION

Distributed Software Systems (DSS) and Multi-Agent System (MAS) are used in a wide spectrum of domains. MAS are a sub-class of DSS. The size of DSS as well as the lack of central control together contribute to complexity of DSS' behavior.

Scenario-based specification is used to model the interaction between the DSS components that provide the overall system functionality. As each scenario presents a partial specification of the system, the identification of unexpected interaction patterns may not be obvious. These unexpected patterns exhibit system behaviors that do not conform to the system requirement and design. These patterns are known as Implied Scenarios (IS) or Emergent Behavior (EB) [1].

The detection of presence of implied scenarios will usually lead to detect potential system failures at run-time that may prevent irreversible damages to the system and reduce the overall system development and maintenance cost. Thus, there is a need for an analysis tool to inspect the interactions of the components and their impact on the overall behavior of the

system.

In this paper, we present a technique that combines two already existing methodologies, which are based on the analysis of Message Sequence Charts (MSC). This work aims to detect potential ISs and alert the developer to address the issue that does not conform to the concurrent nature of distributed systems. The advantage of the combination of the two methodologies is to get benefit from the automatic analysis of the MSCs using the LTSA tool [2], in order to detect the classical types of IS and also those defined in the catalogue of implied scenarios developed in our research group [3]. Unfortunately, the two techniques in their current form are incompatible. In this work, we compare both methodologies and we propose a technique to exploit the results of the extension of the LTSA tool considering the catalogue of implied scenarios.

## II. BACKGROUND AND RELATED WORK

### A. Preliminaries

In this section, we explain the different types of implied scenarios and we review the concepts used in this work.

*a) Implied Scenarios (IS):* An implied scenario is a behavior of the distributed system that does not conform to the system requirements. We can investigate the system behavior in two levels: component level, and system level [3]. For the component level, the implied scenarios can be categorized into four main classes:

- CLEB-I (Shared states)  
The existence of a shared state of one component in different scenarios. The states occurring after these shared states make the behavior of the component non-deterministic.
- CLEB-II (Respond to different components)  
When a component receives the same message from two other different components, bringing it to the same state, an implied scenario could occur if the component receiving the messages loses track of the senders or receivers' information. In this case, the component could be confused in which MSC to proceed.
- CLEB-III (Local branching)  
This could occur when the component is active, i.e. can

initiate sending a message, in one of the MSCs included in the branch of its high level MSc (hMSC) and where no condition is specified to trigger its functionality in the next MSC.

- CLEB-IV (Race conditions)

The race condition occurs when the behavior of the component depends on the order of receiving messages from other components.

*b) Message Sequence Chart (MSC):* Message Sequence Charts (MSCs) are usually used to formalize and model the requirements of distributed software systems. MSCs are standardized by telecommunication standardization of the International Telecommunication Union (ITU). It models the interaction between the system components [4].

*c) Labelled Transition System Analyser (LTSA):* LTSA is a tool used to automatically check and analyze models in order to confirm the expected behavior from the designer and implementer’s viewpoint [5].

LTSA can translate the MSC chart to Finite State Processes (FSP) definition, a form of state machine description, used to draw the correspondent states machine. This tool offers different features to support verification of properties including safety and progress check [6].

### B. Related Work

Among the model-based analysis approach, many researches have been performed so far. Process divergence and non-local branching choice were identified as under specification in MSC that could cause IS [11]. This research has been extended by generating state machines from the scenario specification in order to detect implied scenarios [12]. The formal methods tool, Spin, has been introduced to verify models for distributed software design [13]. Another approach aimed to detect implied scenarios considering the order between the events specified in the scenario specification [14]. In another work causality among events and using ontology to detect IS has been studied [15], [16].

Fard proposed a catalogue to categorize IS according to their reason of occurrence and devise the solution repositories accordingly. This methodology was based on state machines and social network analysis [3].

## III. METHODOLOGY

In this work, we use the LTSA [2] and the catalog of implied scenarios [3] together. In the following, we compare the effectiveness of each as well as our approach that combines both, to find out implied scenarios.

### A. Modeling System behavior using MSCs

The first step is modeling system interaction in the form of MSCs. The MSCs are translated to Finite State Processes (FSP) in order to model the required behavior and then compiled in the form of Labelled Transition System (LTS). For each component, the state machine of the parallel execution of the component FSPs is generated. The approach behind this phase is fully described in [2].

### B. Detection of Implied Scenarios

Based on the generated FSMs, a verification of the existence of implied scenario is realized based on the LTSA tool. Moreover, to detect further type of implied scenarios, we analyze the traces generated by the LTSA taking into consideration the catalogue of IS. The method used to detect IS in this latter is based on extracting identical states for each agent from the send matrices obtained from the MSCs [10].

To detect the different type of ISs, explained in section II.A, we start by analyzing the set of shared states of the process detected by the LTSA tool. Performing a progress and a safety check on a selected component of the system, the tool generates the related trace based on the MSC modeling messages. The analysis consists of fetching the pattern defined by the catalogue to detect the IS.

## IV. EXPERIMENT

### A. Objectives and research questions

In this work, we seek to answer the following questions:

- RQ: How effective are these approaches to detect Implied Scenarios? We apply the approach on three case studies and we compare the number and the type of ISs detected based on the same case studies. The type of ISs corresponds to the classes of IS presented previously in the section II.A.

### B. Case studies specification

To answer the research question, we use and evaluate the three case studies presented below;

*a) Fleet Management System:* The requirements of the Fleet Management System are locating drivers, considering different itineraries and approximately calculating the departure or arrival time for a minimum commute time [8]. The requirements are demonstrated in Fig. 1 to 4

*b) Greenhouse System:* This case represents the interaction between the agents of the Greenhouse Multi Agent System to control the greenhouse’s environment by managing the available resources. The modeling of this system is composed by two MCSs as presented in [8].

*c) Online Auction System:* This case study represents interaction between the six agents (Controller, Auctioneer, Registrar, Seller, Buyer and Credit Associate) in order to assure the sell and the buy of books online according the rules defined by the auction hosting authority [9].

Table 1 is a recapitulative table of the case studies considered in this work.

TABLE I  
SUMMARY TABLE OF THE CONSIDERED CASE STUDIES

Case Study	# MSCs	# Transitions	# Processes
Feet Management	4	37	8
Greenhouse	2	20	8
Online Auction	6	100	6

### C. Evaluation measurement

We use two evaluation measurements which are the type of implied scenarios according to the catalogue and the number of implied scenarios detected.



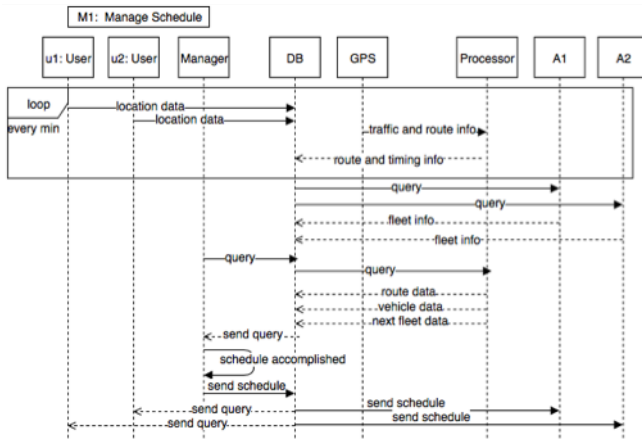


Fig. 1. MSC M1-Schedule management

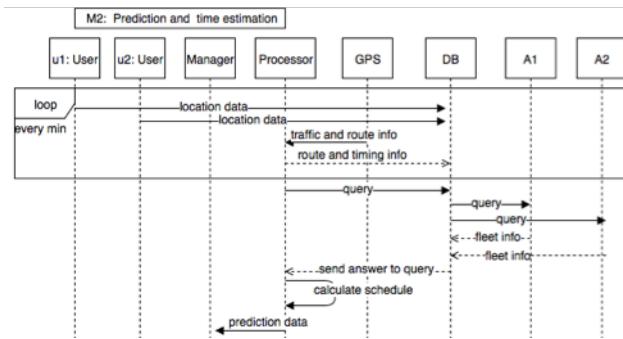


Fig. 2. MSC M2-Prediction and time estimation

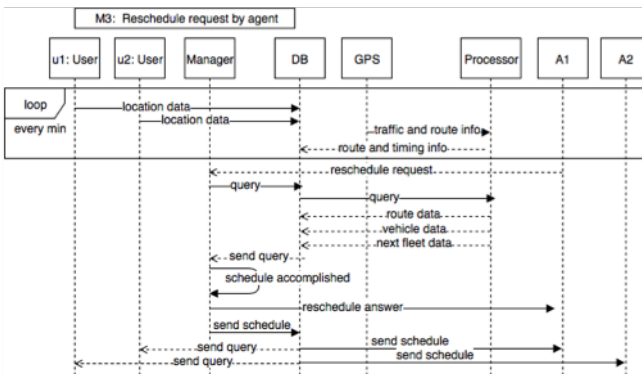


Fig. 3. MSC M3-Reschedule request by Agent

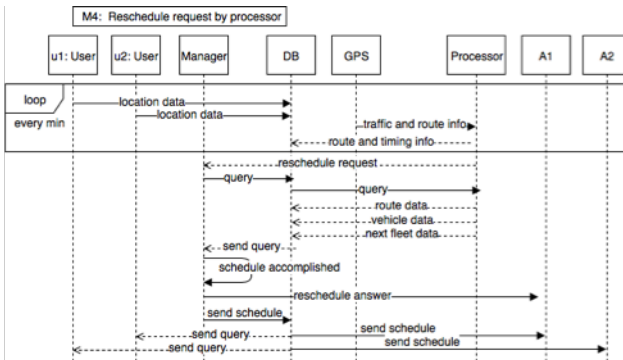


Fig. 4. MSC M4-Reschedule request by Processor

#### D. Procedure

The experiments conducted in this study consist of the detection of implied scenarios from the system modeling. Fig. 5 shows the different components of the system used during this process.

The verification process of the system consists of modeling graphically the system in the form of MSC using LTSA-MSC [17]. MSCs are converted to an FSP specification. LTSA tool checks the system for implied scenarios and safety violation based on this latter. The next step consists in analyzing the traces produced by the LTSA tool to detect further implied scenarios.

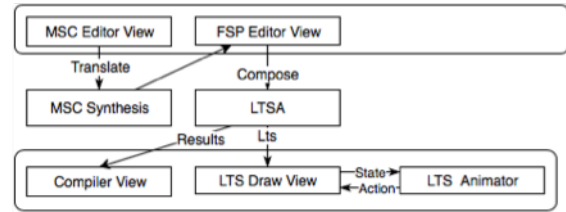


Fig. 5. Different components of the LTSA tool used during the process of detection of IS

#### E. Results

To answer the research question, we compare the number and the type of IS detected. Table II shows the results.

TABLE II  
THE DETECTED ISS IN FLEET MANAGEMENT SYSTEM

Type/Approach	Uchitel	Fard	Combined approach
CLEBI	No IS found	No	Yes
CLEBII	No IS found	Yes	Yes
CLEBIII	No IS found	No	No
CLEBIV	No IS found	No	No
Deadlock	No deadlocks/errors	N/A	Yes

We apply the same procedures on the other case studies presented in Sect.IV.B to get the number of IS detected in each use case as shows the Table III.

TABLE III  
THE NUMBER OF IMPLIED SCENARIOS DETECTED IN EACH CASE STUDY

Type/Approach	Uchitel	Fard	Combined approach
Feet Management System	1	1	3
Greenhouse System	1	1	3
Online Auction System	1	N/A	4

Fig. 6 shows the relation between the number of IS detected and the number of communication messages between the different processes. According to [18], there are a strong correlation between the number of detected IS and the number of messages. Whereas, there are a negative correlation between the number of implied scenarios detected and the number of processes. The probability of emergence of ISs is related to the grow of systems in complexity.

#### F. Discussion

Table III summarizes the number of detected ISs in the considered case studies. The efficiency of our work to detect

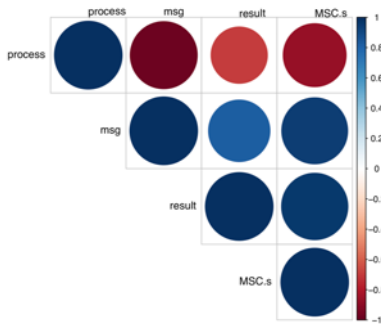


Fig. 6. Correlation between the results and the input data

ISs is based on a hybrid ensemble of the two previous approaches. We show that this method has the ability to detect more IS than Uchitel's and Fard's approaches in the three case studies. The reasons are that we take advantage of the automatic detection of shared states generated by the LTSA tool in order to facilitate the detection of EBs defined by Fard's work and that the approach of this latter do not detect deadlocks. Additional case studies may be needed to build a stronger database of the anti-patterns. By combining the two methodologies through the case studies, we have shown that we can achieve more efficiency in term of the number and type of Implied Scenarios detected.

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

The approach proposed in this work is based on the LTSA tool. The traces generated by the LTSA tool directly impact the number and the type of IS detected. Testing the LTSA tool, we found out that the number of states generated is reduced and limited to a maximum number set by the tool itself to avoid the explosion of states. In future work, we have to further investigate the effect of reducing the states on possibly missing some of the implied scenarios. Moreover, the differentiation between asynchronous and synchronous communications is not considered.

### V. CONCLUSION AND FUTURE WORK

The detection of implied scenarios in requirement and design phase is important for several reasons such as reduction of the project cost, time as well as the prevention of the occurring of unwanted behavior that could have irreversible damages to the systems environment and/or users. Assuring conformity of the system behavior to its requirements is not a trivial task in open ended systems. Therefore, a tool to maximize the detection of IS before the implementation is an immediate need. In this work, we targeted the detection of a greater number of ISs by combining two existing methodologies. The first challenge, was the fact that the explosion of states generated by the LTSA while defining different relation between the MSCs composing the system. The second challenge was to prove that the detected ISs form the system modeling really exist when monitoring the software if the system is developed and we are keen to work further on this challenge in future. Our overall goal is to automatically detect implied scenario at the requirement level using various forms

of communication diagrams. Our technique was a combination of the two existing approaches. We are currently working on a platform for verification, testing and monitoring multi-agent systems using the JADE platform. We plan to extend this work by integrating the generation of the MSCs of the system from the execution trace or from the code within this tool. Moreover, we target the implementation of a plugin for Eclipse IDE that graphically shows the implied scenarios on top of the MSC diagrams.

### VI. ACKNOWLEDGMENT

This work is partially supported by a grant from NSERC, Canada.

### REFERENCES

- [1] J. Chakraborty, D. D'Souza, and K. N. Kumar, "Analysing message sequence graph specifications," in Lecture Notes in Computer Science, 2010, vol. 6415 LNCS, no. PART 1, pp. 549-563.
- [2] S. Uchitel, "Incremental Elaboration of Scenario-Based Specifications and Behavior Models Using Implied Scenarios," 2003.
- [3] F. H. Fard, "Detecting and Fixing Emergent Behaviors in Distributed Software Systems Using a Message Content Independent Method," UNIVERSITY OF CALGARY, 2016.
- [4] S. A. Chernenok and V. A. Nepomniaschy, "Analysis and verification of message sequence charts of distributed systems with the help of coloured Petri nets," Autom. Control Comput. Sci., vol. 49, no. 7, pp. 484-492, 2015.
- [5] G. N. Rodrigues, D. Rosenblum, and J. Wolf, "Reliability Analysis of Concurrent Systems Using LTSA," in ACM/IEEE International Conference on Software Engineering (ICSE) Companion, 2007, pp. 63-64.
- [6] H. Foster, S. Uchitel, J. Magee, and J. Kramer, "LTSA-WS: A Tool for Model-Based Verification of Web Service Compositions and Choreography," 28th Int. Conf. Softw. Eng. (ICSE 06), pp. 771-774, 2006.
- [7] F. H. Fard and B. H. Far, "Detection of implied scenarios in multiagent systems with clustering agents' communications," Proc. 2014 IEEE 15th Int. Conf. Inf. Reuse Integr. IEEE IRI 2014, pp. 237-244, 2014.
- [8] F. H. Fard and B. H. Far, "Detection and verification of a new type of emergent behavior in multiagent systems," in INES 2013 - IEEE 17th International Conference on Intelligent Engineering Systems, Proceedings, 2013.
- [9] F. H. Fard and B. H. Far, "A method for detecting agents that will not cause emergent behavior in agent based systems - A case study in agent based auction systems," in Proceedings of the 2012 IEEE 13th International Conference on Information Reuse and Integration, IRI 2012, 2012.
- [10] F. H. Fard and B. H. Far, "Detecting a certain kind of emergent behavior in multi agent systems applied on MaSE methodology," Can. Conf. Electr. Comput. Eng., pp. 03, 2013.
- [11] H. Ben-Abdallah and S. Leue, "Syntactic detection of process divergence and non-local choice in message sequence charts," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 1217, pp. 259-274, 1997.
- [12] H. Muccini, "Detecting Implied Scenarios Analyzing Non-local Branching Choices," in International Conference on Fundamental Approaches to Software Engineering, 2003, pp. 372-386.
- [13] G. J. Holzmann, "The Model Checker SPIN," Ieee Trans. Softw. Eng., vol. 23, no. 5, pp. 279-295, 1997.
- [14] I. G. Song, S. U. Jeon, A. R. Han, and D. H. Bae, "An approach to identifying causes of implied scenarios using unenforceable orders," Inf. Softw. Technol., vol. 53, no. 6, pp. 666-681, 2011.
- [15] Mousavi Abdelmajid, "Inference of emergent behaviours of scenario-based specifications," University of Calgary, 2009.
- [16] M. Moshirpour, "Model-based Analysis of Software Requirements for Distributed Software Systems," 2016.
- [17] S. Uchitel, R. Chatley, J. Kramer, and J. Magee, LTSA-MSC: Tool support for behaviour model elaboration using implied scenarios, vol. 2619. 2003.
- [18] R. Richard Taylor, EDD, "Interpretation of the Correlation Coefficient: A Basic Review." JDMS, 1990.

# A structured stochastic model for software project estimation in Waterfall models

Ildo Massitela <sup>\*</sup>, Joaquim Assunção <sup>†</sup>, Alan R. Santos <sup>\*</sup>, Paulo Fernandes <sup>\*</sup>

<sup>\*</sup> PUCRS, School of Technology, Porto Alegre, Brazil

<sup>†</sup> UFSM - Department of Applied Computing - Santa Maria, Brazil

<sup>\*</sup> {ildo.massitela, alan.ricardo}@acad.pucrs.br, <sup>†</sup> joaquim@inf.ufsm.br, <sup>\*</sup>paulo.fernandes@pucrs.br

## Abstract

Evaluate team's performance on long-duration projects can be a challenge. It relies on human expertise to perform estimations, which can generate uncertainty and dependency of experienced specialists. Statistical techniques, such as modeling and simulations, are suitable options as tools to support projects estimations. Our goal in this paper is a formal mapping of the main Waterfall model characteristics to stochastic model to predict performance indices of teams such as the real working time.

## 1 Introduction

Simulations of teams' performance can help in solving a series of issues concerning software development, yet estimations and analysis for non-short projects can be a challenge [8]. Considering all the project phases, from extracting requirements to deliver and maintain a product, involves careful planning and resources management. Thus, estimations for a project, play an important role in any medium or big project [7, 10].

Traditional software development methodologies, also known as prescriptive approaches (such as Waterfall), claim their support to comprehensive planning, detailed documentation, and expansive design. Agile approaches have gained significant attention from the software engineering community in the last few years. Unlike traditional methods, Agile methodologies employ short iterative cycles and rely on tacit knowledge within a team as opposed to documentation [1]. In this context, traditional software development approaches will still have their need in large, long-lived projects that have particular safety, reliability or security requirements [1]. Also, Waterfall projects tend to have much longer releases than Agile methods, which makes the estimation of resources even more important.

Performance evaluation for software development projects has become a challenge for both industry and academia. Theoretical models can be adopted as a tool to analyze and to understand different dynamics on software development process to help project leaders to perform a better assessment on issues related to the development context [6, 11, 7].

Characteristics, such as mixed teams, different levels of expertise, different knowledge about the project *etc.*, make it difficult for accurate predictions [13, 5, 3, 12]. In such cases, we may rely on stochastic techniques to represent such apparently random situation. More specifically, a model that uses known data as parameters and known behavior as its structure. Such models should be able to simulate a team's activity, laying on probabilities to estimate its performance.

Although our model does not intend to be a complete solution, we can well cover some of these key points by achieving probabilities through a structured stochastic model. Our model relies on stochastic methods based on a structured formalism, which allows us to perform simulations efficiently, adapt its parameters and scales the model itself depending on the number of team members. This compact and adaptable model can be used both to estimate and evaluate teams' performance. Moreover, we demonstrate the benefits of using the Stochastic Automata Networks (SAN) formalism for the modeling and evaluation of Waterfall development teams.

## 2 Estimation through stochastic models

A stochastic model is a structure which represents a dynamic, and non-deterministic, system or phenomenon through the use of statistical techniques. In our case, a structured modular model, which allows us to manipulate and update the model for different scenarios easily. Such models have a low cost to gain insights compared to the cost, risk or logistics of manipulating a real system.

Our model relies on the stochastic description of the events. With a structure representing a given team, we use parameters caring information concerning the time effectively spent by Junior, Standard, and Senior members. SAN is a structured formalism which describes a complete system as a collection of subsystems that interact with each other [9]. Its behavior is similar to a Markov formalism, except that it is internally represented by a Kronecker descriptor and it allows modular representations, which can be especially useful to create compact representations. SAN models can be solved using specialized tools such as PEPS [2].

<sup>—</sup>\*DOI reference number: 10.18293/SEKE2018-033

### 3 The Model

Our model parameters data was collected through interviews with a project manager at an IT company, a Waterfall project, here in this work, named as project *X*. We used the data collected to feed our model parameters. We use the expression team members to describe the roles of developers and testers.

The model is composed of  $n$  automata representing  $n$  team members; also, we use an abstraction of five different states to describe the team, Working (W), Seeking solution (S), Collaborating (C), Helping others (H) and Reworking (R). Table 1 describes each state.

Table 1: Model states

State	Description
(W)orking	Execution of a given task by each member.
(S)eeking solution	A team member trying to find a solution.
(C)ollaborating	A synchronous contribution between 2 team members, a member receive help or a mutual collaboration is established.
(H)elping	A synchronous contribution with benefit only for another member, $j$ .
(R)eworking	A given team member reworking to solve a problem in a given task.

We assume that a team member can be working (W) and then stop due to the need of information or some necessary condition for a task, we call this state *Seeking solution* (S). Yet, when this team member is not working, sometimes he/she is available to help others, be helped or exchange knowledge about the problem; thus, the states *Collaborating* and *Helping* are used to represent synchronous task between two team members.

Helping (H) is a state that is achieved only when a member  $i$  is helping another member without any benefit for its own problem. Collaborating (C) is a state that represents a member  $i$  being helped by another team member. When an *H* activity ends, a member returns to seek a solution for its problem. A *C* activity ends into two different possibilities, which are: 1, a team member  $i$  can resume working on a new task (going to *W*). 2, a team member  $i$  must return to fix an already started task (*R*), *i.e.*, to rework on a known problem.

For each automaton, five states are linked by a transition associated with events that fire according to specific rates. Those rates are assigned according to flexible parameters based on the average working hours achieved, described by the members of project *X*. For instance, as the event  $a$  is related to an average rate indicating a member that stops working to seek a solution. A frequency of once per day is represented as  $1/9$ , six times per day as  $6/9$ , and so on (Table 2). The event  $q$  is the opposite and has different taxes due to the fact that a member can stop to seek a solution for its problems or to help others.

Table 2: Collected average working hours per team member

Event	Expertise	Rate
$a_i$	Senior	$\tau_{a_i} = 1/9$
	Plain	$\tau_{a_i} = 3/9$
	Junior	$\tau_{a_i} = 6/9$
$q_i$	Senior	$\tau_{q_i} = 6/9$
	Plain	$\tau_{q_i} = 4/9$
	Junior	$\tau_{q_i} = 2/9$
$r_i$	Senior	$\tau_{r_i} = 1/9$
	Plain	$\tau_{r_i} = 1/9$
	Junior	$\tau_{r_i} = 1/9$

We evaluated the model using the project *X* data to set the parameters, however the model parameters can be adjusted and assigned according to different scenarios. For instance, a Junior developer seeks a solution or support more often than a Senior developer, and the combination of professionals with different levels of expertise can be taken into account to accurately describe each specific project behavior. A team, formed by  $n$  members, is formed by a joint network of  $n$  automata, each as illustrated on Figure 1. The events that trigger transitions among the states are presented in Table 3.

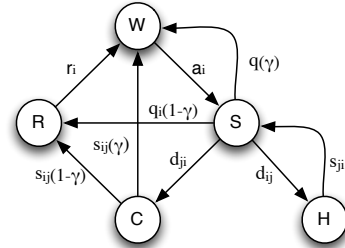


Figure 1: Automaton  $A^{(i)}$  describing possible activities of the  $i^{th}$  member in a team of  $n$  members,  $0 > i, j \leq n \mid i \neq j$ .

Each automaton is described by a set of states (W, S, C, H, R) and a set of events associated with its transitions. For the analysis task, we focus on the state *W* to get probabilities concerning the estimated useful working hours. More clearly, we consider the probability of a state *W* as the productivity of each team member.

In this sense, to determine the rate of the synchronizing events, we started with the following functions, which is given by the expertise of a member, in conjunction with the level of expertise of the other member seeking collaboration (Table 5). So, the better a member is, the fewer this member will be available at state *S*. However, once in *S*, the more likely this member will help the others (Event  $d_{ij}$ ). Also, these members tend to solve problems faster, *i.e.*, they leave the collaboration state faster than those with lower expertise

Table 3: Model events

Event	Description	Rate
$a_i$	Team member $i$ is stop working to seek for the solution for a problem.	$\tau_{a_i}$
$q_i$	Team member $i$ has found a solution by itself and is either resume working (with probability $\gamma$ ), since its problem was not an issue, or he/she is going to re-work (with probability $1 - \gamma$ ) to fix his/her problem.	$\tau_{q_i}$
$d_{ij}$	Team member $i$ is either going to help team member $j$ ( $d_{ij}$ ), or he/she is going to be helped by team member $j$ ( $d_{ji}$ ).	$\tau_{d_{ij}}$
$s_{ij}$	A collaboration between team members $i$ and $j$ is ended and, since $i$ is being helped by $j$ , team member $i$ is going to work or re-work in his/her problem ( $s_{ij}$ ), while team member $j$ is going back to seek his/her own solution. Probability $1 - \gamma$ defines if member $i$ will need to re-work.	$\tau_{s_{ij}}$
$r_i$	A team member $i$ going to work after re-work on a task.	$\tau_{r_i}$

(Event  $s_{ij}$ ). We first use temporary variables ( $\beta$ ) to find the rate set for these events:

$$\beta(s_{ij}) = \tau_{q_i} + (1 - \tau_{q_j}) \dots \beta(s_{ji}) = \tau_{q_j} + (1 - \tau_{q_i})$$

$$\beta(d_{ij}) = \tau_{q_i} + (1 - \tau_{q_j}) \dots \beta(d_{ji}) = \tau_{q_j} + (1 - \tau_{q_i})$$

These temporary variables are needed due to the expertise level's dependency on each team member pair of team member concerned by each synchronous event. Imagine two members, one that has half an hour per day to cooperate and the other has three hours. Between these two members, the cooperation time will be just half hour, which is the available time concerning the busier member. Nevertheless, considering another member of the team, the odds of productive collaboration among members increases as the number of possible member pairs to collaborate increases. Therefore, we use the cumulative sum of all  $\beta$  rates for starting these events. Now, concerning a team with  $n$  members, and a member represented by an automaton  $A^{(i)}$  with  $0 > i, j < n \mid i \neq j$ , the final values for the rates of synchronizing events  $d$  and  $s$  are given by a cumulative sum of the  $\beta$  values:

$$\tau_{d_{ij}} = \sum_{\forall j \mid j \neq i} \beta(d_{ij}) \dots \tau_{d_{ji}} = \sum_{\forall j \mid j \neq i} \beta(d_{ji})$$

$$\tau_{s_{ij}} = \sum_{\forall j \mid j \neq i} \beta(s_{ij}) \dots \tau_{s_{ji}} = \sum_{\forall j \mid j \neq i} \beta(s_{ji})$$

Thus, the cumulative rate for all output events from  $S$  towards  $C$  or  $H$  is given by:  $\sum_{\forall j \mid j \neq i} \tau_{d_{ji}}$ . The same applies for the synchronizing events departing from  $C$ . The cumulative rate for events from  $C$  towards  $R$  and  $W$  is given by:  $\sum_{\forall j \mid j \neq i} \tau_{s_{ij}}$ . Note that from  $H$  there is only one possible

transition that leads back to  $S$ . Once we have a team with  $n$  members, we can define a model composed by  $n$  automata. In such model, it is clear that each event with the member indication  $j$  must represent all team members, but itself ( $i$ ). As for the probability to return to work ( $\gamma$ ) or to re-work ( $1 - \gamma$ ), the more senior a member is the higher will be  $\gamma$ . For project  $X$  a Senior member has  $\gamma = 0.67$ , a Plain member has  $\gamma = 0.44$  and a Junior member has  $\gamma = 0.22$ .

#### 4 Results

Our model target is the state  $W$  since it represents when a team member is performing useful work. Thus, a total working time,  $\Omega$ , is multiplied by the probability of the stationary probabilities of  $W$ . We previously have the real project time and the estimated time, which were given by the project members. Example, for the *Requirements* phase, the estimated total time was 179.7 hours and the real, observed, time was 192.8 hours. Our goal is to achieve a precision as good as the original estimations made by the senior members of the project. Thus, we calculate, for each phase and for each member, the stationary probabilities; which should retrieve a probability of a team member be effectively working. Each of these probabilities is then used to get the calculated working time of phase  $k$  ( $CWT_k$ ). The current parameter makes a simple equation. Given:

- $\Omega$ , team members daily absolute working time;
- $P(W^{(i)})$ , the calculated probability of the  $i^{th}$  team member be actually working;
- $D$ , the total days in the project;
- $n$ , the total number of team members.

$$CWT_k = \sum_{i=1}^n \Omega * P(W^{(i)}) * D$$

Our numerical analysis is obtained by the calculated working time previously described. This is performed for each project phase and for each team member. The model results retrieves the calculated probability for each member. The project phases are: *Requirements, Prototype, Specifications, Development* and *Deployment*.

Table 4: Probabilities achieved to the *Requirements* phase.

Analyst	State	Probability	Analyst	State	Probability
Senior	W	66.4089	Junior	W	20.9505
	S	10.6419		S	56.1932
	C	2.6546		C	17.9955
	H	17.9955		H	2.6546
	R	2.2988		R	2.2060

Considering Table 4, the total time spent to this phase is given by:  $\Omega * P(W^{(i)}) * D$ . Being  $\Omega$  is a constant for the project, 9 hours; and  $D$  is a constant for each phase. In the

first phase, 21 days were spent. Thus, one Senior and one Junior Analyst is respectively:

$$\Omega * P(W^{(1)}) * D = 9 * 0.66 * 21 = 124.7$$

$$\Omega * P(W^{(2)}) * D = 9 * 0.2 * 21 = 37.8$$

The total time is the sum of all team members in the phase, which makes 162.5 hours. Table 5 shows our calculated estimations in hours compared with the actual time and the project manager estimation.

Table 5: Project X, estimated *versus* calculated differences.

Project Phases	Estimated	Actual	Calculated
Requirements	189	192.8	162.5
Prototype	63	61.5	61.1
Prog. specifications	144	138	168.4
Development	912	1069	1285.2
Documentation	171	213	213.7
Deployment	63	51	81.9

Despite a relatively poor performance for the deployment phase, we achieved a precision quite similar to the actual time, and sometimes better than the estimated values by the project manager. These results indicate that this type of approach can be used to support project leads and project managers to evaluate waterfall projects before their execution.

## 5 Final Remarks

Given specific project data, this model can be extended to more sites with different working hours, reworking probabilities and average time spent in activities. The modeler needs to understand more deeply the project and the participants' profile to collect significant data from interviews, surveys or historical data in companies databases. Indeed, we do not explore characteristics such as; the team expertise for a specific task, the historical performance for each phase, the complexity of the tasks and requirements, etc. Yet, these characteristics can be added to adjust the model according to different situations, therefore leading to constant quality and, possibly, improving the accuracy. Nonetheless, the main contribution of this work is a conceptual model for the analysis of effectively working time in Waterfall software development context.

Our model main parameter can be set according to the scenario and previous experiences concerning the targeted team. Despite limitations, our model is structured using Stochastic Automata Network (SAN), which allows new system compositions and behaviors by appending new states and relationships among previously defined entities. As a future work, we will identify characteristics in companies with different sizes and different project types for tests with our model and further research to improve it, which can lead to a more detailed model for various projects.

## References

- [1] M. AWAD, *A comparison between agile and traditional software development methodologies*, University of Western Australia, (2005).
- [2] L. BRENNER, P. FERNANDES, B. PLATEAU, AND I. SBETITY, *PEPS 2007 - Stochastic Automata Networks Software Tool*, in Proceedings of the Fourth International Conference on Quantitative Evaluation of Systems (QEST '07), Washington, DC, USA, September 2007, IEEE Computer Society, pp. 163–164.
- [3] E. CARMEL, *Global software teams: collaborating across borders and time zones*, Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999.
- [4] R. M. CZEKSTER, P. FERNANDES, AND T. WEBBER, *GTA express - A Software Package to Handle Kronecker Descriptors*, in Proceedings of the 6th International Conference on Quantitative Evaluation of SysTems (QEST 2009), Budapest, Hungary, September 2009, IEEE Computer Society, pp. 281–282.
- [5] S. FARAJ AND L. SPROULL, *Coordinating expertise in software development teams*, Management Science, 46 (2000), pp. 1554–1568.
- [6] S. FERREIRA, J. COLLOFELLO, D. SHUNK, AND G. MACKULAK, *Understanding the effects of requirements volatility in software engineering by using analytical modeling and software process simulation*, Journal of Systems and Software, 82 (2009), pp. 1568 – 1577. SI: YAU.
- [7] M. I. KELLNER, R. J. MADACHY, AND D. M. RAFFO, *Software process simulation modeling: why? what? how?*, The Journal of Systems & Software, 46 (1999), pp. 91–105.
- [8] A. MAYRHAUSER, *Experimental Software Engineering Issues: Critical Assessment and Future Directions: International Workshop Dagstuhl Castle, Germany, September 14–18, 1992 Proceedings*, Springer Berlin Heidelberg, Berlin, Heidelberg, 1993, ch. The role of simulation in software engineering experimentation, pp. 177–179.
- [9] B. PLATEAU, *On the stochastic structure of parallelism and synchronization models for distributed algorithms*, SIGMETRICS Perform. Eval. Rev., 13 (1985), pp. 147–154.
- [10] R. SANGWAN, M. BASS, N. MULLICK, D. J. PAULISH, AND J. KAZMEIER, *Global Software Development Handbook (Auerbach Series on Applied Software Engineering Series)*, Auerbach Publications, Boston, MA, USA, 2006.
- [11] S. SETAMANIT, W. WAKELAND, AND D. M. RAFFO, *Using simulation to evaluate global software development task allocation strategies: Research Sections*, Software Process: Improvement and Practice, 12 (2007), pp. 491–503.
- [12] C. U. SMITH AND L. G. WILLIAMS, *Software performance engineering: a case study including performance comparison with design alternatives*, Software Engineering, IEEE Transactions on, 19 (1993), pp. 720–741.
- [13] A. TAWEEL AND P. BRERETON, *Modelling software development across time zones*, Information and Software Technology, 48 (2006), pp. 1–11.

# Revisiting the Conclusion Instability Issue in Software Effort Estimation

Michael Franklin Bosu<sup>1</sup>, Solomon Mensah<sup>2</sup>, Kwabena Bennin<sup>2</sup> and Diab Abuaiadah<sup>1</sup>

<sup>1</sup>Centre for Business, Information Technology and Enterprise, Wintec, Hamilton, New Zealand

<sup>2</sup>Department of Computer Science, City University of Hong Kong, Hong Kong

{michael.bosu, diab.abuaiadah}@wintec.ac.nz, {smensah2-c, kebennin2-c}@my.cityu.edu.hk

**Abstract**—Conclusion instability is the absence of observing the same effect under varying experimental conditions. Deep Neural Network (DNN) and ElasticNet software effort estimation (SEE) models were applied to two SEE datasets with the view of resolving the conclusion instability issue and assessing the suitability of ElasticNet as a viable SEE benchmark model. Results were mixed as both model types attain conclusion stability for the Kitchenham dataset whilst conclusion instability existed in the Desharnais dataset. ElasticNet was outperformed by DNN and as such it is not recommended to be used as a SEE benchmark model.

**Keywords** - Conclusion Instability; Software Effort Estimation; Prediction model; ElasticNet; Deep Neural Network

## I. INTRODUCTION

Software effort estimation (SEE) is part of the broader discipline of empirical software engineering (EMSE) that rely on evidence to make predictions about the estimated effort required to complete software projects. The importance of predicting software effort cannot be overemphasized as it has effect on the estimated budget for software projects. Both the research community and software engineering practitioners have not been able to identify a frontrunner algorithm as several factors such as datasets, experimental team, pre-processing techniques, etc [1] are known to affect the outcome of these algorithms. The challenge therefore is the ability to consistently and uniformly present the results of empirical software engineering models. The current research evidence indicates several conflicting and confusing results especially with regard to the validity of results as it changes with the aforementioned factors. This phenomenon is known as conclusion instability. Conclusion instability refers to the lack of consistent results observed from software engineering experiments [1]. The cause of this effect is attributed to multiple factors such as preprocessing, different datasets, researcher bias, inadequate reporting of research protocol and so on.

The conclusion instability problem in empirical software engineering unfortunately defeats or it is at variance to the goal of science in general which is the ability of an effect to be observed in multiple experimental conditions [1]. This has affected the generalization of results leading to the discovery of the “best” effort estimation algorithm being elusive.

There are two objectives for this study. First is to assess the viability of an approach introduced in this paper to address “the result interpretation issue” which is an aspect of the conclusion instability problem. This will be done by assessing whether the use of different prediction algorithms on a given dataset will yield the same results. The second objective is to evaluate ElasticNet as a benchmark SEE algorithm. Two classical SEE

datasets have been used in the development of effort prediction models in this paper. We will adopt the same procedure employed in our previous paper [2] by generating three effort classes using the density quantile function.

This study applies two SEE modelling techniques on each dataset to assess whether the results are classified into the same classification band. The intention is to assess whether results from multiple estimation models using the same dataset can be classified as generating the same “good” or “bad” results. The advantage of this is that, there will be no need to apply effect size on the models, thus making it easier to interpret and removing one layer of computation because effect size in itself can also lead to conclusion instability when the modelling technique is changed [1]. It is worth noting that we are not comparing estimation models, rather we seek to investigate whether the estimation results of multiple prediction algorithms can be classified into the same band under the same experimental condition (only the dataset changes). We will subsequently assess ElasticNet as a potential benchmarking SEE algorithm with the popular and highly efficient Deep Neural Network (DNN) algorithm.

The rest of the paper is as follows. Section II presents the literature review. Section III is about the method employed in conducting the experiments. Section IV is the analysis of the result and Section V is the discussion and conclusion.

## II. LITERATURE REVIEW

Menzies and Shepperd [1] gave prominence to the conclusion instability issue in an editorial of a Special Edition of the EMSE journal in 2012. The authors explained conclusion instability as the inability of software engineering experiments to discover a certain effect which can be reproduced under multiple experimental conditions such as the use of different datasets, algorithms, researchers, accuracy measures and so on. They [1] identified two major sources of conclusion instability as bias and variance. Bias is said to measure the deviation between predicted values and actual values. Variance on the other hand is measured by the deviation between different predictions of the estimators.

Turhan [3] outlined characteristics of SEE data that could result in conclusion instability. Covariate shift is where the distribution of the training set is different from the validation or test sets and as such the model that was generated by the training data is not able to predict the test data effectively as well as any future project. Prior probability shift is where the distribution of the explanatory variable of the training data and test data are

different. Other types of data shift problems are sample selection bias, imbalanced data, domain shift and source component shift

Menzies et al. [4] conducted a study using 158 SEE methods based on COCOMO dataset features. It was realized that “different datasets sources, different evaluation methods and different random selection of data” led to different results at each occasion which confirm the conclusion instability in SEE methods. They however found four methods that consistently provided better results than the others.

Mair and Shepperd [5] reviewed the result of studies that compared regression techniques with analogy techniques for software cost prediction. They discovered inconsistencies in the result. They found no clear favourite technique as an equal number of studies favoured either approach. They also observed results being inconsistent in cases where even the same datasets were used. This is one of the issues this study is attempting to explain, whether the results were actually inconsistent or there should be a new approach in interpreting the result.

Two research questions (RQ) are used to address the objectives of this study:

RQ1: Do SEE models of different learning algorithms result in the same effort class?

RQ2: How does the performance of Deep Neural Network SEE models differ from ElasticNet SEE models?

### III. METHODOLOGY

#### A. Dataset Description

Two classical datasets from the tera-PROMISE repository (<http://openscience.us/repo/>) have been used in the development of the SEE models. Though these datasets are old, we employed them because they have become the benchmark datasets for SEE studies. A brief description of these datasets is provided.

The **Desharnais** dataset was collected by Jean-Marc Desharnais from ten organizations in Canada. The projects in this dataset were undertaken between 1983 and 1988. The dataset consists of 81 records and 12 attributes, with size measured in function points. We used the 77 version of the dataset as a result of 4 missing records. Summary statistics for relevant features of the dataset are provided in Table 1.

Table 1. Descriptive statistics for Desharnais dataset

Feature	N	Min	Max	Mean	Std.Dev	Skew	Kurt
TeamExp	79	0	4	2.27	1.34	-.042	-1.26
ManagerExp	78	0	7	2.67	1.52	.20	.07
Transactions	81	9	886	179.90	143.32	2.36	7.73
Entities	81	7	387	122.33	84.88	1.34	1.48
Envergure	81	5	52	27.63	10.59	-.11	-.28
PointsNonAjust	81	62	1116	287.05	185.11	1.67	4.16
Effort	81	546	23940	5046.31	4418.77	2.01	4.72

The **Kitchenham** dataset was collected from American-based multinational Computer Sciences Corporation (CSC). This dataset contains information related to 145 software development and maintenance projects that CSC undertook for several clients.

The projects were undertaken between 1994 and 1999 with 10 attributes including the start date and estimated completion

dates. Summary statistics of the relevant features of the Kitchenham dataset are provided in Table 2.

Table 2. Descriptive statistics for Kitchenham dataset

Feature	Min	Max	Mean	Std.Dev	Skew	Kurt
Actual_duration	37	946	206.45	134.09	1.93	6.12
AFP	15.36	18137.48	527.67	1521.99	10.92	126.70
Actual_effort	219	113930	3113.12	9598.01	10.87	125.64

#### B. Data Preprocessing

In order to ameliorate the problem of data quality such as missing data, outliers and inferential data, the two datasets were preprocessed in the following ways for effective and efficient model construction.

We observed all the instances per each dataset to eliminate missing values. Only a handful of missing entries (4 instances) were observed in the Desharnais dataset. Out of the 145 projects in the Kitchenham, three projects were removed as they were found to have missing entries. With the use of kernel density plots and data trimming technique [6], outliers were identified and removed. Cook’s distance was used in the identification and treatment of influential data points during the model construction. In this study, we realized that, these influential data points identified yielded no negative effects on the models when the datasets were normalized using the *z-score* normalization technique as done in our previous study [2].

We selected *prior* features/variables which are known prior to the development of a new software project. Thus, with regard to the Kitchenham dataset, we made use of the *adjusted function points (AFP)* and *project type* as the independent variables and *actual effort* as the dependent variable. With regard to the Desharnais dataset, the independent variables selected are *team experience (teamExp)*, *manager’s experience (managerExp)*, *programming language*, *number of entities*, *number of unadjusted function points (pointsNonAdjust)*, *number of transactions* and *development environment (envergure)*. The dependent variable from the Deshairnais is the development *effort* variable measured in person-hours.

#### C. Effort Estimation Models

Two prediction models have been applied to the two studied datasets. Deep Neural Network (DNN) and the ElasticNet (ENR) algorithms have been used in the development of the SEE models. ENR was found in our previous study [2] as a viable benchmarking model. The ElasticNet proposed by Zou and Hastie [7] is a regularization and variable selection technique which helps in eliminating highly correlated predictor variables from the estimation model.

We benchmarked the prediction results from the ElasticNet model to a complex and robust prediction model, namely a Deep learning model which yielded better prediction accuracy in a previous study [8]. We constructed a DNN which makes use of multiple hidden layers and an output layer with their respective neurons to automatically learn from a set of project cases and gives the resulting prediction for the target (in our case, the software effort of *new* projects). The Levenberg-Marquardt backpropagation optimization training function is employed to update the weights of the neurons in the hidden and output layers



respectively. The hyperbolic tangent activation function is used in each of the neurons for giving the respective outputs. We followed the same experimental setup as done in previous study [2] by using the leave-one-out cross validation for setting up each of the prediction models.

In order to facilitate self-guidance in the interpretation of the level of effort expended, we made use of the classification scheme defined in Mensah et al. [2] to classify the results of our modelling to the appropriate effort class. Software effort classification is the process of categorizing estimated software effort into its respective class. The scheme is based on historical project datasets (historical data being the same as training set in this study). A goal of the classification scheme is to facilitate easy interpretation of the estimated software effort ( $Y_R$ ) in the context of existing organization data. In order to achieve these levels of classification, we discretize the actual efforts of the datasets into three classes (*low*, *moderate* and *high*) based on the density quantile theory. This density quantile theory was utilized because it gives rise to an optimal spacing selection threshold values for categorizing the effort into their respective classes [2].

The selected independent variables together with the software effort (dependent variable) are used to setup the prediction models. We denote the estimated effort values from the models as Output ( $Y_R$ ). The Output ( $Y_R$ ) from the predictive model is then classified into its respective class. The estimated effort, Output ( $Y_R$ ) are categorized into their respective effort classifications. Thus, estimated effort values less than  $Q_l$  are classified as Low Effort and estimated effort values more than  $Q_h$  are classified as High Effort. Lastly, estimated effort values falling within the thresholds [ $Q_l$   $Q_h$ ] are classified as Moderate.

#### D. Accuracy Measures

In order to evaluate the effectiveness of the SEE models, Mean Absolute Error (MAE) and Logarithmic Standard Deviation (LSD) accuracy measures have been employed.

MAE is a risk function that measures the average absolute deviation of the estimated effort values from the actual or true effort values. LSD is defined as the root of the average squared sum of the deviations and the variances between the estimated effort and the true effort. LSD uses the residual in the log-scale, which is independent of size (i.e., homoscedastic). The LSD measure of impurity was applied to the datasets. This index is computed as the within-node variance, adjusted for frequency or case weights (if any). MAE and LSD have been recommended by Foss et al. [9] as a robust and reliable performance measures in setting up SEE models.

Aside of the MAE and LSD evaluation measures, we also considered the Yuen's test and the Cliff's delta ( $\delta$ ) effect size measures as statistical and robust evaluation measures [6]. The statistical test was done at 5% significance level. Even though the Yuen's test is a robust test statistic for assessing statistical significance, it is not enough to make accurate assessment of a tested hypothesis [6]. The Cliff's  $\delta$  effect size is chosen in addition to the Yuen's test since it yields an effective computation measure irrespective of both the experimental and control groups having different sample sizes. Again, it is not affected by outliers and does not assume the sampling data to follow any distribution. The rationale behind the Cliff's  $\delta$  effect

size is that, given two groups of observations without necessarily following the same distribution, this effect size is able to determine the amount of overlap between these two groups.

## IV. RESULTS

We discuss the results of the SEE models built in this section. We provide the classification of the software effort values using the density-quantile function for the normalized/un-normalized datasets in Table 3. Three classes (low, moderate and high) have been created.

Table 3. Software effort classification based on Density-quantile function

Dataset	Normalized data		
	Low	Moderate	High
Kitchenham	0 - 0.1295	0.1296 - 0.2628	> 0.2628
Desharnais	0 - 0.2995	0.2996 - 0.8870	> 0.8871
Dataset	Un-normalized data		
	Low	Moderate	High
Kitchenham	0 - 846.2	846.3 - 2.9122	> 2.9122
Desharnais	0 - 2346.8	2346.9 - 6042.8	> 6042.8

*RQ1: Do SEE models of different learning algorithms result in the same effort class?*

Table 4 presents the recorded losses of the estimated and classified effort from the two learners across each normalized/un-normalized dataset. Here, a '1' denotes a correct classification of the estimated effort and a '0' denotes a wrong classification. It was observed that the estimated effort from the DNN model resulted in correct classification of the effort in both cases of the normalized/un-normalized datasets (Table 4).

Table 4. Classification performance of predicted effort wrt to number of losses

Dataset	Normalized data		Un-normalized data	
	ENR	DNN	ENR	DNN
Kitchenham	1	1	1	1
Desharnais	0	1	0	1

Thus, irrespective of applying data normalization technique, we realized that the DNN resulted in correct classification of the estimated effort values. Table 5 denotes the predicted effort from the two learners for each dataset against the actual effort benchmark.

Table 5. Actual vs. Predicted effort and their respective effort classes

Dataset	Normalized data			Un-normalized data		
	Actual effort	Predicted effort		Actual effort	Predicted effort	
		ENR	DNN		ENR	DNN
Kitchenham	0.2918 (H)	0.5048 (H)	0.3403 (H)	3113.1 (M)	2214.411 (M)	3128.0 (M)
Desharnais	0.7183 (M)	0.9305 (H)	0.8099 (M)	4833.9 (M)	2203.6 (L)	4494.8 (M)

We found that the DNN resulted in improved prediction accuracy against the ElasticNet regression approach.

For the Kitchenham dataset, both normalized and un-normalized, the two learners, ElasticNet and DNN modeling results were correctly classified into the same respective effort class as shown in Table 5. Only the DNN made accurate classification of the estimated effort into the correct classes for the Desharnais dataset whilst the ElasticNet did not. Thus, the Desharnais dataset can be said to exhibit traits of conclusion instability with respect to the ElasticNet prediction model.

*RQ2: How does the performance of Deep Neural Network SEE models differ from ElasticNet SEE models?*

We compared the prediction performance of the Elastic and DNN models using MAE and LSD as shown in Table 6. Note that the shaded cells denote the best performance (minimum values) for either the ElasticNet or the DNN in each case of the normalized/un-normalized dataset. For example, with regard to the normalized Desharnais dataset, we found that the DNN yielded improved prediction accuracy on average irrespective of using the MAE or LSD for evaluation. The results from the statistical test in Table 7 show that the *p-values* from the Yuen’s test resulted in significant differences between the DNN and the ElasticNet irrespective of the application of data normalization. We found from the Cliff’s  $\delta$  effect size that there exists significant differences between using the DNN and ElasticNet for estimating the software effort in all cases with the exception of the normalized Kitchenham dataset (where  $\delta$  was  $0.255 < 0.276$ ).

Table 6. Performance Evaluation of Learners using MAE and LSD

Dataset	Learner	Normalized data		Un-normalized data	
		MAE	LSD	MAE	LSD
Kitchenham	ENR	0.2502	0.5531	3096.7	9598.0
	DNN	0.2177	0.9562	2350.9	7310.6
Desharnais	ENR	0.9922	1.1470	4829.7	4187.8
	DNN	0.5148	0.6908	3136.0	4186.2

Table 7. Statistical significant differences of predicted effort across learners

Learner	Normalized data			Un-normalized data		
	Yuen’s test		Cliff’s	Yuen’s test		Cliff’s
	<i>t</i> -value	<i>p</i> -value	$\delta$	<i>t</i> -value	<i>p</i> -value	$\delta$
<b>Kitchenham Dataset</b>						
DNN vs. ENR	6.37	2.2e-08*	0.255	-9.19	9.9e-14*	0.763**
<b>Desharnais Dataset</b>						
DNN vs. ENR	4.54	2.7e-05*	0.363**	-23.95	4.4e-32*	0.410**

Statistical Significance: \* $p < 0.05$ ; Practical Significance: \*\* $\delta \geq 0.276$

## V. DISCUSSION AND CONCLUSION

In this paper, we built two SEE models, ElasticNet and DNN and applied them to two SEE datasets. The SEE models were executed using both normalized and un-normalized data. The following questions were addressed by the models:

*RQ1: Do SEE models of different learning algorithms result in the same effort class?*

For the Kitchenham dataset, when normalized and un-normalized data were used, both ElasticNet and Deep Neural Network (DNN) modeling results were classified into the correct effort class as the actual classes of the software effort. Correct effort class classification was however, not achieved by ElasticNet models when applied to the Desharnais dataset. This result is particularly interesting as it demonstrates that by using the classification of efforts values it is possible to address the conclusion instability problem as has been achieved for the Kitchenham dataset. Thus, DNN and ElasticNet can achieve conclusion stability irrespective of whether the data is normalized or not.

*RQ2: How does the performance of Deep Neural Network SEE models differ from ElasticNet SEE models?*

Although the ElasticNet models proved superior against other linear regression based SEE models [2], it has performed abysmally against the DNN SEE models and as such it cannot be considered for use as a benchmark model for SEE models. Though this is negative result, we report it in alluding to some of the reasons offered by Kocaguneli et al. [10] in reporting negative scientific results. The result reported in this paper should inform researchers of future studies not to benchmark the ElasticNet SEE models against DNN SEE models. It also offers positive knowledge as it establishes the certainty of the result due to the rigorous experimental approach followed in developing the SEE models in this paper.

Future work will apply DNN and other learners to multiple industrial datasets to determine the existence or otherwise of the conclusion instability issue.

## REFERENCES

- [1] T. Menzies and M. Shepperd, “Special issue on repeatable results in software engineering prediction,” *Empir. Softw. Eng.*, vol. 17, no. 1–2, pp. 1–17, 2012.
- [2] S. Mensah, J. Keung, M. F. Bosu, and K. E. Bennin, “Duplex output software effort estimation model with self-guided interpretation,” *Inf. Softw. Technol.*, vol. 94, pp. 1–13, 2018.
- [3] B. Turhan, “On the dataset shift problem in software engineering prediction models,” *Empir. Softw. Eng.*, vol. 17, no. 1, pp. 62–74, 2012.
- [4] T. Menzies, O. Jalali, J. Hihn, D. Baker, and K. Lum, “Stable rankings for different effort models,” *Autom. Softw. Eng.*, vol. 17, no. 4, pp. 409–437, 2010.
- [5] C. Mair and M. Shepperd, “The consistency of empirical comparisons of regression and analogy-based software project cost prediction,” *2005 Int. Symp. Empir. Softw. Eng. ISESE 2005*, pp. 509–518, 2005.
- [6] B. Kitchenham et al., “Robust Statistical Methods for Empirical Software Engineering,” *Empir. Softw. Eng.*, vol. 22, no. 2, pp. 579–630, 2017.
- [7] H. Zou and T. Hastie, “Regularization and variable selection via the elastic net,” *J. R. Stat. Soc. Ser. B (Statistical Methodol.)*, vol. 67, no. 2, pp. 301–320, 2005.
- [8] S. Mensah, J. Keung, S. G. MacDonell, M. F. Bosu, and K. E. Bennin, “Investigating the significance of bellwether effect to improve software effort estimation,” *Proc. - 2017 IEEE Int. Conf. Softw. Qual. Reliab. Secur. QRS 2017*, pp. 340–351, 2017.
- [9] T. Foss, E. Stensrud, B. Kitchenham, I. C. Society, and I. Myrvtveit, “A Simulation Study of the Model Evaluation Criterion MMRE,” *IEEE Trans. Softw. Eng.* vol. 29, no. 11, pp. 985–995, 2003.
- [10] E. Kocaguneli, T. Menzies, and J. W. Keung, “Kernel methods for software effort estimation. Effects of different kernel functions and bandwidths on estimation accuracy,” *Empir. Softw. Eng.*, vol. 18, no. 1, pp. 1–24, 2013.

# On the UML use in the Brazilian industry: A state of the practice survey

Kleinner Farias, Lucian Gonçalves, Vinicius Bischoff

Graduate Program on Applied Computing (PPGCA)  
Univ. of Vale do Rio dos Sinos, São Leopoldo, Brazil  
kleinnerfarias@unisinos.br, {lucianj, viniciusbischof}  
@edu.unisinos.br

Bruno da Silva<sup>1</sup>, Everton Guimarães<sup>2</sup>, Jacob Nogle<sup>1</sup>

<sup>1</sup>Dept. of Computer Science & Software Engineering  
California Polytechnic State University,  
San Luis Obispo, USA  
{bcdasilv, jnogle}@calpoly.edu

<sup>2</sup>Department of Computer Science, Drexel University,  
Philadelphia, USA  
etg38@drexel.edu

**Abstract— Context:** The Unified Modeling Language (UML) has become the standard for modeling software. Several surveys on the UML usage have been proposed in recent years. However, none of them explores the UML use in specific regional scope, and thus little is known about the practices and perceptions of UML use from the perspective of practitioners in the Brazilian industry. **Objective:** This paper reports on a survey focused on identifying the state-of-the-practice of the Brazilian industry for what concerns the UML usage in real-world settings. **Method:** In total, 222 practitioners from 140 different Information Technology companies have answered an on-line (or printed) questionnaire concerning their UML use experiences, the difficulty in adopting UML and what should be done to increase the UML adoption in practice. **Result:** The results show that: (1) 60 participants (28.2%) have used UML in their daily work, while 73.2% have not; (2) 55.41% of the surveyed participants did not disagree with the statement that UML is the “lingua franca” in software modeling; (3) 61.26% reported to find that the automatic creation of UML diagrams to represent a big picture of the system under development would be useful to boost UML use. **Conclusion:** The UML is not often used in the work life of participants. In addition, no relationship was identified between the use of UML and the participant company being a software factory.

*Unified Modeling Language; UML; Practice; Industry*

## I. INTRODUCTION

The Unified Model Language (UML) [1] is a unification generated from the main methods of software modeling and provides a common notation design covering software analysis to software deployment. Everything in software can be detailed when providing and maintaining software for costumers, resulting in a vast set of diagrams. In the context of the development process, UML assumes that its adoption implies a series of benefits, such as providing a common understanding between team members, comprehension of development details, and increased efficacy in software development. Some studies argue that these benefits are consequences of a full and formal application, where the UML must be applied during the whole software project, and the practitioners have a firm grasp on the usage of this language. As this reality is uncertain, several surveys on the UML usage have been proposed in recent years with the purpose to investigate how the UML is used in practice.

However, there is still a lack of understanding regarding practices and perceptions of UML usage from the perspective of practitioners. In fact, the state-of-the-art UML use in industry diverged on how software industry applies it as well as how practitioners use UML in practice. In addition, the current literature about the UML use in industry was not yet capable of concluding if UML is the de facto standard of modeling languages. Specifically, existing surveys have focused on collecting opinions from participants from different parts of the world. This turns a problem because it assumes that perceptions and fragments from participants opinions spread worldwide are valid on a local and regional scope.

To account for this, this paper focuses on identifying the state-of-the-practice of the Brazilian industry for what concerns the UML usage in world-wide companies. Specifically, this work seeks to investigate how UML is being used in practice, in relation to the relevance of the UML in software projects, i.e. whether the use of UML in software projects and applications is frequent, finding a convergence among developers on the status of UML as a lingua franca, and suggestions for possible improvements in the UML. Exploring these issues is important because we have learned from empirical studies (e.g., [14]) that UML usage has a significantly positive impact on the functional correctness of realized changes in source code in the context of maintenance and evolution tasks.

To achieve these objectives, this study reports on a survey with practitioners of the Brazilian industry. This survey consists of three research questions, and was developed following well-defined guidelines and previous studies, such as [3][4][5][6]. Specifically, a search was conducted on the literature to pinpoint gaps, and to grasp how other studies designed their questionnaires and collected their data. Moreover, we carefully selected a representative set of participants. Thus, all practitioners work or have already acted professionally in several companies in the last years. In total, 222 practitioners from 140 Information Technology companies have answered an on-line (or printed) questionnaire concerning their UML usage experiences, the difficulty in adopting UML, and what should be done to increase the UML usage. In particular, we collected the opinions from practitioners about their practices and perception regarding UML.

## II. RELATED WORK

The literature developed deep discussions on reasons for using UML or not [2][8][10]. However, they did not focus on a particular geographic delimitation. Petre [2] reported an interview-based qualitative investigation involving 50 software engineers in 50 companies over a period of 2 years. The participants were primarily in North America and Europe, but some were from Brazil, India, and Japan. Based on a survey collected from these geographical regions the author found that among the 50 interviewed engineers, 35 reported that they do not use UML. This group reported several reasons for this, including corporate-wide decisions, notation-related issues, and high cost of keeping models synchronized and consistent. This work had an extended version published in [8] which confirmed the previous results of [2]. In a similar manner, the results presented in the study of Gorschek, Temepro, and Angelis [10] pointed out that design models are not used very extensively in industry, and when they are used, the use is informal, with minimal or no tool support, and the notation is not necessarily UML. In addition, they observed that the use of models decreased with an increase in experience and increased with higher level of qualification.

Some studies are optimistic in relation to potential benefits that UML provides during the collaboration and communications between team members [11][12][16]. In [16], the authors collected evidences that UML benefits the collaboration and communication on organizations globally distributed. However, Störrle [12] pointed in his study that there is a possible association between cultural differences and modeling usage, which was considered worth exploring in the future. Finally, the survey conducted by Ho-Quang et al. [11] not target UML practice on industrial closed source projects, but the only the open sources ones.

## III. METHODOLOGY

### A. Goal and Research Questions

The Goals (G) of this study are to (G1) understand the diffusion and relevance of UML use in the Brazilian companies; and (G2) identify improvement points to increase the UML adoption in real-world projects. Based on these goals, the survey aims at addressing three Research Questions (RQ). The two first research questions (RQ 1 and RQ2) address the first goal (G1), while the third question addresses the second goal (G2). The research questions are formulated as follows:

- **RQ1: What is the frequency of UML use in practice?**
- **RQ2: Is UML the “lingua franca” for software modeling?**
- **RQ3: What might improvement points increase the UML use?**

### B. Target Population and Data collection

**Population selection.** According to Kitchenham, the sampling frame of a target population is represented as the finite set of all its members [4]. Thus, a framing population consists of Brazilian software professionals—including developers, software architects, and project managers. These participants represent ones who are in a position to answer the questions and

to whom the results of the survey apply [4]. The participants were selected based on two key criteria: the level of theoretical knowledge and practical experience related to software modeling and programming in mainstream projects.

**Data collection.** In order to enable the data collection, a process of three steps was followed. First, we designed a survey followed well-established guidelines such as [3][4], thereby reducing threats to internal validity. For this, the questionnaire questions concerned on focusing on scrutinizing the research gaps of previous studies; grasp the structures of previously developed questionnaire. In addition, the questionnaire design was based on some of the findings contained on Petre research [2]. In the second step, we concerned in inviting the practitioners. Some practitioners from our industrial contact networks were invited to participate of our research. These participants were from four cities in the southern region of Brazil, including Porto Alegre, São Leopoldo, Canoas, and Novo Hamburgo. Also, working students within six graduate courses at the University of Vale do Rio dos Sinos were invited to respond the questionnaire. Finally, in the last step, data was collected through an on-line questionnaire created by means of Google Docs. We chose this strategy because the questionnaire could be administered quickly, and could also easily collect data from a large number of subjects in geographically diverse locations.

## IV. STUDY RESULTS

### A. RQ1: The frequency of UML use

Figure 1 presents the collected data. Among the 213 participants that answered this question in our study, 60 (28.2%) have used UML in their work day, while 73.2% have not. This result reinforces the finding reported by Petre [2], in which the author reports that 35 out of 50 subjects in her study do not use UML in practice. In order to advance knowledge about the use of UML, we investigated whether the use of UML could be influenced by the type of company. For this, we classify the companies of 113 participants as software factory and not software factory. Table 6 summarizes categorical data about the UML use and company type to create a contingency table. For this, we have applied a series of statistics for checking if there is a relationship between these variables, including Chi-Square and Cramer’s V. We highlight that a chi-square test assumes that observations are independent of one another and that each observation can be assigned to one and only one category.

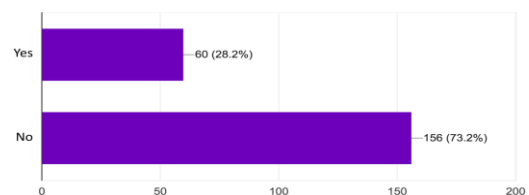


Figure 1. The level of practical experience of participants.

The chi-squared distribution with 1 degree of freedom was 0.115 with p-value = 0.734. Considering the Fisher’s exact test, the p-value produced was 0.445. These small p-values (<0.05) indicates that there is not an association between the UML use and company type. Moreover, the Cramr’s V was also calculated to eliminate any threat related to statistical conclusion validity.

This test is a measure of association between two nominal variables and varies from 0, indicating no association between the variables, to 1, representing a complete association. The value 1 means that the two variables are equal to each other. The low value of the Cramr’s V collected (0.032) also confirmed the previous conclusion.

TABLE I. 2-WAY CONTINGENCY TABLE

	Factory	No Factory	Total
UML use	19	17	36
No UML use	38	39	77
Total	55	56	113
	50.4	49.6	100

**Conclusion of RQ1:** The UML is not often used in the work life of participants. In addition, no relationship was identified between the use of UML and the participant company being a software factory.

**B. RQ2: UML as “lingua franca”**

Among the 222 participants, we obtained 217 (about 97.75%) complete questionnaires considering the UML as “lingua franca” in software modeling and model-driven development. Figure 3 presents the obtained data regarding the RQ1. The data indicate that 55.41% of the surveyed did not disagree with the statement that UML is “the lingua franca” in software modeling. On the other hand, 42.34% agreed that UML is not the “de facto standard” in software modeling, indicating the use of other forms for representing design models, such as drawing or sketching freehand shapes in whiteboard. More specifically, we have recorded that 10% strongly agree, 32% agree, 14% are neutral, 23% disagree, and 19% strongly disagree.

Furthermore, when asked whether the UML would be the “lingua franca” in model-driven development projects, the number of participants that did not disagree with this statement increased from 55.41% to 68.02%—a growth of 22.76%. In addition, following an inverse trend, the amount of surveyed that disagreed also decreased from 42.34% to 29.73%—a drop of 29.79%. In particular, the collected questionnaires indicated 11.71% strongly agree with this statement, 36.94% agree, 19.37% are neutral, 21.17% disagree, and 8.56% strongly disagree.

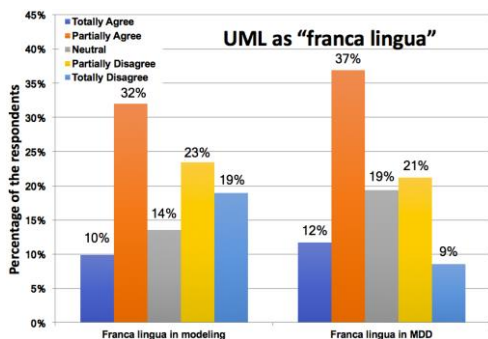


Figure 2. The improvements suggested by participants.

We have learned from previous experimental studies, such as [2][11][13] that there are a set of issues that challenge the

effectiveness of UML as a lingua franca. Nevertheless, practitioners have elaborated ad hoc practices that employ UML effectively in reasoning about and communicating about design, both individually and in collaborative dialogues. For this reason, we believe that such practices have favored the tailored use UML use in companies, justifying the high number of participants who agreed that UML is widely used in real-world settings.

**Conclusion of RQ2:** Although there is no unanimity regarding the adoption of UML as standard modeling language in companies and in MDD projects, the participants see the UML use as broad in companies.

**C. RQ3: Critical Improvements for UML tools**

Figure 4 presents the answers from 217 participants about four suggested improvement points for UML to promote the UML use. These results do not only reveal some existing gaps that prevent the wider use of UML within project teams in real-world settings, but also can be seen as drivers to make the UML a richer and production-ready modeling language. The results show that the UML use could increase if the creation of diagrams representing global aspects of the system could be done automatically, if there were tools that would automatically create function-oriented diagrams rather than generic diagrams, if a round-trip engineering mechanism were a reality, if automatic update of UML diagrams and source code in response to automatically detected inconsistencies between them were allowed, and if there were tools that could provide effective support for collaborative modeling among distributed teams, allowing developers to be aware of changes that developers are making at runtime, something like Google Docs.

In particular, among 222 participants that suggested improvements to UML use, 136 people (61.26%) reported to find that the automatic creation of UML diagrams to represent a big picture of the system under development would be useful to boost UML use. The number of participants who did not disagree is even greater. In total, 210 (94.59%) do not disagree with the use of a Big Picture view as a mechanism to favor the use of the UML. This result is also found considering the other improvement points, i.e., feature-oriented diagrams, round-trip engineering, and collaborative modeling tools.

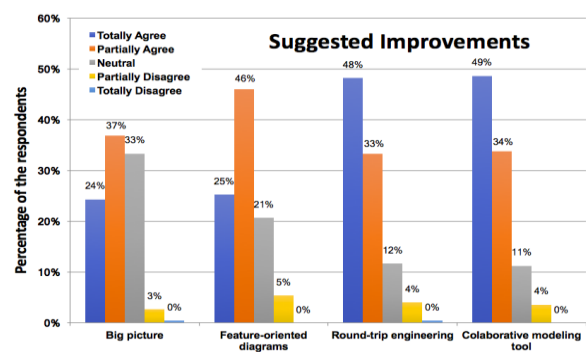


Figure 4. The improvements suggested by participants.

The participants reported the lack of modelling tools, which could support collaboration between teams. For example, this would allow developers to be aware of changes that developers are making at runtime, something that is already supported by

Google Docs. In addition, upcoming modeling tools need to support round-trip engineering for synchronizing related UML diagrams and source code. Given that modeling all structural and behavioral aspects of a software system within a single model is not a trivial task, the UML has proposed a set of diagrams to support a multi-view modeling approach. Thus, different aspects of a system under development are represented by several models, the views. The key challenge identified by the participants is to maintain such different views and their source code consistent and synchronized. According to Chaudron, software development teams use UML during communication and planning of joint implementation efforts [11]. The need for round-trip engineering emerges when project teams need to modify source code and UML diagrams have to be updated to reflect such changes (or vice versa). If UML diagrams are not properly updated, some critical inconsistency may occur.

**Conclusion of RQ3:** *The collected data suggest that creating a “Big Picture” view of the system under development automatically, using functional-oriented diagrams, supporting round-trip engineering, and having more effective collaboration resource would be important improvements to be incorporated into upcoming modeling tools.*

#### D. Discussion

1) *UML usage and Company Issues.* Although the study participants believe, for the most part, that the UML is a “franca lingua” in companies, and that they have theoretical knowledge about UML modeling, they do not use UML frequently. This suggests that there may be some culture issue in the companies, or even in relation to the type of development process adopted, that does not favor the wider use of UML.

2) *Improvement points for Modeling Tools:* We have learned from previous studies, such as [2][8][10][13][15], that UML has been used within teams in the industry for communicating and coordinating their work. Despite this, the results of previous studies, such as [2][8], confirmed in this study, demonstrate that UML has not been widely adopted. The insights from this paper indicate that if the suggested improvement points are implemented by the next modeling tools, UML usage could undergo a positive reversal.

#### V. CONCLUSIONS AND FUTURE WORKS

This paper reported on a survey aimed at identifying the state-of-the-practice of the Brazilian industry for what concerns the UML usage in world-wide companies. The main results show that: (1) 60 (28.2%) have used UML in their work day, while 73.2% have not; (2) 55.41% of the surveyed did not disagree with the statement that UML is the “lingua franca” in software modeling; (3) 61.26% reported to find that the automatic creation of UML diagrams to represent a big picture of the system under development would be useful to boost UML use. Moreover, we might also point out some important improvements to be incorporated into upcoming modeling tools, including the creation of a “Big Picture” view of the system under development automatically, the support to functional-oriented diagrams and round-trip engineering, and more effective collaboration. The results of this research reinforced some evidences already found on state-of-the-art literature about UML in practice, specifically concerning the UML use, which is

barely applied on software projects. In overall, great part of participants know about UML, but they had not used UML in their projects. This work is an initial effort of our research agenda to explore and analyze the UML adoption in industry. Future works will concentrate effort to investigate more aspects regarding the practice of UML in industry such as the benefits and issues that hinder the UML use in practice.

#### ACKNOWLEDGMENT

Thank you to UNISINOS for the teaching and research environment in which they provided to support this research.

#### REFERENCES

- [1] OMG, “UML: Infrastructure specification,” version 2.4. <https://www.omg.org/spec/UML/2.4.1/Superstructure/PDF>, accessed 13 March 2018.
- [2] M. Petre, “UML in practice,” International Conference on Software Engineering (ICSE 2013), 2013, pp. 722–731
- [3] C. Wohlin, P. Runeson, M. Host, M. C. Ohlsson, B. Regnell, A. Wesslen, “Experimentation in Software Engineering,” Springer, 2012.
- [4] B. Kitchenham, S. Pfleeger, “Personal opinion surveys, Guide to Advanced Empirical Software Engineering,” Springer London, 2008, pp. 63–92.
- [5] M. Torchiano, F. Tomassetti, F. Ricca, A. Tiso, G. Reggio, “Relevance, benefits, and problems of software modelling and model driven techniques—a survey in the italian industry,” Journal of Systems and Software, vol. 86, num. 8, 2013, pp. 2110–2126.
- [6] D. Budgen, A. J. Burn, O. P. Brereton, B. A. Kitchenham, R. Pretorius, “Empirical evidence about the UML: a Systematic Literature Review,” Software: Practice and Experience, vol. 41, num. 4, 2011, pp. 363–392.
- [7] M. R. Chaudron, W. Heijstek, A. Nugroho, “How effective is UML modeling?,” Software & Systems Modeling, vol. 11, num 4, 2012, pp. 571–580.
- [8] M. Petre, “‘no shit’ or ‘oh, shit!’: responses to observations on the use of uml in professional practice,” Software & Systems Modeling, vol. 13, num. 4, 2014, pp. 1225–1235.
- [9] J. Singer, S. E. Sim, T. C. Lethbridge, “Software engineering data collection for field studies,” Guide to Advanced Empirical Software Engineering, Springer, 2008, pp. 9–24.
- [10] T. Gorschek, E. Tempero, and L. Angelis, “On the use of software design models in software development practice: An empirical investigation,” Journal of Systems and Software, vol. 95, 2014, pp. 176–193.
- [11] T. Ho-Quang, R. Hebig, G. Robles, M. R. Chaudron, and M. A. Fernandez, “Practices and perceptions of UML use in open source projects,” In International Conference on Software Engineering: Software Engineering in Practice Track (ICSE 2017), 2017, pp. 203–212.
- [12] H. Störrle, “How are Conceptual Models used in Industrial Software Development?: A Descriptive Survey,” In Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering (EASE 2017), 2017, pp. 160–169.
- [13] G. Scanniello, C. Gravino, M. Genero, J. A. Cruz-Lemus, and G. Tortora, “On the impact of UML analysis models on source-code comprehensibility and modifiability,” ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 23, num. 2, 2014, pp. 1–13.
- [14] W. Dzidek, E. Arisholm, L. C. Briand, “A realistic empirical evaluation of the costs and benefits of UML in software maintenance,” IEEE Transactions on Software Engineering, 34(3), pp.407–432.
- [15] C. F. J. Lange, M. R. V. Chaudron and J. Muskens. “In practice: UML software architecture and design description,” IEEE Software, vol. 23, no. 2, pp. 40–46, March–April 2006.
- [16] A.M. Fernández-Sáez, M.R. V. Chaudron and M. Genero. “An industrial case study on the use of UML in software maintenance and its perceived benefits and hurdles”. *Empirical Software Engineering*, pp. 1–65, 2018.

# Using IFML for user interface modeling: an empirical study

Randerson Queiroz, Tayana Conte  
Computer Institute - IComp  
Federal University of Amazonas - UFAM  
Manaus, AM  
(rsq, tayana)@icomp.ufam.edu.br

Anna Beatriz Marques  
Federal University of Ceará - UFC  
Russas, CE  
beatriz.marques@ufc.br

**Abstract** — Front-end interface and user-system interaction are factors that must be carefully considered in software development due to their influence in quality of use. On some occasions, it is the first concern addressed by developers, as it comes naturally from the requirements analysis performed with stakeholders. IFML is a standard language of OMG that supports the abstract description of these front-end interfaces, for software applications on different devices. IFML has been used in the context of Model-Driven Development (MDD) and Model-Driven Architecture (MDA) to describe the elements and behavior of interfaces, aiming to generate code for those interfaces. However, it is necessary to investigate the use of IFML in traditional software development, in order to better understand how it is used for modeling front-end interfaces. This article presents an empirical study that aimed to verify the quality of IFML models created based on a subset of requirements of software two web applications. The quality was defined in terms of models' correctness and completeness. The results showed that the correctness of the models was low, varying from 51% to 55%, while the completeness varied from 66% to 69%. In order to better understand the results, we analyzed syntactic and semantic defects found.

**Keywords-component:** *IFML, User Interface, Software development, Empirical Study.*

## I. INTRODUCTION

The Unified Modeling Language (UML) is widely used to model the system in different stages of traditional software development [6]. However, UML does not present a specific model to describe specifications of front-end interface and user interaction through this interface [9]. To cover this gap, the OMG (Object Management Group) proposed the adoption of Interaction Flow Modeling Language (IFML) [11]. IFML supports the abstract description of front-ends for devices such as computers, laptops, mobile phones and tablets. The objective of IFML is to express the content of these front-end interfaces and the data flows between the front-end components of the application [3].

The IFML uses a single diagram, in which developers can specify the user interface organization, the content displayed for the users and the effect of interface events produced by user interaction or by system notifications [1]. Since IFML is an extension of UML, the artifacts generated by UML notation are usually used as the basis for modeling with IFML [1][11][5].

IFML has often been used in the context of Model Driven Development (MDD) Model Driven Architecture (MDA) [10] to describe the elements and behavior of front-end interfaces, aiming to generate codes of these interfaces [1] [2][8]. However, the concern with the quality of the user interface is not present only in MDD and MDA development contexts. Can the user interface be modeled in a complete way using IFML in traditional software development? What is the correctness of the IFML diagrams created to represent the user interface?

To answer these questions, we conducted an empirical study in which graduate students (with experience in software industry) modeled the front-end interface using IFML, based on requirements of two web applications. The study aimed to verify whether the subjects can model using IFML correctly and completely in the traditional development context (not MDD or MDA). In order to better understand the results, we analyzed the syntactic and semantic defects of the models.

In order to analyze the completeness of a IFML model, we verified whether the elements used by the subjects were sufficient to represent the system requirements. In the analysis of the correctness of a IFML model, we verified whether the elements used by the subjects to represent the requirements were used correctly according to IFML syntax. It is important to conduct empirical studies in order to investigate the models and languages suitable for supporting software development teams in UI design.

The remainder of this paper is organized as follows: Section II presents the basic elements of IFML. Section III shows how we planned and executed the empirical study. Section IV shows the analysis of the results. Finally, Section V presents a discussion and final considerations.

## II. INTERACTION FLOW MODELING LANGUAGE (IFML)

In this section we present a more detailed view of IFML and its elements. For a better understanding, we present a simple example of an IFML diagram.

The Interaction Flow Modeling Language (IFML) is a platform-independent model (PIM) used to express interaction design decisions regardless of the deployment platform [8]. Brambilla et al. [4] claim that IFML is designed to express the content, the user interaction and the control behavior of front-end software applications. Figure 1 shows the basic elements of IFML.

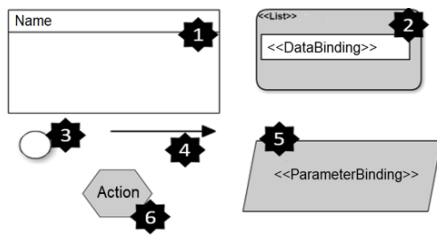


Figure 1. Basic Elements of IFML.

The basic elements of IFML are described below: 1) *ViewContainer* is an interface element that comprises elements displaying content and supporting the interaction and/or other *ViewContainers*; 2) *ViewComponent* is an interface element that displays content, i.e., content and data entry elements contained in *ViewContainers*; 3) *Event* is an occurrence that affects the state of the application. Events can be produced by user interaction, by the application or by an external system; 4) *Navigation Flow* is an update of the interface elements in view or triggering of an action caused by the occurrence of an event. Data may be associated with the flow through *parameter bindings*; 5) *Parameter Binding* is a specification in which an input parameter of a source is associated with an output parameter of a target; 6) *Action* is a piece of business logic triggered by an *event*; it can be server-side (default) or client-side, denoted as (Client).

Figure 2 shows an example of an IFML diagram, describing a user interface where the user can search for a product by entering some search criteria in the Product Search form. The model consists of a Product *view container* (depicting a screen or Web page) that contains two *view components* (visual widgets placed on the screen), i.e., the Product Search form, where the user can enter the search criteria, and the Search Result list, which displays the search results. In addition, a product exclusion *action* can be triggered when the user selects the Exclusion Event associated with the Search Result.

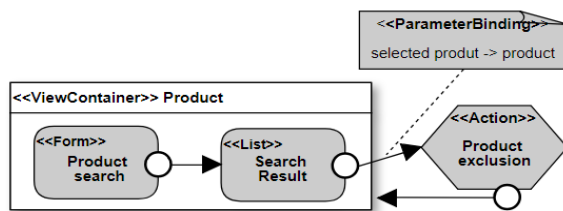


Figure 2. Example of IFML diagram.

### III. EMPIRICAL STUDY

In this section we present the empirical study, the details of the study planning and the execution based on [13]. The artifacts used in the empirical study are available in a technical report [12].

#### A. Study Planning

The empirical study aimed to analyze the use of IFML in the modeling of interfaces in order to analyze the quality of the models in terms of correctness and completeness. Based on the we defined the following research questions: Can the user interface be modeled in a complete way using IFML in traditional software development? What is the correctness of the IFML diagrams created to represent the user interface?

We defined the necessary resources for its execution during the planning of the study, as detailed below:

1) *Context*: we carried out the study in academic context with graduate students.

2) *Subjects*: 16 graduate students participated in the study. All subjects had experience in software industry. However, they had not previously used IFML. We divided the subjects into two groups for modeling different scenarios.

3) *Artifacts used*: we prepared a consent form, in which the subjects could agree or not agree to make their data available for analysis in this research. In order to assist the subjects during modeling, we developed a guide of IFML elements.

4) *Scenarios*: the subjects used functional requirements of a system as the basis for modeling the interface. The requirements were described as scenarios. The scenario that group A received described a system of an airline company, which could be used to track flights to its destination. The scenario that group B received described a website that provided tips of restaurants by area of the city. Both scenarios had the same number of requirements (four requirements).

a) *Group A Scenario*: the scenario of group A contained the following requirements: 1) to access the system with login and password; 2) to track previously registered flights; 3) to register flights to be tracked; and 4) to configure notifications with flight route updates.

b) *Group B Scenario*: this scenario contained the following requirements: 1) to search for restaurant per area; 2) to bookmark a chosen restaurant as favorite; 3) to view tips about the park nest to a favorite restaurant; and 4) to confirm a restaurant booking.

#### B. Study Execution

The study was conducted in a single day lasting 2 hours and 30 minutes. We divided the activity into training, preparation of the activity with receipt of the scenarios and modeling with IFML. The study started with the training. In the training, the subjects received training on all the elements of IFML. The training contained examples and two practical exercises and lasted about 1 hour and 30 minutes. After the training, the subjects received and signed the consent form. The subjects were randomly organized in two groups (A and B). Each subject received a requirements scenario, according to the group he was assigned to.

After receiving the scenarios, the subjects started the modeling step using IFML. After concluding the modeling task, we carried out a discussion with the subjects about the activity they developed. In the discussion, each subject talked a little about their perceptions of the IFML. The discussion was recorded via audio and video. Thus, the audio was transcribed and analyzed.

### IV. DATA ANALYSIS

This section presents the analysis of the results, through which we aimed to identify the completeness and correctness of the modeling. We also performed an analysis in order to identify the syntactic and semantic defects of each model. The subjects' perception about IFML was also analyzed. To perform this



analysis, a researcher inspected the models developed by both groups and a second researcher validated the identified defects. Data from subject S1 were excluded from the analysis because he did not participate in all activity.

A. *Completeness and Correctness*

To achieve the completeness and correctness of each model, we elaborated oracles in order to support the analysis. The oracle corresponds to a possible solution for the scenarios modeling and defines a set of IFML elements that can be used in the solution. In the analysis, we used the oracles as basis for analyzing the elements used by the subjects, the elements not used and the elements that they could use in the model. Each oracle has specific requirements for each scenario. For each requirement, we listed which elements were necessary to model the front-end related to the requirement described. Table I shows part of the oracle, with the required elements for the front-end related to the modeling of the *Access System* requirement.

TABLE I. ORACLE GROUP A

Group A – Scenario A	
Access System	
ViewContainer	
ViewComponent Form	
Submit event	
Type of Data	
Action	
Parameter Binding	

In order to define the completeness of each model, we proceeded with the sum of the number of elements used in the requirements divided by the number of elements necessary to represent the requirements according to the oracle. To obtain the correctness of each model, we also performed a calculation of the number of elements correctly used in the requirements divided by the number of elements defined in the oracle. Table II shows the mean of completeness and correctness of each subject and its respective group.

TABLE II. COMPLETENESS AND CORRECTNESS RESULTS.

Group A									
	S3	S5	S7	S8	S10	S11	S13	S15	Mean
Comple.	81%	67%	46%	86%	31%	82%	70%	89%	69%
Correct.	57%	52%	26%	74%	30%	70%	52%	83%	55%
Group B									
	S2	S4	S6	S9	S12	S14	S16	S17	Mean
Comple.	50%	55%	86%	90%	57%	72%	50%	70%	66%
Correct.	41%	23%	64%	73%	27%	68%	45%	68%	51%

The mean for completeness of the diagrams created by group A and group B were close to 69% and 66%, respectively. This shows that the subjects had difficulty to completely model the requirements. The *ViewComponent* and *Event* elements were not used. In addition, the mean for correctness of the diagrams created by groups A and B were 55% and 51%, respectively. This result shows that even though the elements have been used,

they were used incorrectly, thus decreasing the quality of the diagram created.

Since the subjects from group A and B used different scenarios, the results could have been influenced by the difference between these scenarios. The level of difficulty for modeling a requirement of one scenario could be greater than the requirement of the other scenario. In order to verify whether the scenarios had influenced the results, we applied a statistical hypothesis test. Table III shows the null and alternative hypotheses. The null hypotheses states that: “H<sub>01</sub> - There is no difference in terms of completeness in modeling with IFML based on scenario A or B”, and “H<sub>02</sub> - There is no difference in terms of correctness in modeling with IFML based on scenario A or B”.

TABLE III. NULL AND ALTERNATIVE HYPOTHESES

Null Hypotheses	
H <sub>01</sub> – There is no difference in terms of completeness in modeling with IFML based on scenario A or B	
H <sub>02</sub> – There is no difference in terms of correctness in modeling with IFML based on scenario A or B	
Alternative Hypotheses	
H <sub>A1</sub> – There is a difference in terms of completeness in modeling based on Scenario A in relation to Scenario B	
H <sub>A2</sub> – There is a difference in terms of correctness in the modeling based on Scenario A in relation to Scenario B	

We used the statistical Mann-Whitney non-parametric method. We used  $\alpha = 0.05$  due to the sample size [5]. To perform the tests, we used the SPSS tool v20.0.0. The obtained results support the null hypotheses H<sub>01</sub> and H<sub>02</sub>, indicating that there is no significant difference in the completeness indicator ( $p = 0.878$ ) nor in the correctness indicator ( $p = 0.574$ ) when modeling using IFML with scenario A or B. Therefore, the fact that a group modeled using scenario A or B did not influence in the way the subjects modeled. It means that the scenarios did not influence the results of completeness and correctness, not affecting the results reliability.

B. *Syntactic and semantic defects*

We decided to classify the defects in syntactic or semantic for a better understanding about the defects and their impact. We also explored the possible difficulties in using the elements to model in a correct and understandable way. The concepts of syntactic and semantic properties have been adapted to the context of this study [8]. The definitions we used are: *Syntactic*, defect characterized by the incorrect use of IFML elements; *Semantic*, defect characterized by the incorrect modeling of the problem domain. Figure 3 shows the total number of syntactic and semantic defects, distributed into each group.

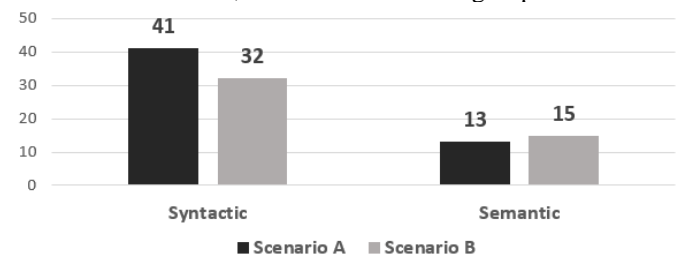


Figure 3. Total number of defect occurrences.

We identified 101 occurrences of defects, being 73 syntactic and 28 semantic, considering all the diagrams created by the subjects. When we did not consider repeated defects, we obtained a list of 24 unique defects. Figure 3 shows that we identified 41 occurrences of syntactic defects and 13 occurrences of semantic defects in diagrams created based on “scenario A”. On the other hand, we identified 32 occurrences of syntactic defects and 15 of semantic defects in diagrams modeled based on “scenario B”. For a better view, we listed the most common defects in Table IV, mentioning the type and number of occurrences for each defect. The list shows only the defects that have been repeated more than once. The other defects that occurred only once are not presented in the list.

TABLE IV. MOST COMMON DEFECTS

Defect	Type of defect	Number of Occurrences
Does not specify the data type	Syntactic	27
Uses the wrong <i>event</i>	Syntactic	15
Does not inform the data that is being passed	Syntactic	14
Uses the wrong <i>ViewComponent</i>	Syntactic	6
Uses Default <i>ViewContainer</i> outside of <i>XOR ViewContainer</i>	Syntactic	5
Did not model the interaction of notification settings	Semantic	5
Did not model the actions cancel and confirm	Semantic	4
Did not model the requirement confirmation View	Semantic	4
The <i>action</i> to choose the View Booking feature was not modeled	Semantic	3

We also listed the number of defects per element, but this was only possible for syntactic defects. Making up a list for semantic defects was not possible because 16 out of the total 28 semantic defects are related to the complete omission of a requirement. These semantic defects are related to the omission of all the elements that subjects could apply in the modeling, so it is not possible to make the exact count of the elements involved in each defect. Figure 4 shows all the syntactic defects found in both scenarios, considering each possible element involved in the defect.

### 1) Syntactic defects

Figure 4 shows the number of occurrences of defects in each element of the IFML. The major number of defects is related to the *Parameter Binding* component, with 16 defects. Although the subjects used this element correctly in order to inform the data related to the interactions modeled, they did not correctly apply the standard specified by the language. There were also 14 occurrences of defects involving the *select event*. The subjects

preferred to only indicate that there was an event, without specifying the element type. There was a large number of defect involving the *ViewComponent List* and *ViewComponent Details*. We identified 34 defects in total. Some of these occurrences are related to the misuse of the *ViewComponent* type used to model the requirement. This may indicate that the subjects did not understand the difference among the types of view components.

As shown in Table IV, the most frequently defect “Does not specify the data type” is related to the *ViewComponent*, in which the subjects did not demonstrate the data of the components by following the standard proposed by the language. The definition of the type of data is typically represented in UML diagrams, such as the class diagram. However, since the only artifact elaborated in this study was the IFML diagram, the omission of this type of information may reduce the understanding of the content of the interface. The “Uses the wrong *event*” defect is directly related to changes in the state of the modeled system. These changes are initiated through the *events* and the subjects did not use the correct *events* for each requirement. In some cases, the subjects did not specify the type of *event*. This shows that they did not understand the difference between *event* types.

The defect “Does not inform the data that is being passed” is related to non-compliance with the standard language in the use of the *Parameter Binding* element. In the particular context of this study, this defect did not impair the comprehension of these data. On the other hand, this defect may be harmful in systems where the data stream is essential for the full operation of the system itself. The defect “Uses Default *ViewContainer* outside of *XOR ViewContainer*” refers to the non-organization of the *containers* in the models. This shows that the subjects who modeled with this defect had difficulty in understanding the organization rule of the *containers*. Considering systems with a large number of tabs and navigations among windows, this defect would be potentially harmful.

### 2) Semantic defects

Among the 28 semantic defects we identified in the diagrams created, 16 of them are complete omissions of a requirement or part of a requirement. We noted that these defects are related to the omission of the elements necessary to adequately model the requirement. The reasons for this phenomenon of omission may be the misunderstanding of the elements involved or the tiresome that the subjects may have felt during the final part of the modeling. However, the semantic defects of omission are related to the requirements that can be considered as the most difficult ones for modeling. For example, in Group A, the semantic defects of omission are related to the requirement “to configure notifications with flight route updates”. This

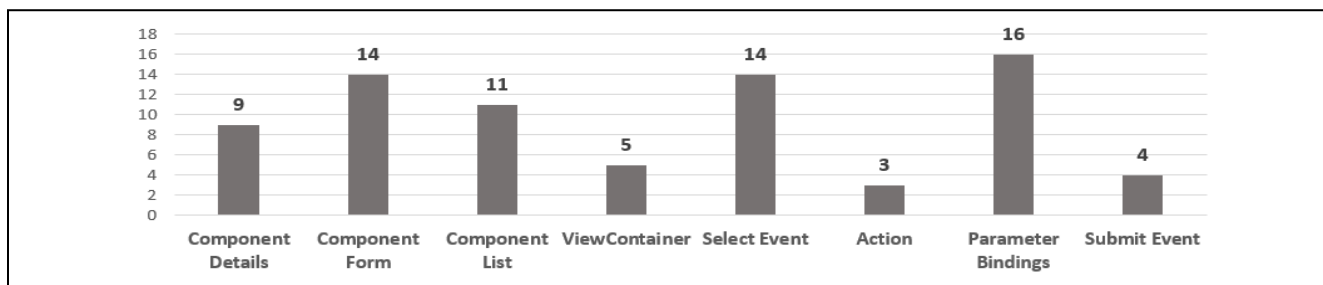


Figure 4. Number of syntactic defects per element

requirement requires a wider combination of elements to model the interaction that the user needed to complete their goal. In Group B, the requirement “to confirm a restaurant booking” requires that the subject models the system feedback for the user. For this requirement, we identified the major number of semantic defects.

The *action* element was related to all semantic defects, so there is a possibility that the major cause of this defect is the misunderstanding about the way that elements should be used. The quantitative data related to semantic defects indicates that the subjects found to be more difficult to use correctly the action element. Other semantic defects were repeated only once, e.g. the defect "Wrong Organization of *Containers*" and "Wrong Flow Sequence between *containers*". Both of them are semantic defects and impair the understanding of the model as a whole.

### 3) Perception of the subjects

The results of the correctness indicated a high number of occurrences of syntactic defects, pointed out a subjects' difficulty in correctly using some of the elements of IFML. Some subjects spontaneously commented something about this difficulty, further reinforcing the results of the study.

Subject S4, for example, reported that “*it is very challenging in the form of representing the screens, even getting tiring because of its many containers and types*”. Subject S15 also reported the same difficulty “*it is difficult to use [the IFML] due to the numerous containers, it ends up leaving the diagram a little messy, making it difficult to visualize and have an idea of the requirements which are being put there*”.

We observed that some subjects considered difficult the IFML elements with similar features. For example, elements like *ViewComponent* and *Event* have several types to be used in different situations in modeling. That difficulty was also reported by subject S3 “*it is difficult to use because it may have many similar components, making it a little confusing when it comes to choosing. There are too many elements, it's very confusing when it comes to doing it*”.

## V. DISCUSSION AND FINAL CONSIDERATIONS

This study aimed to verify how graduate students model the user interface using IFML. The overall results of this study showed that the subjects had difficulties in modeling correctly the front-end based on a scenario describing a set of requirements. Furthermore, the elements of IFML were misused. Syntactic defects showed that the subjects had major difficulties in using the *events* and *view component* elements correctly. Regarding semantic defects, 16 out of the 28 defects were of total omissions of the requirements, which indicate models that do not specify all the requirements described in the scenarios.

In the context of this study, the subjects were able to model the requirements contained in the scenario with a completeness of 69% and 66% (Groups A and B respectively). The correctness of the models was even lower, with a mean of 55% and 51% respectively. The low number of the correctness can be related to the difficulties that the subjects have faced in using the elements of IFML.

With the results of this study, we expect that this research provides a better direction for professionals interested in using

IFML in the user interface design. The results explore how the IFML diagram can be used in the interface design and possible difficulties the professionals can face when using some IFML elements. It is necessary to investigate the use of IFML in different contexts, with subjects from different levels of experience. The results of this research show that it is possible to comprehensively model the front-end interface of a web application using the IFML language. However, some difficulties regarding the elements of the IFML language can affect the correctness of the front-end interfaces.

Finally, as the study was applied in a small sample in an academic environment, it should be replicated with a more representative and heterogeneous sample.

## ACKNOWLEDGMENTS

We would like to thank the financial support granted by UFAM, CNPq through processes numbers 423149/2016-4 and 311494/2017-0, and CAPES through process number 175956/2013.

## REFERENCES

- [1] C. Bernaschina, S. Comai and P. Fraternali, “Formal semantics of OMG’s Interaction Flow Modeling Language (IFML) for mobile and rich-client application model driven development”, *Journal of Systems and Software*, 137, 239-260, 2018.
- [2] C. Bernaschina, S. Comai and P. Fraternali, “IFMLEdit.org: model driven rapid prototyping of mobile apps”, In *Proceedings of the 4th International Conference on Mobile Software Engineering and Systems* p. 207-208, IEEE Press, 2017.
- [3] M. Brambilla and P. Fraternali, “Interaction flow modeling language: Model-driven UI engineering of web and mobile apps with IFML”, Morgan Kaufmann, 2014.
- [4] M. Brambilla, E. Umuhoza and R. Acerbis, “Model-driven development of user interfaces for IoT systems via domain-specific components and patterns”, *Journal of Internet Services and Applications*, 8(1), 14, 2017.
- [5] T. Dyba, V. Kampenes and D. Sjøberg, “A systematic review of statistical power in software engineering experiments”, *Information and Software Technology*, Volume 48, Issue 8, 2006.
- [6] K. Frajták, M. Bureš and I. Jelínek, “Transformation of IFML schemas to automated tests”, In *Proceedings of the 2015 Conference on research in adaptive and convergent systems* p. 509-511, ACM, 2015.
- [7] P. Kamthan, “A framework for understanding and addressing the semiotic quality of use case models” *Model-driven software development: Integrating quality assurance*. Hershey, PA: IGI Global, 2008
- [8] N. Laaz and S. Mbarki, “A model-driven approach for generating RIA interfaces using IFML and ontologies”, In *Information Science and Technology (CiSt)*, 2016 4th IEEE International Colloquium on p. 83-88, IEEE, 2016.
- [9] N. Moreno, P. Fraternali and A. Vallecillo, “WebML modelling in UML”, *IET software*, 1(3), p. 67-80, 2007
- [10] J. R. P. Moreira and R. S. P. Maciel, “Towards a Models Traceability and Synchronization Approach of an Enterprise Architecture” In *The 29th International Conference on Software Engineering & Knowledge Engineering (SEKE)*, 2017.
- [11] OMG, 2015. Interaction flow modeling language (IFML), version 1.0. <http://www.omg.org/spec/IFML/1.0/>.
- [12] R. Queiroz, A. Marques and T. Conte, “USES Technical Report TR-USES-2018-005. Using IFML for user interface modeling”. Technical Report of Usability and Software Engineering Group (USES), 2018. Available in <http://uses.icomp.ufam.edu.br/relatorios-tecnicos/>
- [13] C. Wohlin, P. Runeson, and M. Höst, “Experimentation in Software Engineering: An Introduction, Kluwer International Series in Software Engineering, 2000.

# Modeling mobility and communication in a unified way

Jian-Min Jiang, Xiaofei Yu and Zhong Hong

Department of Software Engineering, Fujian Normal University, Fuzhou 350007, China

*Abstract*—Traditional formalisms model communication and mobility in a separate way. This may cause complex name management and complex analysis for a communicating and mobile system. In this paper, following the ambient calculus [2], we first propose two types of special events, entering and exiting an ambient, as movement events and discuss the relationship of ambients based on mobility. Then a communication model is introduced based on message movement, which can represent synchronous communication, asynchronous communication and broadcasting communication in a unified way. Finally, we show that such a communication model is contained in a general event-based formal model called a dependency structure [4], [5].

## I. INTRODUCTION

Communication and mobility are two essential aspects of complex mobile systems including mobile cyber-physical systems. Most existing formal models and languages have not unified communication and mobility modeling and analysis yet. It is difficult to model and analyze complex mobile systems using formal methods. In the communication aspect, most work assumed a synchronous communication model (e.g., [7]), regarded synchronous communication as special asynchronous communication (e.g., [3]) or independently considered broadcast communication (e.g., [8]). In the mobility aspect, different formal methods have different mechanisms. Mobile Petri nets [1] express process mobility by using variables and colored tokens in an otherwise static net, while dynamic Petri nets [11] extend mobile Petri nets with mechanisms for modifying the structure of a Petri net. The  $\pi$ -calculus [7] is a process algebra where the movement of processes is represented as the movement of channels that refer to processes. One of the most outstanding methods is a calculus of mobile agents called ambient calculus [2]. Ambients are administrative domains and can enter and leave other ambients and perform computations. However, in the ambient calculus, the non-deterministic choice control of processes cannot be expressed like CCS [6]. Moreover, communication needs to use special primitives. More detail refers readers to the literature [5].

Event-based models such as event structures [10] and dependency structures [4], [5] model synchronous communication, asynchronous communication and broadcast communication in a unified way. Movement events are also used to represent mobility [5]. In this paper, we

present a unified approach for modeling communication and mobility.

## II. EVENT AND MOVEMENT EVENT

Event is the primitive notion of event-based formal models [9], [5]. Generally, an event refers to an occurrence of an activity or action. It implicitly contains space and time information.

As defined in the ambient calculus [2], an ambient is a closed and bounded place where computation happens. It can be nested in other ambients and can be moved as a whole [2]. To model mobility, we use the two types of special events: *entering* and *exiting* an ambient. The name of an ambient is contained in the two types of events. When a mobile object enters or exits an ambient, the entering or exiting event itself can contain such an ambient. Such consideration can avoid complex name management. For convenience,  $\mathbf{M}$  and  $\mathbf{A}$  denote the sets of mobile objects (agents) and ambients, respectively.

**Definition II.1** Let  $\mathcal{M} \in \mathbf{M}$  and  $\mathcal{A} \in \mathbf{A}$ .

(1) The event of  $\mathcal{M}$  for entering  $\mathcal{A}$  is called an *entering event*, denoted by  $en_{\mathcal{A}}^{\mathcal{M}}$ , and the event of  $\mathcal{M}$  for exiting  $\mathcal{A}$  is called an *exiting event*, denoted by  $ex_{\mathcal{A}}^{\mathcal{M}}$ .

We define that  $\mathcal{M}$  passes through  $\mathcal{A}$  iff the two events  $en_{\mathcal{A}}^{\mathcal{M}}$  and  $ex_{\mathcal{A}}^{\mathcal{M}}$  occur in sequence.

(2) An event  $e$  is called a *movement event* in  $\mathcal{M}$  iff there exists an ambient  $\mathcal{A} \in \mathbf{A}$  such that  $(e = en_{\mathcal{A}}^{\mathcal{M}}) \vee (e = ex_{\mathcal{A}}^{\mathcal{M}})$ .  $E(\mathcal{M})$  denotes the set of all movement events in  $\mathcal{M}$ .

In the definition, the two events of entering and exiting an ambient are called *movement events*. Note that our framework will not involve other movement events because the two events are sufficiently used to model mobility. According to the definition, movement events in fact contain mobile objects and the ambients involved in the events. For simplicity, *non-movement events* do not consider these information in our framework. Let  $\mathbf{E}$  denote the domain (set) of events including movement and non-movement events. Let  $e_1, e_2 \in \mathbf{E}$ . The notation  $e_1 \rightarrow e_2$  is called a *dependency* that denotes that the occurrence of the event  $e_2$  depends on the previous occurrence of the event  $e_1$ .

### III. THE RELATIONSHIP OF AMBIENTS

The ambient calculus [2] of Cardelli and Gordon focuses on the handling of administrative domains where mobile objects may enter a domain or exit from a domain and in this way may change the topology of the network. Since an ambient is a closed and bounded place, any *movement step* is that a mobile object moves from an ambient to one of its adjacent sibling ambients or from a parent ambient to a child ambient, or vice versa. Therefore, there only exists one of the two kinds of relationships between any two ambients in a mobile system: *parent-child* and *adjacency* (see Figure 1). The parent-child relationship is the inclusion relationship while the adjacency relationship is the sibling relationship.

When a mobile object  $M$  moves from an ambient  $X$  to its adjacent sibling ambient  $Y$ , it needs to first exit  $X$  and then enter  $Y$ . We can use the two events  $ex_X^M, en_Y^M$  to model this situation. When a mobile object  $M$  moves from a parent ambient  $X'$  to its child ambient  $Y'$ , since  $X'$  contains  $Y'$  and  $M$  is located in  $X'$ ,  $M$  only need to enter  $Y'$ , that is, only one entry movement event  $en_{Y'}^M$  occurs. Similarly, when a mobile object  $M'$  moves from a child ambient  $Y'$  to its parent ambient  $X'$ , only one exit movement event  $ex_{Y'}^{M'}$  occurs. Therefore, we give the following definition.

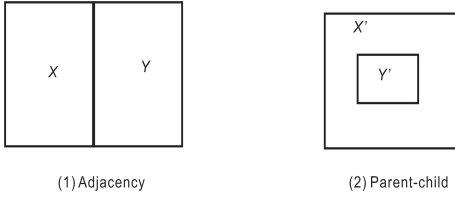


Fig. 1. The relationship of two ambients

**Definition III.1** Let  $\mathcal{A}_1, \mathcal{A}_2 \in \mathbf{A}$ .

(1) (*Parent-child*)  $\mathcal{A}_2$  is called a *child ambient* of  $\mathcal{A}_1$ , denoted by  $\mathcal{A}_2 \in \mathcal{A}_1$  or  $\mathcal{A}_1 \ni \mathcal{A}_2$ , iff for all  $M \in \mathbf{M}$ : (i) if  $M$  moves from  $\mathcal{A}_1$  to  $\mathcal{A}_2$ , there exists only one movement event  $en_{\mathcal{A}_2}^M$ , and (ii) if  $M$  moves from  $\mathcal{A}_2$  to  $\mathcal{A}_1$ , there exists only one movement event  $ex_{\mathcal{A}_1}^M$ . The notation  $\mathcal{A}_2 \notin \mathcal{A}_1$  denotes that  $\mathcal{A}_2$  is not a child ambient of  $\mathcal{A}_1$ .

(2) (*Adjacency*)  $\mathcal{A}_1$  is said to be *adjacent to*  $\mathcal{A}_2$ , denoted by  $\mathcal{A}_1 \ni \mathcal{A}_2$ , iff for all  $M \in \mathbf{M}$ , if  $M$  moves from  $\mathcal{A}_1$  to  $\mathcal{A}_2$ , there only exist two movement events  $ex_{\mathcal{A}_1}^M$  and  $en_{\mathcal{A}_2}^M$  that occur in sequence. The notation  $\mathcal{A}_1 \not\ni \mathcal{A}_2$  denotes that  $\mathcal{A}_1$  is not adjacent to  $\mathcal{A}_2$ .

(3) (*connectivity*)  $\mathcal{A}_1$  is said to be *connected to*  $\mathcal{A}_2$ , denoted by  $\mathcal{A}_1 \gg \mathcal{A}_2$ , iff there exists a sequence  $\mathcal{B}_1 \cdots \mathcal{B}_n (\mathcal{B}_1, \dots, \mathcal{B}_n \in \mathbf{A})$  such that  $\forall i \in \{1, \dots, n-1\}, \mathcal{B}_i \in \mathcal{B}_{i+1} \vee \mathcal{B}_{i+1} \in \mathcal{B}_i \vee \mathcal{B}_i \ni \mathcal{B}_{i+1}$ . The notation  $\mathcal{A}_1 \not\gg \mathcal{A}_2$  denotes that  $\mathcal{A}_1$  is not connected to  $\mathcal{A}_2$ .

The parent-child relationship is bidirectional, that is, mobile objects can move between the parent-child ambients. The adjacency relationship is unidirectional because one ambient is adjacent to another and the reverse adjacency relationship between the two ambients does not necessarily hold. The isolation between ambients means that there does not exist a mobile object that moves between the ambients while the connectivity indicates that any mobile object can move between ambients, but is very possibly unidirectional (because the adjacency relationship as a part of the connectivity is unidirectional).

**Proposition III.1** Let  $\mathcal{A}, \mathcal{B}, \mathcal{C} \in \mathbf{A}$ .

- (1)  $\mathcal{A} \in \mathcal{B} \implies \mathcal{A} \gg \mathcal{B} \wedge \mathcal{B} \gg \mathcal{A}$ .
- (2)  $\mathcal{A} \ni \mathcal{B} \implies \mathcal{A} \gg \mathcal{B}$ .
- (3)  $\mathcal{A} \in \mathcal{B} \in \mathcal{C} \implies \mathcal{A} \gg \mathcal{C} \wedge \mathcal{C} \gg \mathcal{A}$ .
- (4)  $\mathcal{A} \ni \mathcal{B} \ni \mathcal{C} \implies \mathcal{A} \gg \mathcal{C}$ .
- (5)  $\mathcal{A} \gg \mathcal{B} \gg \mathcal{C} \implies \mathcal{A} \gg \mathcal{C}$ .

**Proof** This proof is straightforward.  $\square$

Proposition III.1 shows that there exists the following properties of the relationship between ambients: (1) if one ambient is a child of another, the two ambients are connected to each other; (2) adjacent ambients have unidirectional connectivity; (3) the transitivity of parent-child relations implies bidirectional connectivity; (4) the transitivity of adjacency relations indicates unidirectional connectivity; (5) the connectivity relation is transitive; and the isolation relation is symmetric.

**Theorem III.1** Let  $\mathcal{A}, \mathcal{B} \in \mathbf{A}$ .

- (1) If  $\mathcal{A} \in \mathcal{B}$ , then  $\forall X \in \mathcal{A}, \forall Y \in \mathcal{B}, X \gg Y \wedge Y \gg X$ .
- (2) If  $\mathcal{A} \ni \mathcal{B}$ , then  $\forall X \in \mathcal{A}, \forall Y \in \mathcal{B}, X \gg Y$ .
- (3) If  $\mathcal{A} \gg \mathcal{B}$ , then  $\forall X \in \mathcal{A}, \forall Y \in \mathcal{B}, X \gg Y$ .
- (4) If there exist  $X \in \mathcal{A}, Y \in \mathcal{B}$  such that  $X \gg Y$ , then  $\mathcal{A} \gg \mathcal{B}$ .

**Proof**

(1) By Proposition III.1(1),  $\forall X \in \mathcal{A} \implies X \gg \mathcal{A} \wedge \mathcal{A} \gg X$ ,  $\forall Y \in \mathcal{B} \implies Y \gg \mathcal{B} \wedge \mathcal{B} \gg Y$ , and  $\mathcal{A} \in \mathcal{B} \implies \mathcal{A} \gg \mathcal{B} \wedge \mathcal{B} \gg \mathcal{A}$ . Since  $X \in \mathcal{A}$  and  $\mathcal{A} \in \mathcal{B}$ ,  $X \in \mathcal{A} \in \mathcal{B}$ . Then, according to Proposition III.1(3),  $X \gg \mathcal{B} \wedge \mathcal{B} \gg X$ . Since  $Y \gg \mathcal{B} \wedge \mathcal{B} \gg Y$ ,  $X \gg \mathcal{B} \gg Y$  and  $Y \gg \mathcal{B} \gg X$ . Therefore, by Proposition III.1(5),  $X \gg Y \wedge Y \gg X$ .

(2) By Proposition III.1(1),  $\forall X \in \mathcal{A} \implies X \gg \mathcal{A}$  and  $\forall Y \in \mathcal{B} \implies \mathcal{B} \gg Y$ . Since  $\mathcal{A} \ni \mathcal{B}$ , by Proposition III.1(2),  $\mathcal{A} \gg \mathcal{B}$ . Therefore,  $X \gg \mathcal{A} \gg \mathcal{B} \gg Y$ . According to Proposition III.1(5),  $X \gg Y$ .

(3) Since  $X \in \mathcal{A}, Y \in \mathcal{B}$ , by Definition III.1(3),  $X \gg \mathcal{A} \wedge \mathcal{B} \gg Y$ . Also, since  $\mathcal{A} \gg \mathcal{B}$ ,  $X \gg \mathcal{A} \gg \mathcal{B} \gg Y$ . By Proposition III.1(5),  $X \gg Y$ .

(4) Since  $X \in \mathcal{A}, Y \in \mathcal{B}$ , by Definition III.1(3),  $\mathcal{A} \gg X \wedge Y \gg \mathcal{B}$ . Also, since  $X \gg Y$ ,  $\mathcal{A} \gg X \gg Y \gg \mathcal{B}$ . By Proposition III.1(5),  $\mathcal{A} \gg \mathcal{B}$ .  $\square$

Theorem III.1 states that (1) if one ambient is a child of another, then their children are connected together, (2) if two parent ambients are adjacent to each other, then their children are connected together, (3) if two parent ambients are connected together, their children are all connected together, and (4) if there exist child ambients of two ambients are connected together, then such two ambients are connected together.

#### IV. COMMUNICATION MODEL

At a high abstract level, the exchanged information on communication is generally called *messages*. Different communication mechanisms and media form different types of communication such as asynchronous communication, synchronous communication and broadcasting communication. Synchronous communication requires that the sender should wait until the receiver is ready for the message exchange, and then they synchronize by executing the sending and the receiving activity simultaneously. Thus, since a message is directly sent to the receiver by a sender under synchronous communication, we may consider that a message directly moves from the sender into the receiver. By contrast, asynchronous communication requires specific communication media (queues or channels) which store messages. In this setting, a message leaves the sender, and then enters a communication medium. If the receiver needs the message, then the message exits the communication medium and enters the receiver. The broadcasting communication model means that one message can be cloned and transmitted from one sender to multiple receivers.

In a mobile system, mobile objects and ambients may communicate with each other. They are generally composed together by the communication devices (media) and these communication media are also ambients. Messages are mobile objects, and the senders and receivers of messages are mobile objects or ambients in a mobile system. Moreover, there exist many communication media, for example, any communication node on any network. We sometimes need to model all the communication media in order to model and reason about networks and their protocols.

**Definition IV.1** A communication model is a tuple  $\mathbb{CM} = \langle \mathbf{Mm}, \mathbf{Am}, \mathbf{Em}, \mathbf{Rd} \rangle$  where

- $\mathbf{Mm} \subseteq \mathbf{M}$  is the set of mobile messages,
- $\mathbf{Am} \subseteq \mathbf{A}$  is the set of ambients participating in the movement of messages,
- $\mathbf{Em} \subseteq \{e \mid m \in \mathbf{Mm}, A \in \mathbf{Am}, e = ex_A^m \vee e = en_A^m\}$  is the set of movement events of messages, and
- $\mathbf{Rd}$  is the dependency relation on the set  $\mathbf{Em}$  such that for all  $m \in \mathbf{Mm}$ ,
  - (1) if  $\exists A, B \in \mathbf{Am} : A \Subset B$  and  $m$  moves between  $A$  and  $B$ , then  $ex_A^m, en_B^m \in \mathbf{Em}$ ,

- (2) if  $\exists A, B \in \mathbf{Am} : (A \Rightarrow B) \vee (\exists C : A \Subset C \wedge B \Subset C)$  and  $m$  moves from  $A$  to  $B$ , then  $ex_A^m, en_B^m \in \mathbf{Em} \wedge ex_A^m \rightarrow en_B^m \in \mathbf{Rd}$ ,
- (3) if  $\exists A, B, C \in \mathbf{Am} : A \Subset B \Subset C$  and  $m$  moves between  $A$  and  $C$ , then  $ex_A^m, ex_B^m, en_A^m, en_B^m \in \mathbf{Em} \wedge (ex_A^m \rightarrow ex_B^m, en_B^m \rightarrow en_A^m \in \mathbf{Rd})$ , and
- (4) if  $\exists A \in \mathbf{Am} : m$  passes through  $A$ , then  $en_A^m \rightarrow ex_A^m \in \mathbf{Rd}$ .

The communication model considers not only message exchanges between general senders and receivers, but also can handle message migration from a parent ambient to a child, or vice versa.

**Proposition IV.1** Let  $\mathbb{CM} = \langle \mathbf{Mm}, \mathbf{Am}, \mathbf{Em}, \mathbf{Rd} \rangle$  be a communication model. If  $\exists A, B \in \mathbf{Am} : A \gg B$ , then there exists a message  $m \in \mathbf{Mm}$  such that  $m$  moves from  $A$  to  $B$ .

Next, we discuss the relationship of synchronous communication, asynchronous communication and broadcasting communication.

**Definition IV.2** Let  $\mathbb{CM} = \langle \mathbf{Mm}, \mathbf{Am}, \mathbf{Em}, \mathbf{Rd} \rangle$  be a communication model.

- (1)  $\mathbb{CM}$  is said to be *synchronous* if  $\forall (e_1, e_2) \in \mathbf{Rd}, \forall e, e' \in \mathbf{Em} : (e, e_1) \notin \mathbf{Rd} \wedge (e_2, e') \notin \mathbf{Rd}$ .
- (2)  $\mathbb{CM}$  is said to be *asynchronous* if  $\forall (e_1, e_2) \in \mathbf{Rd}, \exists e, e' \in \mathbf{Em} : (e, e_1) \in \mathbf{Rd} \vee (e_2, e') \in \mathbf{Rd}$ .
- (3)  $\mathbb{CM}$  is said to be *broadcasting* if  $\exists e \in \mathbf{Em} : |e^\bullet| > 1$  where  $e^\bullet = \{e' \in \mathbf{Em} \mid (e, e') \in \mathbf{Rd}\}$ .

In a synchronous communication model, messages only move between two ambients and there does not exist an ambient between sender and receiver ambients. In an asynchronous communication model messages may move from sender ambients to receiver ambients and pass through the ambients between senders and receivers. A broadcasting communication model means that there exist multiple message movement events depending on the same movement events.

**Proposition IV.2** If a communication model is broadcasting, it is synchronous or asynchronous.

This proposition shows that broadcast communication is synchronous or asynchronous communication.

#### V. DEPENDENCY STRUCTURE

An event is an occurrence of an activity or action. If an event occurs, such an event is said to be *available*; otherwise it is *unavailable*. A dependency structure uses an *event set* (a set of events) as a basic element. If all events in an event set are *available*, such an event set is said to be *available*; otherwise it is said to be *unavailable*. For convenience, we first give some notations. Given a set  $\mathcal{E}$ ,  $|\mathcal{E}|$  and  $P'(\mathcal{E})$  respectively denote the size and the

power set of  $\mathcal{E}$ , and  $P(\mathcal{E})=P'(\mathcal{E}) \setminus \{\emptyset\}$ .  $\mathbf{E}$  denotes the set of events.

**Definition V.1** A **dependency structure** ( $\mathcal{DS}$ ) is a tuple  $\langle \mathcal{E}, \mathbb{I}, \mathbb{T}, \mathbb{S}, \mathbb{C}, \mathbb{P}, \mathbb{F} \rangle$  with

- $\mathcal{E} \subseteq \mathbf{E}$ , a finite set of events,
- $\mathbb{I} \subseteq P'(\mathcal{E})$ , the set of initially available event sets,
- $\mathbb{T} \subseteq P(\mathcal{E}) \times P(\mathcal{E})$ , the (asymmetric) *transformation* relation,
- $\mathbb{S} \subseteq P(\mathcal{E})$ , the *synchronism* relation such that  $\forall A \in \mathbb{S} : |A| > 1$ ,
- $\mathbb{C} \subseteq P(\mathcal{E})$ , the *choice* relation such that  $\forall A \in \mathbb{C} : |A| > 1$ ,
- $\mathbb{P} \subseteq P(\mathcal{E}) \times P(\mathcal{E}) \setminus \mathbb{T}$ , the (irreflexive and asymmetric) *priority* relation, and
- $\mathbb{F} \subseteq P'(\mathcal{E})$ , the set of finally available event sets.

Here, for all  $A, B \subseteq \mathcal{E}$ ,  $(A, B) \in \mathbb{T}$  (resp.  $\mathbb{P}$ ) is called a *transformation* (resp. *priority*) *dependency*, denoted as  $A \rightarrow B$  (resp.  $A \rightarrow\!\!\!-\! B$ ), all read as  $B$  *depending on*  $A$ .

When the occurrences of some events completely depend on those of other events, the two groups of events form a (causal) transformation relationship. Transformation is a binary relation between event sets where the intuitive interpretation of a transformation  $(A, B)((A, B) \in \mathbb{T})$  is that the availability of all events in  $B$  depends on the occurrences of all events in  $A$ . A set  $A \in \mathbb{S}$  is called a *synchronism set*, and a set  $B \in \mathbb{C}$  is called a *choice set*. Definition V.1 requires that any synchronism or choice set should have at least two events. The synchronism relation is not a binary relation. Any set  $A \in \mathbb{S}$  means that all events in  $A$  synchronize with each other. Only if all events in  $A$  have occurred, the events that depend on them will occur. Any set  $B \in \mathbb{C}$  means that all events in  $B$  are mutually exclusive, that is, if one event occurs, the others cannot occur.

Priorities only control the transformation relation and are not related to synchronism and choice. Actually, the synchronism and choice relations also control the transformation relation. The initially available event set means that the events in such an event set have been available before a system starts to run. The finally available event set means that when the execution of a system makes the events in such an event set available, the system or its subsystems stop running.

**Theorem III.1** If  $\mathbb{CM} = \langle \mathbf{Mm}, \mathbf{Am}, \mathbf{Em}, \mathbf{Rd} \rangle$  is a communication model of a communicating and mobile system, then there exists a dependency structure  $\mathcal{DS} = \langle \mathcal{E}, \mathbb{I}, \mathbb{T}, \mathbb{S}, \mathbb{C}, \mathbb{P}, \mathbb{F} \rangle$  such that  $\forall (e, e') \in \mathbf{Rd}, (\{e\}, \{e'\}) \in \mathbb{T}$ .

**Proof** Let  $\mathbf{CMS}$  be a communicating and mobile system. Let  $\mathbb{CM} = \langle \mathbf{Mm}, \mathbf{Am}, \mathbf{Em}, \mathbf{Rd} \rangle$ ,  $\mathcal{DS} = \langle \mathcal{E}, \mathbb{I}, \mathbb{T}, \mathbb{S}, \mathbb{C}, \mathbb{P}, \mathbb{F} \rangle$  be the communication model and the behaviour model of  $\mathbf{CMS}$ , respectively. Then, since  $\mathbb{CM}$  represent the communication behavior of  $\mathbf{CMS}$ , by Definition IV.1, we can obtain the dependency relation  $\mathbf{Rd}$  of message movement. Obviously, since

the communication behavior of  $\mathbf{CMS}$  is part of the whole behavior of  $\mathbf{CMS}$ , and a dependency structure can model the whole behavior of  $\mathbf{CMS}$  (including event dependency, synchronism, choice, priority and loop), the dependency structure can model the communication behavior of  $\mathbf{CMS}$ . According to Definition IV.1, for all  $(e, e') \in \mathbf{Rd}$ , we can guarantee  $(\{e\}, \{e'\}) \in \mathbb{T}$ .  $\square$

This theorem shows that the dependency structure model contains the communication model of unifying mobility and communication, that is to say, a dependency structure can model not only behavior of a communicating and mobility system, but also represent mobility and communication of such a system in a unified way.

## VI. CONCLUSION

We have discussed the relationship of ambients based on mobility and presented an event-based communication model. We have also shown that a dependency structure can not only unify synchronous, asynchronous and broadcasting communication, but also specify mobility and communication in a unified way. As a general event-based formal model, a dependency structure is easily used to model and reason about mobile cyber-physical systems.

## ACKNOWLEDGMENTS

This work is supported by National Natural Science Foundation of China (No. 61772004) and the Natural Science Foundation of Fujian province (No. 2018J01777).

## REFERENCES

- [1] Andrea Asperti and Nadia Busi. Mobile petri nets. *Mathematical Structures in Computer Science*, 19(06):1265–1278, 2009.
- [2] Luca Cardelli and Andrew D Gordon. Mobile ambients. In *International Conference on Foundations of Software Science and Computation Structure*, pages 140–155. Springer, 1998.
- [3] Florent Chevrou, Aurelie Hurault, and Philippe Queinnec. On the diversity of asynchronous communication. *Formal Aspects of Computing*, 28(5):847–879, 2016.
- [4] Jian-Min Jiang, Huibiao Zhu, Qin Li, Yongxin Zhao, Lin Zhao, Shi Zhang, Ping Gong, and Zhong Hong. Analyzing event-based scheduling in concurrent reactive systems. *ACM Transactions on Embedded Computing Systems (TECS)*, 14(4):86, 2015.
- [5] Jian-Min Jiang, Huibiao Zhu, Qin Li, Yongxin Zhao, Lin Zhao, Shi Zhang, Ping Gong, Zhong Hong, and Donghuo Chen. Event-based mobility modeling and analysis. *ACM Transactions on Cyber-Physical Systems*, 1(2):9:1–9:32, February 2017.
- [6] Robin Milner. *A Calculus of Communicating Systems*. Springer-Verlag, New York, NY, 1982.
- [7] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes. *Information and Computation*, 100(1):1–40, September 1992.
- [8] Sebastian Nanz, Flemming Nielson, and Hanne Riis Nielson. Static analysis of topology-dependent broadcast networks. *Information and Computation*, 208(2):117–139, 2010.
- [9] Mogens Nielsen, Gordon Plotkin, and Glynn Winskel. Petri nets, event structures and domains, part i. *Theoretical Computer Science*, 13(1):85–108, 1981.
- [10] Glynn Winskel and Mogens Nielsen. *Models for Concurrency*. Handbook of Logic in Computer Science, Semantic Modelling. Oxford Science Publications, Oxford, 1995.
- [11] Dianxiang Xu, Jianwen Yin, Yi Deng, and Junhua Ding. A formal architectural model for logical agent mobility. *IEEE Transactions on Software Engineering*, 29(1):31–45, 2003.

# Towards Formal Modeling and Verification of Probabilistic Connectors in Coq

Xiyue Zhang and Meng Sun

LMAM & DI, School of Mathematical Sciences, Peking University, Beijing, China  
{zhangxiyue, sunm}@pku.edu.cn

**Abstract**—The coordination language Reo has played an important role in organizing the interactions among different components in large-scale distributed applications. A probabilistic extension on classical Reo is necessary to deal with the uncertainty of the real world. In this paper we developed a framework in Coq for formalizing probabilistic connectors and reasoning about their probabilistic properties. Different types of probabilistic channels are characterized by the relations on their input and output timed data distribution streams. More complex probabilistic connectors can be further constructed based on the probabilistic channels and composition operators. Within such a framework, properties under analysis and refinement / equivalence relations between probabilistic connectors can be naturally established as theorems and proved using tactics in Coq.

**Keywords:** Reo, Probabilistic Connector, Coq, Modeling, Verification

## I. INTRODUCTION

The coordination of interactions among large numbers of concurrent entities in large-scale distributed applications cannot be easily dealt with and has become a challenge for software technology. Coordination models and languages provide a mechanism to meet this challenge by introducing a formalization of connectors that integrate a number of heterogeneous components together and organize the mutual interaction among them in a distributed environment. There are many coordination models and languages that have been proposed in the past decades, such as Reo [2], Linda [21], BIP [10], [13] and Orc [12]. Almost all of these coordination models enhance modularity and reuse of existing components and portability. However, they still differ in many dimensions: the entities being coordinated, the mechanism of coordination, the coordination medium architecture, and so on.

Reo [2], [8], as a famous coordination model, forms a paradigm for coordination of software components based on the concept of *channel*. Such channel-based models have some inherent advantages over other coordination models, especially when it comes to concurrent systems that are distributed, mobile and whose communication topologies may dynamically evolve. Channels in Reo are in fact the simplest connectors and they can be composed to construct more complex connectors that are used as the glue code to organize the interaction and communication of components in distributed applications.

The reliability of large-scale distributed applications highly depends on the correctness of coordination models, which makes the formal analysis and verification of connectors much more crucial. There are several works that have been done in

the formalization and verification of connectors in the past years: A coalgebraic semantics for Reo was developed in [5] in which connectors are interpreted as relations on timed data streams. And constraint automata (CA) was proposed as an operational model for connectors in [8]. A scheme to determine the behavior of connectors by resolving the synchronization and exclusion constraints based on connector coloring was introduced in [11]. The symbolic model checker Vereofy was developed in [7] which can be used to verify CTL-like properties for connectors. Kokash et al. presented a mapping from Reo to the specification language mCRL2 based on process algebra, where the models can be further verified conveniently using the model checker for mCRL2 in [15].

Complex distributed applications usually involve important features like real-time, probability, resource consumption, and so on. Various proposals on extending Reo to deal with such features have been reported, for example, in [3], [4], [6], [9], [17]. In this paper, we use Coq [19] to provide a formalization of Reo connectors with probabilistic behavior and show how the refinement / equivalence relations and properties of such probabilistic connectors can be further verified based on the formalization. This is a further extension to our previous work of the formalization of Reo and its timed extension in Coq [14], [22] on the probabilistic dimension, which is still based on the UTP (Unifying Theories of Programming) semantic framework for Reo that has been developed in [1], [18]. Probabilistic connectors are constituted by channels that can behave probabilistically, such as the probabilistic variant of *LossySync* channel or randomized *Sync* channel.

This is certainly not the first work to investigate probabilistic connectors. For example, probabilistic constraint automata (PCA) [6], which is a variant of CA, characterize the behavior of probabilistic connectors. However, the formalization by means of CA (and its extensions) is generally constrained by the memory limitation problem since infinite behavior is usually considered for Reo connectors. Modeling and verifying unbounded primitives or even bounded primitives with unbounded data domains, which always leads to the state space explosion problem, cannot be achieved with such finite CA-like models. But they can be specified and verified efficiently in theorem provers like Coq. The previous work in [22], [14] can certainly model a wide range of scenarios, but it is not good at dealing with the uncertainty of the real world. With this formalization for the probabilistic extension of Reo provided, more scenarios with uncertainty can be captured, and various properties under analysis or relations between such probabilistic connectors can be further verified in Coq.

The paper is organized as follows. After this general



introduction, we briefly review some main concepts of the coordination language Reo in Section II. In Section III, we present the basic specification for timed data distribution sequences and some auxiliary functions and predicates for more concise modeling. Section IV introduces the formal modeling of basic probabilistic channels and an adaptive deformation of the specification for other primitive untimed and timed channels, as well as the composition operators. Section V shows how to reason about refinement / equivalence relations between probabilistic connectors in our framework. Finally, Section VI concludes the paper and discusses some future work. The complete implementation in Coq can be found at [20].

## II. PRELIMINARY

In this section, we briefly introduce some basic concepts of the coordination language Reo. Complex coordinators, called connectors in Reo, are compositionally built out of simpler ones. We only review the primary concepts of Reo here. More details can be found in [2], [8].



Fig. 1. Basic channels in Reo

The focus in Reo is on connectors and their composition, not the different entities being connected by the connectors. It works very well in practice for controlling and organizing the communication, synchronization and cooperation among the distributed components. Each channel in Reo has exactly two channel ends with their own identifiers. There are two types of channel ends: *source end* and *sink end*. Data items are accepted into a channel through its source end and dispensed out of the channel through its sink end. It is not necessary for a channel to have both source end and sink end, i.e., a channel can have two source ends or two sink ends. Each channel end can be connected to at most one component instance at any given time. Reo allows an open-ended set of user-defined channel types as primitives for constructing connectors. Figure 1 shows some widely-used channel types in Reo which are interpreted as follows:

**Sync:** The *synchronous* channel has one source end and one sink end. A channel is called synchronous because it accepts a data item if and only if the dispensation of the data item through the sink end can simultaneously occur.

**LossySync:** The *lossy synchronous channel* is similar to the *Sync* channel except that it always accepts all data items through its source end, but it can only deliver some of the data items that it accepts, and lose the rest. Data items are transferred successfully only when the write operation on the source end and the take operation on the sink end occur simultaneously, otherwise the data items are lost.

**FIFO1:** A *FIFO1* channel is an asynchronous channel with a buffer whose capacity is bounded with 1. Initially, the buffer is empty. After accepting a data item through the source end, the data item will be kept in the buffer before being dispensed out of the channel. The next data item can only be accepted into the buffer after the data item in the buffer is dispensed.

**SyncDrain:** The *synchronous drain* has two source ends and no sink end. The pair of write operations on its two ends to accept data items can succeed only simultaneously, and all the data items written to this channel are lost.

**t-Timer:** The *t-Timer* channel is used to capture more time-related behavior. It accepts any data item through its source end and produces a *timeout* signal after a delay of  $t$  time units on its sink end.

Apart from these channel types, more well-defined exotic channels can be found in [2], [3], [18]. Users can also define new channels according to their own demands and interaction policies. For example, several probabilistic and stochastic extensions of Reo have been proposed in [6], [9], [16].

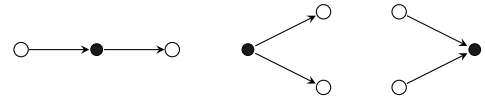


Fig. 2. Operators for channel composition

A connector is actually a set of channel ends together with the connecting channels, organized in a graph of nodes and edges, in which different channel ends coincide on each node and each edge captures the type of channel linking its two nodes. Nodes are categorized into source, sink and mixed nodes depending on whether all channel ends that coincide on a node are source ends, sink ends or a combination of the two types. According to the node types, we have three types of operators for channel / connector composition: *flow-through*, *replicate* and *merge*, as shown in Figure 2.

A source node corresponds to the replicate operator. Data items flowing through the source node are replicated and written into all the channels /connectors that are associated with the the source node. A sink node acts as the merge operator. A take / read operation on the sink node succeeds only when at least one of the channel ends coincident on the sink node offers a data item. If more than one channel ends offer data items, then one of them is selected non-deterministically. The flow-through operator simply allows the data item to flow through the mixed node without any change. When there are more than one sink ends and source ends coincident on the mixed node, the merge operator acts first, takes a data item offered by one of the sink ends, and then the selected data item flows through the mixed node, then the data item is replicated and the copies are written into all of its source ends.

## III. BASIC DEFINITIONS

An obvious way to represent a connector is to model it as a relation on its inputs and outputs, which take place through the source and sink nodes of the connector. Taking probabilistic behavior into consideration, sequences of data distributions in which the data pass through a node together with the moments in time when the data items are observed emerge as the key building blocks for describing connectors. Therefore, we can naturally specify the observations as timed data distribution sequences on nodes, which are used to characterize probabilistic connectors. Some auxiliary functions and predicates are introduced for concise modeling and reasoning about probabilistic connectors in Coq.

The main libraries of Coq being used here are the *Stream* library, the *Reals* library, and the *Utheory* library. The *Stream* library provides an appropriate co-inductive definition of infinite sequences on the input and output nodes of connectors. However, unlike the specification we used to describe the behavior of primitive channels in [14], [22], the observation sequences are adjusted here to timed data distribution streams instead of timed data streams. The *Reals* library is mainly used to support continuous time behavior; as a result, various operations and axioms on real numbers can be adopted directly for the modeling and reasoning. The *Utheory* library axiomatizes the properties required on the abstract type  $U$  representing the real interval  $[0,1]$ , which facilitates the description of probabilistic behavior. The definition of the infinite data flow can be specified as follows, with the help of these libraries.

```
Definition Time := R.
Definition Data := nat.
Definition DataDist := Data * U.
Definition TDD := Time * DataDist.
```

In this framework, time is specified as real numbers, which is very natural and expressive enough for the modeling approach. Data, for simplicity, is defined as the set of natural numbers. Generally, probability functions that map from natural numbers to the corresponding probabilities are very common and other abstract sets of data can be processed first by mapping them to a set of natural numbers in an appropriate way. Moreover, the definition of data can be expanded easily in accordance with concrete application domains. The representation of the data distribution, denoted as *DataDist*, is the Cartesian product of the type *Data* and the abstract type  $U$ , namely the real interval  $[0,1]$ . The timed data distribution *TDD* can be further defined as the Cartesian product of time and data distribution. *Stream TDD* represents the timed data distribution sequences efficiently via the *Stream* module.

Several auxiliary functions and predicates are used to facilitate the representation of channels and composition operators, and can be exploited and extended for further reasoning.

Functions *PrL* and *PrR* take an instance  $(a,b)$  of the Cartesian product type as the argument and return the first or second element of the pair, respectively. If  $b$  in this instance is still a Cartesian product  $(c,d)$ , functions *LPrR* and *DPrR* can be applied on  $(a, (c,d))$  and return the elements  $c$  and  $d$ , respectively.

The following axiom which specifies that the elements of all time streams should be in a strictly monotonous increasing order is a general requirement. More requirements that some streams should satisfy can be specified in a similar way, and some of them are provided in the definitions of specific channels.

```
Axiom Inc_T:forall (T:Stream TDD) (n:nat),
  PrL(Str_nth n T) < PrL(Str_nth n (tl T)).
```

Several predicates about time and data are defined for a more clear and concise formalization of the connectors. For example, *Tlt* (*Tgt*) represents that each element of the first time stream is strictly less (greater) than the second stream. For the modeling and verification of probabilistic connectors including timer channels, some predicates about time but with an extra

parameter  $t$  are defined in a similar way. Each element of the first time stream is added by a delay of  $t$  time units which can be customized. Therefore, *Tltt* (*Tgtt*) represents that the time of the first stream with an addition of  $t$  is less (greater) than the second stream accordingly. Complete definitions of such auxiliary functions and predicates can be found at [20].

#### IV. PROBABILISTIC CHANNELS AND OPERATORS

Modeling of basic probabilistic channels and composition operators serves as the basis of the whole modeling and reasoning framework.

##### A. Probabilistic Channels

Constraints on input and output timed data distribution streams are used in Coq to specify primitive channels' behavior. The specification of channels with probabilistic behavior can be captured by the disjunction or conjunction of different predicates about time and data distributions as well. In this section we consider four types of probabilistic channels: *message-corrupting synchronous channel*, *randomized synchronous channel*, *probabilistic lossy synchronous channel* and *faulty FIFO channel*. Specifications of other primitive channels are omitted here and can be found at [20].

**CptSync:** The message-corrupting synchronous channel  $-p \rightarrow$  is a synchronous channel which has an extra parameter  $p$  compared with the primitive synchronous channel. The delivered message can be corrupted with probability  $p$ . Hence, if a data item flows into the channel through the source end, then the correct data value will be obtained at the sink end with probability  $1-p$  and a corrupted data value  $\perp$  will be obtained with probability  $p$ . The corrupted data value is represented by the initial letter  $c$  in the Coq specification.

```
Parameter CptSync:
  Stream TDD -> Stream TDD -> U -> Prop.
Axiom CptSync_coind:
  forall (Input Output:Stream TDD) (p:U),
  CptSync Input Output p ->
  ( PrL(hd Output) = PrL (hd Input)
  /\
  (
    PrR(hd Output) =
    (c, p*(DPrR(hd Input)))
  \/
  PrR(hd Output) =
  (LPrR(hd Input), ([1-]p)*(DPrR(hd Input)))
  )
  /\
  CptSync (tl Input) (tl Output) p
  ).
```

The *CptSync* channel is defined recursively here. The first predicate of the conjunction is the constraint on the equality of time reflecting the synchronous behavior. The disjunction formula in the middle captures the probabilistic behavior. The data of the output can be the exact value of the input with the updated probability, i.e., the original companion probability multiplied by  $1-p$  or the corrupted value with probability  $p$ .

**RdmSync:** The randomized synchronous channel  $\xrightarrow{rand(0,1)}$  can generate a random number  $b \in \{0,1\}$  with equal probability

when it is activated through an arbitrary write operation on its source end, and this random number will be taken on the sink end synchronously.

```

Definition RdmSync(Input Output:Stream TDD)
:Prop :=
(forall n:nat,
PrR (Str_nth n Output)=(1%nat, [1/]1+1)
\ /
PrR (Str_nth n Output)=(0, [1/]1+1))
/\ Teq Input Output.

```

The formula in the first disjunction branch with the universal quantifier properly describes the probabilistic behavior observed on output streams. Each element of the output data distribution stream can be 1 or 0 both with the probability  $\frac{1}{2}$ . As for the constraint about time, the predicate *Teq* is used here to indicate that the time dimension of input and output streams are equal, conforming to the synchronous behavior.

**ProbLossy:** The message transmitted by the probabilistic lossy synchronous channel  $\xrightarrow{q}$  can get lost with a certain probability  $q$ . It can also act like a *Sync* channel and the message will be delivered successfully with probability  $1 - q$ .

```

Parameter ProbLossy:
Stream TDD -> Stream TDD -> U -> Prop.
Axiom ProbLossy_coind:
forall (Input Output:Stream TDD) (q:U),
ProbLossy Input Output q ->
(
(PrL(hd Output) = PrL(hd Input)
/\
PrR(hd Output) =
(LPrR(hd Input), DPrR(hd Input)*([1-]q))
/\
ProbLossy (tl Input) (tl Output) q)
\ /
ProbLossy (tl Input) Output q
).

```

The *ProbLossy* channel is defined recursively but may take two different courses in each step. The data can be totally lost when going through the channel, which leads to the recursive behavior that the last formula reflects in the specification. If the data item is successfully delivered, then there are three constraints that need to be satisfied. These constraints are represented by the conjunction of three formulas. The first formula is specified for the equality of time in accordance with the synchronous delivery. The second formula reflects that the current data item is transmitted successfully with an extra multiplication  $1 - q$  to the original probability of the input. The third formula is the second course that the recursion takes when last data item has a successful transmission.

**FtyFIFO1:** The messages flowing into a faulty FIFO1 channel  $\xrightarrow{r}\square\rightarrow$  can get lost with probability  $r$  when it is inserted into the buffer. In this case, the buffer remains empty. It can also behave as a normal *FIFO1* channel when the insertion of data into the buffer is successful with probability  $1 - r$ .

```

Parameter FtyFIFO1:
Stream TDD -> Stream TDD -> U -> Prop.
Axiom FtyFIFO1_coind:

```

```

forall (Input Output:Stream TDD) (r:U),
FtyFIFO1 Input Output r ->
(
(PrL(hd Output) > PrL(hd Input)
/\
PrL(hd Output) < PrL(hd (tl Input))
/\
PrR(hd Output) =
(LPrR(hd Input), DPrR(hd Input)*([1-]r))
/\
FtyFIFO1 (tl Input) (tl Output) r)
\ /
FtyFIFO1 (tl Input) Output r
).

```

The *FtyFIFO1* channel is also defined recursively here but can take two different ways of recursion in each step. The data written into the channel on the source end can be lost before being inserted into the buffer, which leads to the first way of recursion represented by the last formula. If the data item is successfully written into the buffer, then there are four constraints that need to be satisfied, represented by the conjunction of four formulas. The first and the second formula are used to constrain the behavior on time dimension. The time delay from input to output is captured by the first formula. Since the buffer capacity is 1, next data item cannot be written into the buffer unless the data item currently in the buffer is taken on the sink end first, which is reflected by the second formula. The third formula indicates that the current data item is transmitted successfully through the buffer with an extra  $1 - r$  being multiplied to the original probability of the input. The fourth formula reflects the second way of recursion when last data item is written into the buffer successfully.

Another kind of faulty FIFO1 channel  $\xrightarrow{r}\square\rightarrow$  works perfectly on the insertion of data item into its buffer but may lose messages from the buffer before being taken on the sink end. The difference between this channel and *FtyFIFO1* is that the loss of data items happens in different steps. Loss behavior in this channel arises in the process of being taken from the buffer, while loss behavior in *FtyFIFO1* arises in the process of storage into the buffer. But in our modeling framework, channels are specified only by the relations between observations on input and output channel ends. As a result, the specifications of these two faulty FIFO1 channels in Coq are exactly same.

The specification of primitive untimed and timed channels in [22], [14] are properly adjusted in this modeling framework. Moreover, this new formalization is still consistent with the untimed / timed version by means of assigning the value 1 to the companion probability of the data in the definitions of channels with no probabilistic behavior. Hence, the observations on input and output are all specified by timed data distribution streams, and connectors composed by primitive untimed / timed channels and probabilistic channels can be constructed without a hitch. Once there is a probabilistic channel in a connector, it will be taken as a probabilistic connector.

## B. Composition Operators

Composition operators are the other essential factor for the construction of complex connectors. As described in Section II, there are three kinds of composition operators: *flow-through*, *replicate* and *merge*.

The *flow-through* and *replicate* operators do not need to be adjusted. The specification for these operators in [22], [14] can be adopted here without any change, since the behavior of these two operators are independent of the form or content of the data flow. Both of them can still be specified implicitly by means of renaming. For example, for two channels  $ProbLossy(A, B)$  and  $FIFOI(C, D)$ , the *replicate* operator has been implemented directly by renaming  $C$  with  $A$  for the  $FIFOI$  channel. The *flow-through* operator can be implemented in a similar way. For example, when we illustrate channels  $ProbLossy(A, B)$  and  $FIFOI(B, C)$ , the *flow-through* operator that acts on node  $B$  has already been implemented.

The *merge* operator seems to depend on the content of the data flow. However, it is easy to understand that the comparison of time in the original specification of the operator does not need any change, since the time dimension is exactly same between the timed data streams and timed data distribution streams. As for the data dimension, the equality also does not need to change with the aid of the *Utheory* library. But in this framework, the equality relation for data is changed to the equality relation on data distribution. Therefore, both data items and their companied probabilities should be equal.

```
Parameter merge:
Stream TDD->Stream TDD->Stream TDD->Prop.
Axiom merge_coind:
  forall s1 s2 s3:Stream TDD,
  merge s1 s2 s3->
  ~ (PrL(hd s1) = PrL(hd s2)) /\
  ((PrL(hd s1) < PrL(hd s2)) ->
  (hd s3=hd s1)/\merge (tl s1) s2 (tl s3))
  /\
  ((PrL(hd s1) > PrL(hd s2)) ->
  (hd s3=hd s2)/\merge s1 (tl s2) (tl s3)).
```

## V. REASONING ABOUT RELATIONS

The formalization of all types of basic channels and composition operators completes the ground modeling framework, which serves well to construct different probabilistic connectors that users are interested in. Complex connectors can be constructed according to their topological orders. As soon as the construction is done, connector properties and refinement / equivalence relations between connectors can be further specified and reasoned about in Coq.

The concept of refinement has been widely used in different system descriptions. The refinement relation for connectors is defined in [18], in which the refinement order over connectors is established based on the implication relation of predicates. Connector  $C_2$  is a refinement of connector  $C_1$ , both represented by a set of predicates, if and only if  $C_2 \rightarrow C_1$ , i.e., the behavioral properties of  $C_1$  can be derived from the properties of  $C_2$ . In this case, the properties of connector  $C_2$  are regarded as the hypothesis and the properties of connector  $C_1$  as the conclusion. The refinement relation between  $C_1$  and  $C_2$  is denoted as  $C_1 \sqsubseteq C_2$  and the equivalence relation is defined typically by mutual refinement, i.e.,  $C_1 \equiv C_2$  iff  $C_1 \sqsubseteq C_2 \wedge C_2 \sqsubseteq C_1$ . Thus the equivalence relation can be represented by the implications in both directions  $C_2 \leftrightarrow C_1$ .

Fig.3 shows two probabilistic connectors both are built from the same set of five channels  $RdmSync$ ,  $FIFOI$ ,  $t$ -Timer,

$SyncDrain$  and  $Sync$ , but in different topological orders. In fact, the four channels  $FIFOI$ ,  $t$ -Timer,  $SyncDrain$  and  $Sync$  make up a timed connector  $tFIFOI$  that we have studied in [14]. Different from the primitive  $FIFOI$  channel whose output timed data distribution streams will be of the same data distribution as the input, but with an arbitrary time delay, the time delay of  $tFIFOI$  channel is fixed by the parameter  $t$ , apart from the same data distribution between input and output streams. Therefore, connectors  $R_1$  and  $R_2$  are actually constituted by the same two subconnectors  $RdmSync$  and  $tFIFOI$  but with interchanged positions. In general, two connectors composed with same set of subconnectors as we call, in commutative orders are not equivalent, i.e., the construction of connectors does not satisfy the commutative law. But in this case  $R_1$  and  $R_2$  are equal. The equivalence relation between these two connectors has been proved in Coq.

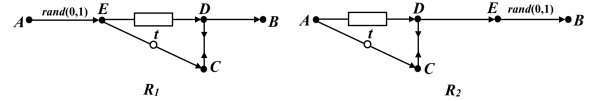


Fig. 3. Equivalence between connectors

The configurations of both  $R_1$  and  $R_2$  can be reduced to the constitution of a  $RdmSync$  channel and a  $tFIFOI$  subconnector with interchanged topological orders. Therefore, the equivalence relations between the construction from basic channels and the reduced method of construction from a  $RdmSync$  channel and a  $tFIFOI$  subconnector are proved first as lemmas in Coq, to make the subsequent proof process simplified and easier to understand.

```
Lemma RSync_tFIFO_eq:
  forall (A B:Stream TDD) (t:Time),
  exists E: Stream TDD,
  (RdmSync A E) /\ (t_FIFOI E B t)
  <->
  (RdmSync A E) /\ (exists (D C:Stream TDD),
  (FIFOI E D) /\ (SyncDrain D C)
  /\ (Timert E C t) /\ (Sync D B)).
```

The above lemma shows the equivalence relation between the reduced construction and the construction from basic channels for  $R_1$ . Another lemma is defined for  $R_2$  in Coq similarly. As a result, the goal of equivalence relation between connectors  $R_1$  and  $R_2$  boils down to the following theorem:

```
Theorem equivalence:
  forall (A B:Stream TDD) (t:Time),
  (exists E, (RdmSync A E) /\ (t_FIFOI E B t))
  <->
  (exists R, (t_FIFOI A R t) /\ (RdmSync R B)).
```

Intuitively, the core of the proof for the goal is to find the corresponding mediated timed data distribution stream to complete the construction, given the construction method of the other connector. However, a matched timed data distribution stream cannot be found directly. Therefore, we need to construct two timed data distribution streams first and then prove that they serve well as precise matches for the refinement relations in both directions, respectively. The two streams  $A_t$  and  $B_t$  are constructed to satisfy the following properties which act as hypotheses in Coq:

Hypothesis  $A\_R\_t$ :  $\text{Deq } A \ A\_t \wedge \text{Teqt } A \ A\_t \ t$ .  
Hypothesis  $B\_R\_t$ :  $\text{Deq } B \ B\_t \wedge \text{Teqt } B \ B\_t \ t$ .

There are other hypotheses being used directly in the proof, such as *transfer\_eqt* which has been proved in [14]. Some new properties that aid in proving the current goal but not proved before are formalized as lemmas and get proved first. For example, the following lemma demonstrates a property that two streams which are both greater than a same stream by  $t$  are equal in the time dimension. It is simple to formalize and prove this property using some commonly-used tactics like *intro*, *destruct* and *rewrite*.

```
Lemma trans_t_eq:
  forall (s1 s2 s3:Stream TDD) (t:Time),
    (Teqt s1 s2 t) /\ (Teqt s1 s3 t) ->
      (Teqt s2 s3).
```

All these lemmas and hypotheses make the main proof more concise. The proof of the theorem actually has two steps. The original goal is *split* first into two subgoals that represent refinement relations in both directions. For the first subgoal, the antecedent or the precondition of the implication serves as another hypothesis. After asserting that the matched timed data distribution stream  $R$  is  $A\_t$ , the current subgoal is reduced to  $t\_FIFO1 \ A \ A\_t \ t \wedge RdmSync \ A \ B$ , which can be *split* again and proved using specific tactics based on the lemmas and hypotheses, especially the fact that *exists E:Stream TDD, RdmSync A E  $\wedge$  t\\_FIFO1 E B t*. The subgoal of the refinement relation in the other direction can be proved similarly. A complete proof process can be found at [20].

## VI. CONCLUSION

This paper proposes a method of modeling and reasoning about probabilistic connectors in Coq, which is also compatible with the formalization for the primitive untimed / timed connectors. Basic probabilistic channels and composition operators are formalized as the ground framework. After adjusting timed data streams to timed data distribution streams, all the channels can be specified by a set of predicates capturing the relations between inputs and outputs. The formalization of composition operators makes it possible to construct more complex connectors. Properties related to probability distributions and refinement / equivalence relations between probabilistic connectors can be specified easily and further presented with machine-checked proofs. Compared to LTL or CTL formulas, Coq expressions are more powerful to depict properties and we can be free from worrying about the state space explosion problems in other verification approaches like model checking. Moreover, this architecture is promising to capture the uncertainty of different applications in real world.

In the future, we plan to investigate some more scenarios related to coordination in real world based on this architecture. In particular, we would like to deal with more probabilistic properties users care about among different applications or services. The modeling and verification of hybrid behavior of connectors in Coq is in our scope as well.

## ACKNOWLEDGEMENTS

The work is partially supported by NSFC under grant no. 61772038, 61532019, 61202069 and 61272160.

## REFERENCES

- [1] B. K. Aichernig, F. Arbab, L. Astefanoaei, F. S. de Boer, S. Meng, and J. Rutten. Fault-based test case generation for component connectors. In *Proceedings of TASE 2009*, pages 147–154. IEEE Computer Society, 2009.
- [2] F. Arbab. Reo: A Channel-based Coordination Model for Component Composition. *Mathematical Structures in Computer Science*, 14(3):329–366, 2004.
- [3] F. Arbab, C. Baier, F. S. de Boer, and J. J. M. M. Rutten. Models and temporal logical specifications for timed component connectors. *Software and System Modeling*, 6(1):59–82, 2007.
- [4] F. Arbab, T. Chothia, R. van der Mei, S. Meng, Y.-J. Moon, and C. Verhoef. From Coordination to Stochastic Models of QoS. In J. Field and V. T. Vasconcelos, editors, *Proceedings of Coordination'09*, volume 5521 of *LNCS*, pages 268–287. Springer, 2009.
- [5] F. Arbab and J. Rutten. A coinductive calculus of component connectors. In M. Wirsing, D. Pattinson, and R. Hennicker, editors, *WADT 2002*, volume 2755 of *LNCS*, pages 34–55. Springer-Verlag, 2003.
- [6] C. Baier. Probabilistic Models for Reo Connector Circuits. *Journal of Universal Computer Science*, 11(10):1718–1748, 2005.
- [7] C. Baier, T. Blechmann, J. Klein, S. Klüppelholz, and W. Leister. Design and verification of systems with exogenous coordination using vereofy. In *Proceedings of ISO LA 2010*, volume 6416 of *LNCS*, pages 97–111. Springer, 2010.
- [8] C. Baier, M. Sirjani, F. Arbab, and J. Rutten. Modeling component connectors in Reo by constraint automata. *Science of Computer Programming*, 61:75–113, 2006.
- [9] C. Baier and V. Wolf. Stochastic Reasoning About Channel-Based Component Connectors. In P. Ciancarini and H. Wiklicky, editor, *COORDINATION 2006*, volume 4038 of *LNCS*, pages 1–15. Springer-Verlag, 2006.
- [10] S. Bliudze and J. Sifakis. The algebra of connectors - structuring interaction in BIP. *IEEE Trans. Computers*, 57(10):1315–1330, 2008.
- [11] D. Clarke, D. Costa, and F. Arbab. Connector colouring I: Synchronisation and context dependency. *Science of Computer Programming*, 66:205–225, 2007.
- [12] W. R. Cook, S. Patwardhan, and J. Misra. Workflow Patterns in Orc. In *Proceedings of COORDINATION 2006*, volume 4038 of *LNCS*, pages 82–96. Springer, 2006.
- [13] R. Edelmann, S. Bliudze, and J. Sifakis. Functional BIP: embedding connectors in functional programming languages. *Journal of Logical and Algebraic Methods in Programming*, 92:19–44, 2017.
- [14] W. Hong, S. Nawaz, X. Zhang, Y. Li, and M. Sun. Using Coq for Formal Modeling and Verification of Timed Connectors. In *Software Engineering and Formal Methods: SEFM 2017 Collocated Workshops, Revised Selected Papers*, volume 10729 of *LNCS*, pages 558–573. Springer, 2018.
- [15] N. Kokash, C. Krause, and E. de Vink. Reo+mCRL2: A framework for model-checking dataflow in service compositions. *Formal Aspects of Computing*, 24:187–216, 2012.
- [16] Y. Li, X. Zhang, Y. Ji, and M. Sun. Capturing Stochastic and Real-time Behavior in Reo Connectors. In *Proceedings of SBMF 2017*, volume 10623 of *LNCS*, pages 287–304. Springer, 2017.
- [17] M. Sun and F. Arbab. On resource-sensitive timed component connectors. In *Proceedings of FMOODS 2007*, volume 4468 of *LNCS*, pages 301–316. Springer, 2007.
- [18] M. Sun, F. Arbab, B. K. Aichernig, L. Astefanoaei, F. S. de Boer, and J. Rutten. Connectors as designs: Modeling, refinement and test case generation. *Science of Computer Programming*, 77(7-8):799–822, 2012.
- [19] The Coq Proof Assistant. <https://coq.inria.fr/>.
- [20] The source code. <https://github.com/Xiyue-Selina/Prob-Reo>.
- [21] M. Viroli, D. Pianini, and J. Beal. Linda in space-time: An adaptive coordination model for mobile ad-hoc environments. In *Proceedings of COORDINATION 2012*, volume 7274 of *LNCS*, pages 212–229. Springer, 2012.
- [22] X. Zhang, W. Hong, Y. Li, and M. Sun. Reasoning about Connectors in Coq. In *Proceedings of FACS 2016*, volume 10231 of *LNCS*, pages 172–190. Springer, 2017.

# Reo2PVS: Formal Specification and Verification of Component Connectors

M. Saqib Nawaz and Meng Sun

LMAM & Department of Informatics, School of Mathematical Sciences, Peking University, Beijing, China  
{msaqibnawaz, sunm}@pku.edu.cn

**Abstract**—Compositional coordination models such as Reo provide powerful support for the development of large-scale distributed systems by allowing construction of complex connectors that coordinate behavior among different components. The reliability of such distributed systems highly depends on the correctness of connectors. In this paper, we use the proof assistant PVS for formal modeling, analysis and verification of component connectors. We first present the modeling of primitive channels and the composition operators that are used to combine channels for building complex connectors. Furthermore, we show how to model and analyze connector’s behavior in PVS and prove some interesting connector properties. The model reflects the original topological structure of connectors simply and clearly. With the provided approach, different kinds of connector properties can be naturally formalized and proved in PVS.

**Index Terms**—Reo, Connector, PVS, UTP, Design

## I. INTRODUCTION

Nowadays, most modern software systems are distributed over large networks of computing devices. However, software components that comprise the whole system usually do not fit together exactly and leave significant interfacing gaps among them. Such gaps are generally filled with additional “*glue code*”. Compositional coordination languages offer such a glue code among components and facilitate the mutual interactions between components in a distributed environment. Reo [2] and Linda [11] are two popular examples of such compositional coordination languages, which have played an important role in the success of component-based systems in the past decades.

Reo is a channel-based exogenous coordination language where complex component connectors are orchestrated from channels via certain composition operators. Exogenous coordination [1] means coordination from outside, where the primitives that support the coordination of an entity with others reside outside of that entity. Connectors in Reo provide the protocols that control and organize the communication, synchronization and cooperation among the components that they interconnect. Despite its simplicity, Reo has been used successfully in various application domains, such as service-oriented computing [10], [21], [22], business processes [25] or biological systems [8].

The reliability of component-based systems highly depend on the correctness of connectors. Formal analysis and verification of connectors is gaining more interest in recent years with the evolution of software systems and advancements in Cloud

and Grid computing technologies. Furthermore, the increasing growth in size and complexity of computing infrastructure has made the modeling and verification of connector properties a more difficult and challenging task. From the modeling and analysis context, the formal semantics for Reo allows us to specify and analyze the behavior of connectors precisely. In literature, different formal semantics have been proposed for Reo [13], such as the co-algebraic semantics in terms of relations on infinite timed data streams [3], operational semantics using constraint automata [6], the coloring semantics by coloring a connector with possible data flows [9] in order to resolve synchronization and exclusion constraints, and the UTP (Unified Theories of Programming) semantics [20], [23].

In the past decade, a lot of work has been done towards formal verification and analysis of Reo connectors. A symbolic model checker “Vereofy” has been developed in [5] to check the CTL-like properties of systems with exogenous coordination. Another approach is to take advantage of existing verification tools by translating Reo model to other formal models such as Alloy [14], mCRL2 [15], etc. Since infinite behavior is usually taken into consideration for connectors, the modeling and analysis of connectors are expected to be achieved efficiently in theorem provers. In [16], a method for formal modeling and verification of Reo connectors in Coq is provided. Reo connectors were represented in a constructive way and verification was based on the simulation of the behavior and output of Reo connectors. A different modeling and analysis framework in Coq was proposed in [24], which adopted the UTP design model for Reo connectors developed in [20], [23], i.e., a pair of predicates  $P \vdash Q$  where the predicate  $P$  specifies what the designer can rely on when the communicating operation is initiated by input to the connector, and  $Q$  is the condition on output that must be true when the communicating operation terminates. Work done in [24] was extended in [12] to cover the modeling and verification of timed channels and connectors in Coq.

In this paper, the aim is to provide an approach for formal modeling and reasoning about Reo connectors constructed from primitive (both untimed and timed) channels under the UTP semantic framework using the proof assistant PVS [17]. We first provide the modeling for a family of primitive channels and compositional operators in PVS. Then we show how to model and reason about more complex connectors. The basic idea is to model the observable behavior of a connector as a relation on the timed data sequences being observed

at its input and output nodes. In PVS, this is achieved by representing a connector as a logical predicate that describes the relation among the timed data sequences on its input and output nodes. The model makes it possible to prove complex and generic connectors' properties easily in PVS. The PVS dump file for this work can be found at [19].

The rest of this paper is organized as follows: Reo and PVS are briefly introduced in Section II. PVS specifications for basic definitions being used in the modeling of primitive channels are presented in Section III. In Section IV, formal modeling of primitive channels is described, followed by the modeling of compositional operators. Section V shows how to verify and reason about connector properties in PVS by several examples. Finally, Section VI concludes the paper.

## II. PRELIMINARIES

In this section, a brief introduction to the coordination language Reo and the PVS system is provided.

### A. Reo

Reo is a channel-based exogenous coordination language where complex *connectors* are compositionally constructed out of simpler ones. Further details on Reo can be found in [2], [6]. Connectors provide the protocol to control and organize the communication, synchronization and cooperation among different components. The simplest connectors are channels with well-defined behavior. Each Reo channel has two channel ends, which can be of type *source* or *sink*. A source channel end accepts data into the channel and a sink channel end dispenses data out of the channel. Few primitive channel types in Reo are shown in Figure 1.



Figure 1. Some primitive channels in Reo

A *synchronous (Sync) channel* has one source and one sink end. I/O operations can succeed only if the writing operation at source end is synchronized with the read operation at its sink end. A *lossy synchronous (LossySync) channel* is a variant of synchronous channel that accepts all data through its source end. The written data is lost immediately if no corresponding read operation is available at its sink end. A *FIFO1 channel* is an asynchronous channel with one buffer cell, one source end and one sink end. The channel accepts a data item whenever the buffer is empty. The data item is kept in the buffer and dispensed to the sink end later in the FIFO order. A *synchronous drain (SyncDrain) channel* has two source ends and no sink end, which means that no data can be obtained from such channels. The write operation on both source ends should happen simultaneously and the data items written to this channel are irrelevant. A *t-timer channel* accepts any data item at its source end and produces a *timeout* signal on its sink end after a delay of  $t$  time units.

Complex connectors are constructed by composition of different channels with *join* and *hiding* operations. The result

can be represented visually as a graph where a node represents a set of channel ends that are combined together through the join operation, while the edges in the graph represent the channels between the corresponding nodes. Nodes are categorized into *source*, *sink* or *mixed nodes*, depending on whether the node contains only source channel ends, sink channel ends, or both. Source nodes are analogous to input ports, sink nodes to output ports and mixed nodes are internal details of a connector that are hidden. The internal topology of any connector can be hidden from outside by applying the hiding operation. The behavior of a connector can be captured by the data-flow on its source and sink nodes. The hidden nodes can not be accessed or observed from outside.

### B. PVS

PVS (Prototype Verification System) offers a formal specification language and a mechanical theorem proving environment. The PVS system consists of a specification language, a parser, a type-checker, a prover, specification libraries, and various browsing tools. Specification language of PVS is build on a higher order logic and type system of PVS supports predicate sub-typing and other type dependencies. The type system of PVS is not algorithmically decidable and theorem proving may be required to establish the type-consistency of a PVS specification. Theorems that need to be proved are called type-correctness conditions (TCC's). Here, we give a simple example for factorial function that is defined recursively in PVS.

```
factorial(n:nat): RECURSIVE posnat =
  IF n = 0 THEN 1 ELSE factorial(n-1)*n ENDIF
MEASURE n
```

```
the: THEOREM FORALL(k:nat): factorial(k) >= k
```

In PVS, recursive definitions are treated as constant declarations and it must be total, so that the function is defined for every value of its domain. In order to ensure this, recursive functions must be specified with a measure ( $n$  in the factorial function). Theorem *the* in this example shows that factorial of a number should be greater than or equal to that number. PVS offers inference rules, proof commands and decision procedures that can be used to prove theorems. PVS prover is based on *sequent calculus* where each proof goal is a *sequent* consisting of a sequence of formulas called *antecedents* and a sequence of formulas called *consequents*. The intuitive interpretation of a sequent is that the *conjunction* of the antecedents implies the *disjunction* of the consequents. During proof construction, PVS builds a graphical proof tree in which remaining proof obligations are at the leaves of tree. If a proof gets stuck, then this tree helps to see where the proof goes wrong. Further details on PVS can be found in [18].

## III. BASIC DEFINITIONS IN PVS

The behavior of a connector can be formalized by means of data-flows on its sink and source nodes which are essentially infinite sequences. In PVS, record structure is used to represent *timed data (TD)* sequences on the sink and source nodes,

where *time* is defined as *positive real numbers* ( $\mathbb{R}^+$ ) and *data* is defined as a *positive* type. The advantage of using the record structure for representing a TD sequence is that it offers names for both time and data of the sequence, which makes the specification more convenient and understandable.

```
Time: Type = posreal
Data: TYPE+
TD: TYPE = [# T: sequence[Time],
            D: sequence[Data] #]
Input, Output: VAR TD
```

A TD is a record structure type that has two components: *T* and *D*. The *D* component is a sequence of *data items*. The *T* component is a sequence of *time points* being used to represent the *time* when the data items in the *D* component being observed. *Input* and *Output* are declared as variables of type TD. The following predicates are used for primitive channels specification later:

```
Teq(Input,Output):bool = T(Input) = T(Output)
Tle(Input,Output):bool = T(Input) < T(Output)
Tgt(Input,Output):bool = T(Input) > T(Output)
Deq(Input,Output):bool = D(Input) = D(Output)
```

*Teq* takes two TD sequences and returns *true* if the time of two sequences are exactly equal to each other. *Tle* represents that time of the first sequence is strictly less than the second sequence and *Tgt* means that the time of the first sequence is strictly greater than the second sequence. *Deq* shows the equality of data: data sequence at *Input* is equal to data sequence at *Output*. *Teq*, *Tle* and *Tgt* only checks the time component of the record structure, whereas, *Deq* checks the data field. Since the type of component *T* in TD is defined as *sequence[Time]*, we have to define the operators “<” and “>” for sequences of times. A strict order (that is both transitive and irreflexive) is assumed for “<” and “>”.

```
<: (strict_order?[sequence[Time]])
>: (strict_order?[sequence[Time]]) =
  LAMBDA (s1, s2: sequence[Time]): s2 < s1
```

Defining “<,>” for sequence of time generated two TCC’s. Proof steps for these two TCC’s can be found at [19].

For timed channels, three new predicate formulas are introduced, which are similar as the previous definitions for primitive untimed channels with one of the time sequences is added by a *t* time delay. An extra *t* is appended to the names of these new predicates to distinguish them from the ones for untimed channels. Definitions *Teq*, *Tle* and *Tgt* can also be specified with the terms used in the *Teqt*, *Tltt* and *Tgtt*.

```
Teqt(T1,T2) (t:Time): bool = FORALL (n:nat):
  FrS(str_nth(n,T1)) + t = FrS(str_nth(n,T2))

Tltt(T1,T2) (t:Time): bool = FORALL (n:nat):
  FrS(str_nth(n,T1)) + t < FrS(str_nth(n,T2))

Tgtt(T1,T2) (t:Time): bool = FORALL (n:nat):
  FrS(str_nth(n,T1)) + t > FrS(str_nth(n,T2))
```

## IV. REO CHANNELS AND OPERATORS

The modeling of primitive untimed / timed channels and operators for connectors composition is presented in this section. These channels and operators are used later in the modeling and analysis of complex connectors.

### A. Primitive Channels

For the *Sync* channel, the time and data of a sequence that flows into the channel are exactly the same as those of the sequence flowing out. The channel is modeled as follows in PVS:

```
Sync(Input,Output):bool = Teq(Input,Output) &
                          Deq(Input,Output)
```

The *SyncDrain* channel has two source ends and no sink end. It works as a synchronous channel that allows pairs of write operations pending on its two ends to succeed simultaneously. In this channel, all written data items are consumed and lost. Thus, this channel is used just for synchronizing two TD sequences being observed on its two ends. This channel is specified in PVS as follows:

```
SyncD(Input1,Input2):bool= Teq(Input1,Input2)
```

A *LossySync* channel is analogous to the *Sync* channel, except that the write operation on the source end always succeeds immediately. If a corresponding read operation is already pending on the sink end, then the written data item is transferred to the sink end and both operations succeed. Otherwise, only the write operation on the source end succeeds and the data item is lost. *LossySync* channel is defined inductively as follows: \*

```
Lossysync(Input,Output) (n:nat):INDUCTIVE bool
= ( nth(Output,n)= nth(Input,n)
  & Lossysync(next(Input),next(Output)) (n)
  OR Lossysync(next(Input),Output) (n))
```

Another important channel in Reo is the asynchronous one with buffering capacity 1, known as *FIFOI* channel ( $-\square\rightarrow$ ). The time when a *FIFOI* channel takes a data item at its source end is earlier than the time when the data item is delivered at its sink end. Furthermore, the time of the next data item that flows in at the source end should be later than the time when the data in the buffer is delivered at the sink end. The buffer is empty if no data item is in the buffer, and it contains a data element *d* after *d* is written through the source end and before *d* is taken out through the sink end.

```
Fifol(Input,Output):bool= Tle(Input,Output) &
                          Tle(Output,next(Input)) & Deq(Input,Output)
```

A *FIFOIe* channel is a variant of *FIFOI* where the buffer already contains a data element “*e*”. The communication can only be initiated when *e* is taken out through the sink end. So the data sequence that flows out of the channel always get an extra element *e* settled at the beginning of the sequence.

\*In PVS, inductive definitions are similar to recursive definitions, in that both involve induction and must satisfy additional constraints to guarantee that they are total.



Moreover, the time of the sequence that flows into the channel should be earlier than time of the tail of the sequence that flows out. As the buffer contains  $e$ , new data can be written into the channel only after the element  $e$  has been taken. Therefore, time of the sequence that flows out is earlier than time of the sequence that flows in.

```
Fifole(Input,Output) (e:Data) (n:nat) : bool =
  Tgt (Input,Output) & Tle (Input,next (Output)
    & e? (nth (Output,n)) (e) &
  Deq (Input, (next (Output)))
```

The  $t$ -timer channel accepts input data through its source end and returns a *timeout* signal on its sink end exactly after a duration of  $t$  time units. It is specified in PVS as follows:

```
Timert (Input,Output) (t:Time) (d:Data) : bool =
  FORALL (n:nat) : FrS (str_nth (n,Input)) + t <
    FrS (str_nth (n, (next (Input))))
  & Teqt (Input,Output) (t)
  & SrF (str_nth (n,Output)) = timeout (d)
```

The definitions of more channels can be found at [19]. Defining primitive channels by intersection and disjunction of predicates in PVS makes the modeling of channels more concise, easy to understand as each predicate describes a simple order relation (requirement) on time or data. Furthermore, we can easily split the predicates for proofs of different properties which can make the reasoning and proving process simpler.

### B. Operators Modeling in PVS

Three main composition operators (shown in Figure 2) are used in Reo for connector construction, which are (i) flow-through, (ii) replicate and (iii) merge.

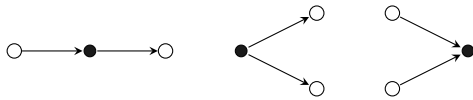


Figure 2. Operators for channel composition

The *flow-through* operator simply allows data items to pass the mixed node. It can be achieved explicitly without specifying it in PVS. This is explained with the simple example in Figure 3, which represents a flow-through operator that connects two channels  $Sync(A, B)$  and  $FIFO1(B, C)$  at node  $B$ . Such a flow-through operation at node  $B$  can be implicitly implemented by just writing the connector as " $Sync(A, B) \wedge FIFO1(B, C)$ ".



Figure 3. A connector composed of a Sync and a FIFO1 channel

The *replicate* operator puts the source ends of different channels together into one source node. Write operation on this node succeeds only if all the channels are capable of consuming a copy of the written data. Similar to the flow-through operator, it can be implicitly represented by the structure of connectors. For example, If we put one  $Sync(A, B)$  channel

and one  $FIFO1(C, D)$  channel together, we can simply write  $Sync(A, B) \wedge FIFO1(A, D)$  in PVS instead of defining a recursive or inductive function, and the replicate operator is achieved directly by renaming  $C$  with  $A$  for the  $FIFO1$  channel. The use of conjunction and node renaming for flow-through and replicate operator allows us to define connectors directly in lemmas and theorems.

The modeling of *merge* operator is a bit more complicated. When the merge operator acts on two channels, it leads to a choice of taking the data item from one of them. The merge operator is defined inductively as the intersection of two predicates.

```
Merge (s1,s2,s3) (n:nat) : INDUCTIVE bool =
  ( NOT (FrS (nth (s1,n)) = (FrS (nth (s2,n))))
  AND ((FrS (nth (s1,n)) < (FrS (nth (s2,n))))
    IMPLIES nth (s3,n) = nth (s1,n))
  & Merge (next (s1), s2, next (s3)) (n))
  AND ((FrS (nth (s1,n)) > (FrS (nth (s2,n))))
    IMPLIES nth (s3,n) = nth (s2,n))
  & Merge (s1, next (s2), next (s3)) (n)))
```

## V. REASONING ABOUT CONNECTORS

In this section, we investigate and prove some interesting properties for connectors in PVS.

**Example 1.** We first consider the connector shown in Figure 3. Let  $a, b, c$  denote the time sequences when the corresponding data sequence flows through nodes  $A, B$  and  $C$ . According to the semantics of  $Sync$  and  $FIFO1$  channels, we know that  $a = b < c$ . Let  $\alpha, \beta$  represent the data sequence being observed at the source node ( $A$ ) and the sink node ( $C$ ) respectively, we have  $\alpha = \beta$ . In PVS, these results are proved with the following theorem.

**Theorem 1.**  $Sync(A,B) \wedge Fifo1(B,C) \Rightarrow Tle(A,C) \wedge Teq(A,B) \wedge Deq(A,C)$

*Proof.* When the PVS proof checker is run on this theorem, it gives the following sequent (proof goal) which consists of no antecedent and one consequent formula:

```
|-----
{1} FORALL (A,B,C:TD) :
  Sync(A,B) & Fifo1(B,C) => Tle(A,C)
  & Teq(A,B) & Deq(A,C)
```

The “*skolem!*” command is used first that creates a fresh free *skolem* variable for the universal quantifier ( $\forall$ ) in consequent. Then the “*flatten*” command is used to simplify the proof goal by removing  $\Rightarrow$  from consequent. This changes the formula to:

```
{-1} Sync(A!1, B!1)
{-2} Fifo1(B!1, C!1)
|-----
{1} Tle(A!1,C!1) & Teq(A!1,B!1) & Deq((A!1,C!1)
```

It now consists of two antecedent formulas and one consequent formula. In next steps, the definitions of  $Sync$  and  $FIFO1$  channels, as well as predicates  $Tle, Teq, Deq$  and

*next* are expanded with the command (*expand "id"*). Conjunction ( $\&$ ) in the antecedents are removed with the command "*flatten*". The formula now simplifies to:

```

{-1} T(A!1) = T(B!1)
{-2} D(A!1) = D(B!1)
{-3} T(B!1) < T(C!1)
{-4} T(C!1) < (suffix(B!1 `T, 1))
{-5} D(B!1) = D(C!1)
|-----
[1] T(A!1) < T(C!1) & T(A!1) = T(B!1) & D(A!1) = D(C!1)

```

The sequent is then divided (with "*split*" command) into three sub-sequents (sub-goals), which are all proved with decision procedure command "*assert*".<sup>†</sup>  $\square$

**Example 2.** We now consider the Lower Bounded FIFO1 connector as given in Figure 4. This connector has one source node *A* and one sink node *B*. It ensures the lower bound " $> t$ " for the take operation on node *B*. Every data item received by this connector need to stay in its buffer for more than *t* time units. Let  $\alpha, \beta$  represents the data sequences being observed at nodes *A* and *B*, and  $a, b$  represents the time sequences corresponding to  $\alpha$  and  $\beta$ , i.e., the *i*-th element  $a(i)$  in  $a$  (and  $b(i)$  in  $b$ ) denotes exactly the time moment of the occurrence of  $\alpha(i)$  (and  $\beta(i)$ ). For this connector, it is proved in theorem 2 that  $\alpha = \beta$  and  $a + t < b$ .

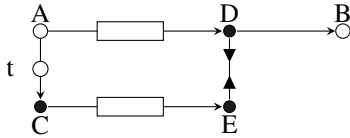


Figure 4. Lower Bounded FIFO1 Connector

**Theorem 2.**  $\forall A, B, C, D, E \in TD, t \in Time, d \in Data:$

$$Timert(A, C)(t)(d) \wedge Fifo1(A, D) \wedge SyncD(D, E) \wedge Fifo1(C, E) \wedge Sync(D, B) \Rightarrow Deq(A, B) \wedge Tltt(A, B)(t)$$

*Proof.* After applying *skolemization*, *expansion* and *flattening*, the main goal is split into two sub-goals. The first sub-goal is for the data dimension, i.e., the data sequence being received at *A* should be equal to the data sequence being taken at *B*. The *Fifo1(A, D)* channel satisfies the predicate *Deq(A, D)* and the *Sync(D, B)* channel satisfies *Deq(D, B)*. The conjunction of both predicates results in *Deq(A, B)*. For the time dimension, we have predicates *Teqt(A, C, t)*, *Tltt(C, E)*, *Teq(E, D)* and *Teq(D, B)*, which can be obtained from the definitions of *Timert*, *Fifo1*, *SyncD* and *Sync* channels respectively. These four predicates introduce the constraints  $A + t = C, C < E, E = D, D = B$  for time. The combination of these predicates results in *Tltt(A, B, t)*, such that  $A + t < B$  holds for time.  $\square$

**Example 3.** Figure 5 shows an expiring FIFO1 connector that can be constructed with a normal FIFO1 channel and a *t*-timer (and some other channels). In this connector, a data item received through the source node *A* is dropped from the

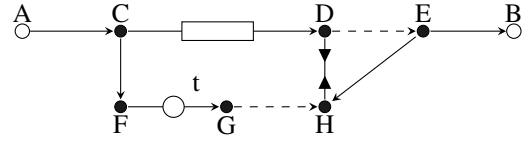


Figure 5. Expiring FIFO1 channel

buffer if it is not taken out through the sink node *B* within *t* time units.

**Theorem 3.**  $\forall A, B, C, D, E, F, G, H \in TD, t \in Time, d \in Data, n \in nat:$

$$Sync(A, C) \wedge Sync(C, F) \wedge Fifo1(C, D) \wedge Timert(F, G)(t)(d) \wedge Lossysync(G, H)(n) \wedge SyncD(D, H) \wedge Lossysync(D, E)(n) \wedge Sync(E, H) \wedge Sync(E, B) \Rightarrow Teqt(A, B)(t) \wedge Tgt(B, A)$$

*Proof.* The PVS proof process of this theorem is presented as following:

(*induct "n"*) (The proof starts by applying induction on *n*) Main goal is split into two sub-goals. For the first sub-goal (the base case):

(*skosimp*) (*expand Fifo -3*) (*expand "Lossysync"*) (*expand "Sync"*) (*expand "SyncD"*) (*expand "Timert"*) (*expand "Teq"*) (*expand "Teqt"*) (*expand "Tgt"*) (*assert*) (*split*)

The first sub-goal is split into two more sub-goals. For the first sub-sub-goal:

(*skosimp*) (*inst? -10*) (*skosimp*) (*assert*).

This proves the first sub-sub-goal.

For the second sub-sub-goal:

(*assert*)(*inst? -5*) (*skosimp*) (*assert*).

This proves the second sub-sub-goal and the proof of the first sub-goal is complete.

For the second sub-goal:

(*skosimp\**) (*inst? -1*) (*expand "Fifo1"*) (*assert*) (*expand "Teqt"*) (*expand "Tgt"*) (*skosimp*) (*assert*) (*split*)

The second sub-goal is divided into two more sub-goals. For the first sub-sub-goal:

(*skosimp*)(*typepred "t!1"*) (*inst? -3*) (*assert*) (*grind*).

This proves the first sub-sub-goal.

For the second sub-sub-goal:

(*assert*) (*typepred ">"*) (*expand "strict\_order"*) (*flatten*) (*expand "transitive"*) (*assert*) (*grind*).

This proves the second sub-sub-goal and the proof of the second sub-goal is complete.

This completes the proof of the theorem.  $\square$

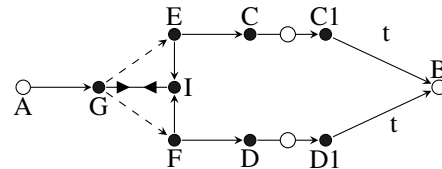


Figure 6.  $2 \times t$  Timer Connector

**Example 4.** A timed connector  $\xrightarrow{n \times t}$  can be built by using *n* *t*-timer channels and an exclusive router (with *n* sink nodes).

<sup>†</sup>Note that a theorem in PVS can be proved in different ways that depends on the proof commands, inference rules and decision procedures being used.

Such a connector produces a timeout signal after a delay  $t$  for every input it receives. The duration between the arrival time for the  $i$ -th input and that for the  $(i + j)$ -th input for  $j < n$  can be less than  $t$  whereas the duration between the arrival time for the  $i$ -th input and that for the  $(i + n)$ -th input should be at least  $t$ . Figure 6 shows the topology structure of



Let  $a, b$  represent the time sequences corresponding to the data sequences flowing into  $A$  and out of  $B$ , respectively. Theorem 4 states the property that  $a + t = b$  for the  $2 \times t$  timed connector.

**Theorem 4.**  $\forall A, B, C, D, E, F, G, I, CI, DI \in TD, t \in Time, d \in Data, n \in nat:$

$$\begin{aligned} & Sync(A, G) \wedge Lossysync(G, E)(n) \wedge Lossysync(G, F)(n) \wedge \\ & Sync(E, I) \wedge Sync(F, I) \wedge SyncD(G, I) \wedge Merge(E, F, I)(n) \wedge \\ & Sync(E, C) \wedge Sync(F, D) \wedge Timert(C, CI)(t)(d) \wedge \\ & Timert(D, DI)(t)(d) \wedge Merge(CI, DI, B)(n) \Rightarrow Teqt(A, B)(t) \end{aligned}$$

Since the *Lossysync* channel and the composition operator *merge* are both defined inductively, this theorem is proved by induction on the parameter  $n$ . The main goal is divided into two sub-goals. The first sub-goal is for the base case and the second subgoal is for the inductive step. The details of the proof process is similar to the proofs for previous theorems in this section, and we omit it here due to the length limitation.

## VI. CONCLUSION

This paper presents a method for formal modeling of Reo connectors and reasoning about Reo connectors in PVS. The formalization is based on the UTP design semantics for Reo and preserves the original structure and behavior semantics of Reo channels and composition operators, which makes their description in PVS reasonably readable. Connector properties are specified with predicates which offer an appropriate description of the relations between different timed data sequences being observed on the nodes of a connector. The proofs of connector properties are completed with the help of PVS proof-commands, inference rules and decision procedures. Generalized property for connectors for arbitrary  $n$ , which cannot be verified explicitly with model checkers, can be proved here as well.

The main problem of this approach is that the analysis and proof process of complex connector properties in PVS always requires heavy interactions between users and the proof assistant, and thus consumes a lot of time. Even for simple properties, the proof process can become hard and requires the users to have good knowledge on PVS to make the proof successfully. In the future, efforts will be made to encapsulate frequently-used proof patterns as PVS strategies in order to make the proof process easier and reduce repetitive work. Machine learning techniques are also expected to provide some help to automate the proof process and reduce the amount of human efforts. On the other hand, extension of the approach to deal with probabilistic [4] or stochastic [7] behavior of connectors is in our plan for future work as well.

**Acknowledgement.** The work was partially supported by the National Natural Science Foundation of China under grant no. 61772038, 61532019, 61202069 and 61272160.

## REFERENCES

- [1] F. Arbab. The IWIM Model for Coordination of Concurrent Activities. In *Proceedings of COORDINATION 1996*, pages 34–56, 1996.
- [2] F. Arbab. Reo: A Channel-based Coordination Model for Component Composition. *Mathematical Structures in Computer Science*, 14(3):329–366, 2004.
- [3] F. Arbab and J. Rutten. A Coinductive Calculus of Component Connectors. In *Proceedings of WADT 2002*, volume 2755 of *LNCS*, pages 34–55. Springer-Verlag, 2002.
- [4] C. Baier. Probabilistic Models for Reo Connector Circuits. *Journal of Universal Computer Science*, 11(10):1718–1748, 2005.
- [5] C. Baier, T. Blechmann, J. Klein, S. Klüppelholz, and W. Leister. Design and Verification of Systems with Exogenous Coordination using Vereofy. In *Proceedings of ISO LA 2010*, volume 6416 of *LNCS*, pages 97–111. Springer, 2010.
- [6] C. Baier, M. Sirjani, F. Arbab, and J. Rutten. Modeling Component Connectors in Reo by Constraint Automata. *Science of Computer Programming*, 61:75–113, 2006.
- [7] C. Baier and V. Wolf. Stochastic Reasoning about Channel-Based Component Connectors. In *Proceedings of COORDINATION 2006*, volume 4038 of *LNCS*, pages 1–15. Springer-Verlag, 2006.
- [8] D. Clarke, D. Costa, and F. Arbab. Modelling Coordination in Biological Systems. In *Proceedings of ISO LA'04*, volume 4313 of *LNCS*, pages 9–25. Springer, 2004.
- [9] D. Clarke, D. Costa, and F. Arbab. Connector Coloring I: Synchronization and Context Dependency. *Science of Computer Programming*, 66(3):205–225, 2007.
- [10] N. Diakov and F. Arbab. Compositional Construction of Web Services using Reo. In *Proceedings of ICEIS 2004*, pages 13–14, 2004.
- [11] D. Gelernter and N. Carriero. Coordination Languages and their Significance. *Communications of the ACM*, 35(2):96, 1992.
- [12] W. Hong, M. S. Nawaz, X. Zhang, Y. Li, and M. Sun. Using Coq for Formal Modeling and Verification of Timed Connectors. In *Proceedings of SEFM 2017*, volume 10729 of *LNCS*, pages 558–573. Springer, 2017.
- [13] S. T. Q. Jongmans and F. Arbab. Overview of Thirty Semantic Formalisms for Reo. *Scientific Annals of Computer Science*, 22(1):201–251, 2012.
- [14] R. Khosravi, M. Sirjani, N. Asoudeh, S. Sahebi, and H. Iravanchi. Modeling and Analysis of Reo Connectors using Alloy. In *Proceedings of COORDINATION 2008*, volume 5052 of *LNCS*, pages 169–183. Springer, 2008.
- [15] N. Kokash, C. Krause, and E. de Vink. Reo+mCRL2: A Framework for Model-checking Dataflow in Service Compositions. *Formal Aspects of Computing*, 24:187–216, 2012.
- [16] Y. Li and M. Sun. Modeling and Verification of Component Connectors in Coq. *Science of Computer Programming*, 113(3):285–301, 2015.
- [17] S. Owre, J. M. Rushby, and N. Shankar. PVS: A Prototype Verification System. In *Proceedings of CADE 1992*, pages 748–752. Springer, 1992.
- [18] S. Owre, N. Shankar, J. M. Rushby, and D. W. J. Stringer-Calvert. PVS System Guide, PVS Prover Guide, PVS Language Reference. Technical report, NASA, November 2001.
- [19] PVS dump files. Available at: <https://github.com/maqibdola/Reo-in-PVS>.
- [20] M. Sun. Connectors as Designs: The Time Dimension. In *Proceedings of TASE 2012*, pages 201–208. IEEE Computer Society, 2012.
- [21] M. Sun and F. Arbab. Web Services Choreography and Orchestration in Reo and Constraint Automata. In *Proceedings of SAC'07*, pages 346–353. ACM, 2007.
- [22] M. Sun and F. Arbab. A Model for Web Service Coordination in Long-Running Transactions. In *Proceedings of SOSE'10*, pages 121–128. IEEE Computer Society, 2010.
- [23] M. Sun, F. Arbab, B. K. Aichernig, L. Astefanoaei, F. S. de Boer, and J. Rutten. Connectors as Designs: Modeling, Refinement and Test Case Generation. *Science of Computer Programming*, 77(7-8):799–822, 2012.
- [24] X. Zhang, W. Hong, Y. Li, and M. Sun. Reasoning about Connectors in Coq. In *Proceedings of FACS 2016*, volume 10231 of *LNCS*, pages 172–190. Springer, 2016.
- [25] Z. Zlatev, N. Diakov, and S. Porkaev. Construction of Negotiation Protocols for E-Commerce Applications. *ACM SIGecom Exchanges*, 5(2):12–22, 2004.

# Modeling and Analyzing Hybrid Systems Using Hybrid Predicate Transition Nets

Dewan Mohammad Moksedul Alam

Xudong He

School of Computing and Information Sciences

Florida International University

Miami, Florida 33199, USA

dalam004@fiu.edu, hex@cis.fiu.edu

William (Cheng-Chung) Chu

Department of Computer Science

TungHai University, Taichung, Taiwan

cchu@thu.edu.tw

**Abstract**— Hybrid systems, especially in the form of cyber physical systems, have become ubiquitous and are playing critical roles in the functioning of society, however their design and implementation are extremely difficulty, especially regarding their dependability. In this paper, we propose a hybrid high level Petri net formalism, hybrid predicate transition nets (HPrTNs), for modeling and analyzing hybrid systems. We discuss some critical concepts and features of HPrTNs. We demonstrate the applicability of HPrTNs through several well-known benchmark hybrid systems and compare our results with other relevant methods. HPrTNs are fully supported in the tool environment PIPE+.

**Keywords**— formal methods; high-level Petri nets; hybrid Petri nets, differential Petri nets

## I. INTRODUCTION

Hybrid systems refer to the systems that arise out of the interaction of continuous dynamics and discrete dynamics. Many modern embedded systems, especially cyber physical systems are hybrid systems that contain physical devices having continuous dynamics and computational control processes with discrete behaviors. Hybrid systems have become ubiquitous and are playing critical roles in the functioning of society, however their design and implementation are extremely difficulty, especially regarding their dependability.

Hybrid systems have been the focus of intense research in the past few decades since they provide a convenient framework to accurately model a wide range of engineering systems and provide the flexibility to abstract complex physical behaviors away and to model dynamics having varying scales. Following the early works on the verification of digital circuits, many formalisms, methods and tools have been proposed to model and verify more complex embedded systems such as air traffic control systems, automotive control, bioengineering, process control, real-time communication protocols, manufacturing control, etc. One early prominent work is the hybrid automata [1] that provided a concrete mathematical framework for the analysis and verification of hybrid systems. Hybrid automata integrate diverse models such as differential equations and state machines in a single formalism with a uniform mathematical semantics and novel algorithms for multi-modal control synthesis and for safety and real-time performance analysis [2]. However, despite providing powerful methods to analyze hybrid

systems, the major inconvenience of hybrid automata is the dramatical increase of model dimensions for complex systems due to the intrinsic global state configurations and sequential behaviors of automata.

Petri nets, a concurrent and distributed formal models, provide a great flexibility to model complex systems. Petri nets have evolved in the past half century in many directions: including continuous Petri nets [3], fluid stochastic Petri nets [4]. Continuous Petri nets have further been extended to hybrid Petri nets [5] for modeling hybrid systems. Hybrid Petri nets inherit all the advantages of the Petri net model such as the ability to capture distributed behaviors, concurrency, synchronization and conflicts. Similar concepts have also been extended to high-level Petri nets [6, 7, 8] to model data dependent hybrid systems.

In this paper, we present our results in introducing continuous features into predicate transition nets (PrTNs) [9] for modeling hybrid systems. Specifically, we introduce two different kinds of places and transitions namely continuous places and continuous transitions with differential and difference equations. Our approach has a well defined priority rule to resolve the conflict of firing enabled discrete and continuous transitions. We have implemented the whole hybrid PrTN framework in our modeling tool PIPE+ [9]. We demonstrate how to model some classic examples of hybrid systems using PIPE+ and compare the modeling and simulation experience and performance with some existing and well known hybrid Petri net modeling tools.

## II. RELATED WORK

The concept of extending Petri net formalism to provide means to model continuous and hybrid systems was first presented in [10]. Based on this concept, several other extended Petri net formalisms were proposed. In the following subsections some of related formalisms and their applications and supporting tools are discussed.

### A. Hybrid Petri net Formalism

In [10], the authors combined a continuous Petri net representing continuous dynamics with a discrete Petri net capturing discrete behaviors. Subsequently, the authors extended their formalism to provide distinction between

autonomous and timed hybrid Petri nets and provided rules to resolve conflicts between continuous and discrete part [5, 11]. Hybrid Petri nets are based on low level Petri nets where tokens in continuous places are numerals and change rates associated with continuous transitions are simple difference equations. A slightly different approach was introduced in [12]. Here new kind of places and transitions were introduced, namely differential places and differential transitions. Differential places constitute the continuous state of the system being modeled. Differential transitions are always enabled and associated with a firing frequency, where first-order ordinary differential equations are used to represent the evolution rules. Another class of hybrid Petri nets is fluid stochastic Petri nets introduced in [4], which extended stochastic Petri nets to model hybrid stochastic systems. Apart from these, several other prominent works were published to extend other classes of Petri nets, batch Petri nets, hybrid flow nets, etc., to support modeling of hybrid systems.

Along with the research on the extension of the low-level Petri nets, several classes of high-level Petri nets have also been extended for modeling hybrid systems. One of such early approaches was proposed in [8], where a method was presented to extend timed hierarchical object-related nets (THORNs). In this extension, the author introduced real data type to THORNs to represent the continuously changing state variable and continuous transitions to capture the continuous evolution. In this approach, a continuous transition was enabled or disabled by inhibitor arcs and the evolution was specified using ordinary differential equations. However, this approach was not fully developed and supported by any tool. Among other classes of high-level Petri nets, Colored Petri nets were studied extensively and several approaches for extending them to model hybrid systems were proposed in [6, 7, 8, 13].

### B. Modeling and Analysis Tools

Although, both low-level and high-level Petri nets have been undergone rigorous studies and many extensions are proposed to model hybrid systems, not many efforts are made to provide proper tool support. Among low-level hybrid Petri net tools HYPENS [14], SimHPN [15] and HISim [16] are worth mentioning. Both HYPENS and SimHPN are not native Petri net tool, and are based on MATLAB and Simulink. They do not provide proper net editing capabilities. A user needs to use MATLAB/Simulink components to specify the semantics of the Petri net model of the system being modeled. HISim on the other hand integrates modeling and simulation in a unified tool but is functionally incomplete. In [7], the authors proposed a different approach to create a model using MATLAB components for simulation and provided a methodology to translate that into CPN for analysis. Among the tools in this context, Snoopy [13, 17] provides a unified experience of creating graphical model, simulation and analysis; but focuses on modeling biological systems. This tool supports several hybrid low-level and high-level Petri nets.

Our work provides a unified framework for system modeling and analysis using Hybrid high-level Petri nets leveraging our tool environment PIPE+.

## III. HYBRID PREDICATE TRANSITION NETS

In the following sections, we provide a formal definition of hybrid predicate transition nets (HPrTNs) by extending the definitions of PrTNs [18].

**Definition 1.** A HPrTN is a tuple  $N = (P, T, F, \Sigma, \alpha, \beta, \gamma, M_0)$ , where

- (1)  $P = P_d \cup P_c$  is a non-empty finite set of discrete places  $P_d$  and continuous places  $P_c$  (graphically represented by circles and double circles respectively);
- (2)  $T = T_d \cup T_c$  is a non-empty finite set of discrete transitions  $T_d$  and continuous transitions  $T_c$  (graphically represented by bars and boxed bars respectively), which disjoins  $P$ , i.e.  $P \cap T = \emptyset$ ;
- (3)  $F \subseteq P \times T \cup T \times P$  is a flow relation (the arcs of  $N$ ) such that  $\forall p \in P_c, t \in T_d. ((p, t) \in F \Leftrightarrow (t, p) \in F)$ ;
- (4)  $\Sigma = (St, Op, Eq)$  is the underlying algebraic specification with sorts  $St$ , operations  $Op$ , and equations  $Eq$ .  $\Sigma$  defines the set  $Token$  of tokens, the set  $Label$  of labels, and the set  $Constraint$  of constraints of  $N$ . In our tool environment, the  $\Sigma$ -algebra is instantiated with a subset of Java data types and their associated operations and laws;
- (5)  $\alpha: P \rightarrow \wp(St)$  associates each place  $p$  in  $P$  with a subset of sorts in  $St$  such that  $p \in P_d \Rightarrow real \notin \alpha(p)$  and  $p \in P_c \Rightarrow real \in \alpha(p)$ . The above constraints refer to projected component when  $\alpha(p)$  is a composite type;
- (6)  $\beta: T \rightarrow Constraint$  associates each transition  $t$  in  $T$  with a first order logic formula that defines the enabling condition (precondition) and the processing result (post-condition) of  $t$ ;
- (7)  $\gamma: F \rightarrow Label$  associates each flow relationship  $f$  in  $F$  with a label denoting the data flow of a relevant transition satisfying  $\forall p \in P_c, t \in T_d. (\gamma(p, t) = \gamma(t, p))$  and  $\forall p \in P_d, t \in T_c. (\gamma(p, t) = \gamma(t, p))$ , i.e. discrete transition can only read but not change continuous place, and continuous transition cannot change discrete place. Thus, this restriction corresponds to the concept of elementary hybrid Petri nets in [11], which does not allow the conversion between discrete and continuous markings;
- (8)  $M_0: P \rightarrow \wp(Token)$  is a sort-respecting initial marking such that  $\forall p \in P_c. (M_0(p) \neq \emptyset \wedge |M_0(p)| = 1)$ , i.e. each continuous place contains one and only one token.

The dynamic semantics of HPrTNs are defined on the concept of markings (states)  $M: P \rightarrow \wp(Token)$  that are mappings from places to tokens.

**Definition 2.** A transition  $t$  in  $T$  is *enabled* in a marking  $M$  if  $\forall p \in P. (\bar{\gamma}(p, t): \theta \subseteq M(p) \wedge \beta(t): \theta)$ , where  $\bar{\gamma}(p, t)$  is a generalization of  $\gamma$  such that  $(p, t) \notin F \Rightarrow \bar{\gamma}(p, t) = \emptyset$ .  $e: \theta$  is the result of instantiating all arc variables with tokens in  $p$  according to substitution  $\theta$ .

The enabling condition of a continuous transition here is similar to the strongly enabled concept in [11].

**Definition 3.** An enabled transition  $t$  in marking  $M$  with substitution  $\theta$  can *fire* and results in a new marking  $M'$  defined by:  $\forall p \in P. (M'(p) = M(p) \cup \bar{\gamma}(t, p) : \theta - \bar{\gamma}(p, t) : \theta)$ .

Two enabled transitions may fire at the same time as long as they are not in conflict, i.e. the firing of one them disables the other. Furthermore, a discrete transition has priority over a continuous transition if they are in conflict. All enabled continuous transitions are fired in a single round to reflect a snapshot of the time passage. The dynamic semantics (behavior) of a HPrTN is the set of all possible transition firing sequences starting from the initial marking. Various conflict situations involving continuous transitions are further discussed in the following sections.

#### IV. MODELING HYBRID SYSTEMS

In this section, we present new features implemented in PIPE+ to model hybrid systems. PIPE+ provides full support to model, simulate, and model check (using external dedicated model checkers) discrete event systems [9]. In the following sub-sections, we discuss only the features related to modeling continuous and hybrid systems.

##### A. Modeling Continuous Components

To provide support for modeling continuous components, two different elements, continuous places and continuous transitions, are introduced, which are significant different from their discrete counterparts in terms of both structure and dynamic semantics.

###### 1) Continuous Places

As in [5], continuous places represent the continuous part of the state space of the hybrid system being modeled. However, it is possible to design these places to represent more than one continuous attributes in HPrTNs. In other words, during modeling the continuous dynamics, the modeler can design these places to represent as many continuous dynamics as needed as long as they satisfy the constraints imposed on the dynamic semantics of the continuous components. This capability is quite useful to group together related dynamics instead of scattering those across multiple continuous places. We provide more insights on this in section V.

Continuous places share similar behavior as discrete places. The only differences between these are their data types. The data type of a continuous place (1) must be a singleton (or not a power set in our implementation), and (2) must have at least one number element. Thus, a continuous place can hold only one token and at least one element of this token is a numeral capturing the dynamic changing value of the system. A useful guideline is that a continuous place is modified by only one continuous transition. This helps to avoid (1) conflict between continuous transitions, and (2) inconsistent behavior. This restriction may be overcome by using more efficient scheduling mechanism of continuous transitions. Finally, continuous places are represented by double circle in PIPE+.

###### 2) Continuous Transition

Continuous transitions are used to model the continuously changing behavior of the hybrid system being modeled. Unlike other hybrid Petri nets, the behavior of continuous transitions in

HPrTNs is strategically different in many ways, including (1) continuous transitions are always enabled unless the discrete parts are a part of their preconditions, (2) the continuous transitions can modify discrete places, (3) the marking of the continuous places can control the speed of changes made by the continuous transitions. These strategies offer several benefits. The first strategy allows us to map the behavior of the continuous transitions analogous to the semantics of the hybrid systems, where the continuous part continuously changes the state of the system and the discrete part only controls the speed of the change. For example, when a car is stopped, the engine is running and consumes fuel and produces power; but its speed and acceleration remain zero since the transmission is detached. Thus, depending on the discrete control, some part produces positive/negative changes while other parts do not change. The second strategy provides modeling flexibility and the third allows us to model feedback mechanism. As an example, consider the movement of a pendulum. At every point, the dynamics (acceleration and speed) of the next depends on the dynamics of the current.

In HPrTNs, the constraints of continuous transitions are defined the same way as discrete transitions, and consist of preconditions and post-conditions specified in a first order logic formula. Generally, the preconditions of the continuous transitions consist of the data flow of the discrete input places attached to the transition in question. However, there is no restriction to use tokens from continuous places in the precondition, which is needed to model conditional branches (emulating if-else conditions) to compute new marking. This flexibility keeps the overall net size smaller.

The post-conditions of continuous transitions are similar to those of discrete transitions. However, some new concepts are introduced. First order ordinary differential equations can be used to compute continuous dynamics. These equations are used as part of integral operator. Listing A shows the format of ODE used with integral operation. One example of integral equation is shown there with changing variable  $q$ . The lower and upper limits of the equation is specified using the tokens of some place.

Table 1 – (A) Example of using differential equations

Format	$\int (ode, lower\_limit, upper\_limit) \partial change\_var$
Example	$\int (q/0.98, d1[1], d[1]) \partial q$

Post-conditions can be made dependent on time. An approximation of logical time is introduced for this purpose which is discussed in sub-section IV.C. A special operator,  $\tau$  (tau), is used to represent the current logical time. The following expression shows an example of using logical time.

$$y = x + \int (5, \tau - 1, \tau) \partial \tau$$

###### 3) Dynamic Semantics

Historically, PrTN is assumed to be autonomous without explicit timing information. HPrTNs follow the same concept. The evolution of its discrete components is the same as that of PrTNs. To compute the evolution of the continuous part, a slightly different approach is adopted. Since the evolution of the continuous part may depend on the time and/or other continually

changing components. Hence, some representation of time is needed. We discuss the modeling of time in HPrTNs in the next sub-section (IV.B).

All the continuous transitions are assumed to be always ready and to have equal firing speed. Thus, in each execution step, all the continuous transitions are evaluated to test whether they are enabled or not. All enabled continuous transitions are fired in every execution step. However, this strategy may result in conflict if two continuous transitions have the same input places but have different enabling conditions. The resolution of such conflict is discussed in sub-subsections (IV-C). The evaluation of the preconditions and post-conditions are the same as the discrete part as discussed in our earlier works [9].

Another interesting aspect of the dynamic behavior is the execution order of discrete and continuous parts. The discrete component takes the precedence. After all enabled discrete transitions fire once, the enabled continuous transitions start firing.

### B. Modeling Time

The continuous dynamics of hybrid systems is generally specified using differential or difference equations. To evaluate these equations, a reference clock is needed. Continuous dynamics is computed with respect to this reference clock. In PIPE+ there are several ways to specify this reference clock. This may be a random number, a function or a set of values specified in a place. This can be achieved by adding a pair of continuous place and a continuous transition. The place would hold the reference clock value and the transition would be responsible for the coherent evolution of the reference clock value. One can choose whatever one needs as the basis of change. Alternatively, a convenient and simple way is also provided for the basis of change, which approximates a logical clock. The clock simply counts the number of steps the execution performed so far. The step size should be assumed by the modeler, which can be an hour, one second, one milli-second or even one Nano-second. The current time provided by the clock can be accessed using the operator  $\tau$ .

### C. Conflict Resolution

In the context of Petri nets, a conflict arises when multiple transitions are enabled and firing one of them disables others. Conflicts can be categorized according to three common causes: (1) between two discrete transitions, (2) between one discrete and one continuous transitions and (3) between two continuous transitions. The conflict between discrete transitions is resolved using non-determinism. In this case, one of the enabled transitions is chosen non-deterministically to be fired and it is ensured that only one discrete transition is fired. This type of conflicts is already discussed in our prior work. The other two categories are interesting in the context of hybrid Petri net and are discussed below.

#### 1) Conflict Between Discrete and Continuous Transitions

This type of conflict arises when a discrete and a continuous transition are connected to the same input place, either discrete or continuous. In our approach, both discrete part and continuous part have their own separate methods of execution and we allow the execution of both these two parts in the same

step. We have a well-defined precedence between these two components that ensures that these two components do not modify the same place at the same time, which nicely resolves this type conflict.

#### 2) Between Continuous Transitions

There are three different ways that can lead to this type of conflict: (1) the transitions have the same input discrete place, (2) the transitions have the same input continuous place, and (3) they both have same output place. In the first case, there is conflict when any of the following conditions is true – (a) the satisfiability of the preconditions depends on the token from the common input place, and (b) the common place is a power set and the token satisfying the preconditions of the conflicting transitions is the same. To resolve this conflict, a token from a discrete place used in the evaluation of the precondition of a continuous transition must be returned to the place unaltered. This means that continuous transitions cannot modify such places. In the second case, since continuous places can hold only one token, the removal of that token in the process of execution of one of the conflicting transitions makes other transitions disabled. To resolve this, a different method of executing continuous transition is used. The output places of continuous transitions are updated once the preconditions are evaluated and post-conditions are computed for all the continuous transitions. The third case is an undesired situation, and should be avoided. It is the modeler's responsibility to ensure that no unexpected results can be produced. Although, the above is an undesired situation, no restriction is implemented for simplicity. Furthermore, it is very convenient when one continuous place is designed to store multiple continuous attributes with separate continuous transitions are used to access those attributes.

### D. Analysis

In PIPE+, only simulation and evolution graph are supported to analyze a hybrid system model. Model checking is not supported yet due to complexity of numerical functions used in modeling hybrid states. The simulator executes the net following the dynamic semantics of HPrTNs, and stores the state sequences of the system during the execution that can be used later to analyze the evolution. Simulation can be done using one step at a time for better understanding of the evolution or multiple steps in a single run to quickly have an overall picture of the system behavior. However, both these two-execution methods support (1) configuring the evolution graphs, and (2) exporting of the snapshots of the states for statistical analysis using other sophisticated tools.

#### 1) Evolution Graph

Evolution graphs show the evolution of the continuous attributes over time. Before starting simulation, a chart configuration UI is provided to select the attributes for evolution to be shown in charts. For each of these attributes a separate chart window is created. It is also possible to configure multiple attributes to be shown in the same chart window for better comparison. By default, the evolution of the attributes over time is shown in the charts. It is also possible to configure the charts to plot the evolution of one attribute against another.

#### 2) Export of Results

Evolution graph provides simple means to show how the value of some continuous attributes changes over time. It does not provide support to generate any other insight. To address this problem, simulation results can be exported for more sophisticated analysis. Generally, after each simulation step the state of the whole net along with the inputs and generated outputs are stored in a file. These can be exported to some external data analysis tools to better understand the behavior of the system.

## V. CASE STUDY

We have applied HPRTNs to model and analyze several well-known benchmark hybrid systems [19], including bouncing ball, thermostat, robotic motion controller, and obstacle avoidance. Due to space limit, we only show two systems here.

### A. Bouncing ball

In this model, the physics of a bouncing ball, i.e. its motion before, during and after the impact against another surface is modeled. In this model, the state of the ball is captured when it falls freely from a place 10 meters above the surface assuming 0.75 coefficient of restitution. Here, only the effect of gravity is considered. Figure 1 shows a pictorial diagram of the hybrid Petri net model. Table 2 lists the inscriptions of the net. Here, the continuous place *Dynamics* is used to store the velocity and height, real valued numbers as reflected in its datatype definition shown in Table 2. The continuous transition *Compute* computes these dynamics by following the basic laws of motion of freely falling objects as shown by  $\beta(\text{Compute})$  in Table 2. The discrete transition *Change* simply changes the direction of the motion by negating the velocity whenever the height falls below zero.

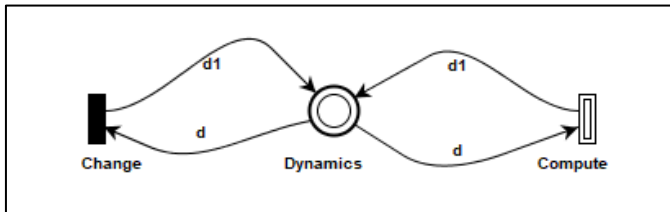


Figure 1 – Pictorial diagram of the bouncing ball model

Table 2 – Net inscription of the model in Figure 1

---

$\alpha(\text{Dynamics}) = \langle \text{number}, \text{number} \rangle$
$\beta(\text{Change}) = d[2] \leq 0 \wedge d[2] = 0 \wedge d[1] = -d[1] * 0.75$
$\beta(\text{Compute}) = d[1] = d[1] - 0.98 * \tau \wedge d[2] = d[2] * \tau - 0.49 * \tau * \tau + d[2]$

---

This model is simulated with various initial conditions, i.e. initial speed and height. Figure 2 shows the result of a simulation run when a ball is dropped from a height of 10m. Initial marking for this case is  $M_0(\text{Dynamics}) = \langle 0, 10 \rangle$ . The left chart in the figure shows the evolution of the velocity of the ball and the right chart shows the evolution of height with respect to time.

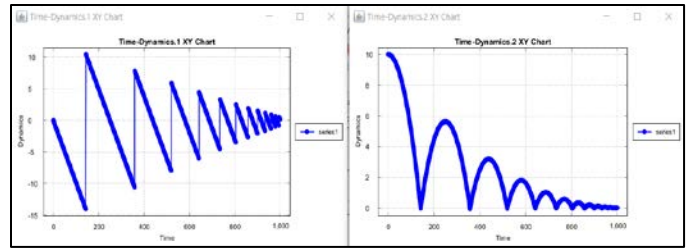


Figure 2 – Simulation result of the model in Figure 1

### B. Air Traffic Collision Avoidance

In air traffic control, collision avoidance maneuvers are used to resolve conflicting flight paths that arise during free flight [20]. These are very important and complex applications. A great number of different successful maneuvers are proposed and verified in the literature, many of them are also used in practice. As a case study, we model one of these maneuvers – straight line maneuver with instant turn. This maneuver involves a series of linear movement of the aircrafts. These movements can be controlled either from a central command center or from the approaching aircrafts' local control system. In our model, a central control system is used. Figure 3(a) shows the required movements of the aircrafts participating the straight-line collision avoidance maneuver, and Figure 3(b) shows the pictorial diagram of the hybrid PrTN model.

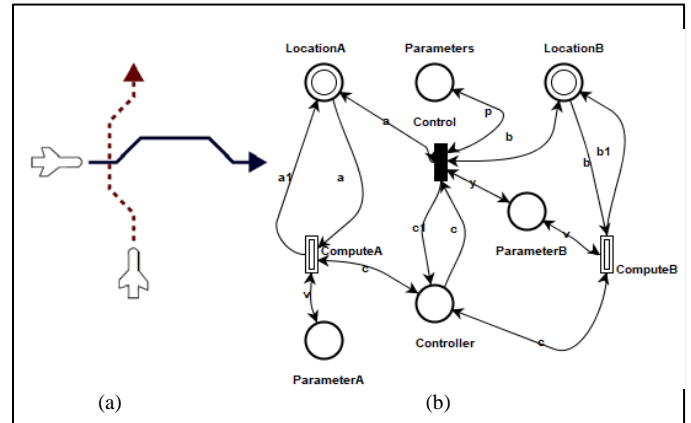


Figure 3 –(a) The movement of the aircrafts in straight line maneuver with instant turn. (b) A pictorial diagram of the hybrid PrTN model

In this model, two aircrafts *A* and *B* are participating the straight line with instant turn collision avoidance maneuver. Here, the place *Controller* stores the parameters to control the directions of the aircrafts participating the maneuver. The transition *Control* generates these control parameters depending on the state. The control parameter here is basically an angle that dictates the direction of the aircrafts. The place *ParameterA* stores the velocity and the angle of direction of the aircraft *A*. *LocationA* stores the location of *A*. The transition *ComputeA* computes the location of *A* using its parameters and the control parameter. *ParameterB*, *LocationB* and *ComputeB* do the same for aircraft *B*. The place *Parameter* defines the safe horizontal and vertical distances. Due to space limit, the detailed net inscription is omitted here. We have simulated this model with different sets of initial conditions, i.e. initial locations, velocity and directions of the aircrafts, different safe distances. Figure 4



shows the result of a simulation run where the aircraft A starts from the location (0,0) along X-axis and aircraft B starts from (18, 0) towards the opposite direction of A. Both have equal ground speed of 200 m/s. The safe horizontal and vertical distances are 12 and 2 kms respectively. In Figure 4, the charts (a) and (b) show the entrance and exit of the collision avoidance maneuver of the aircrafts A and B respectively. Initially both A and B move towards each other, when A reaches just after 3 and B reaches 15, the vertical distance falls below the safe distance. Both planes turn left and follow that direction until they reach the safe vertical distance. When the safe distance is reached, they turn right and follow their own course. An error of 200m from the original course is allowed as shown according to the constraint of transition *Controller*.

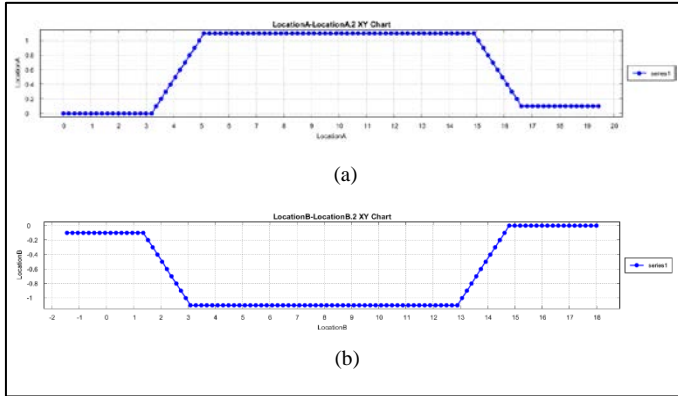


Figure 4 – Simulation results of the model in Figure 3(b).

## VI. CONCLUSIONS AND FUTURE DIRECTIONS

In this paper, we presented hybrid predicate transition nets (HPrTNs) to model and analyze hybrid systems. The whole framework is implemented in our tool environment PIPE+. We have shown how the new features of PIPE+ can be utilized to model and analyze hybrid systems through several well-known hybrid systems. We have studied several other tools providing modeling and analyzing capability of hybrid Petri nets. Most of these tools are tailored to specific applications such as bio-medicine. Some of these tools use other modeling languages, like MATLAB, to generate Petri net models. PIPE+ provides a unified environment for modeling and analyzing high-level Petri nets including HPrTNs.

This work is an initial attempt to extend PrTN towards hybrid system modeling and analysis. We will study the explicit time representation as in timed Petri nets. We will investigate new and improved scheduling algorithms of discrete and continuous transitions to cover more realistic and sophisticated applications.

## ACKNOWLEDGEMENT

Alam and He were partially supported by AFRL under FA8750-15-2-0106. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

## REFERENCES

- [1] Henzinger T.A., The Theory of Hybrid Automata. In: Inan M.K., Kurshan R.P. (eds) Verification of Digital and Hybrid Systems. NATO ASI Series (Series F: Computer and Systems Sciences), vol 170. Springer, Berlin, Heidelberg, 2000.
- [2] J. Lygeros, C. Tomlin, and S. Sastry, "Hybrid Systems: Modeling, Analysis and Control", December 2008, retrieved from <http://www-inst.cs.berkeley.edu/~ee291e/sp09/handouts/book.pdf>
- [3] H. Alla and R. David, "A modeling and analysis tool for discrete events systems: continuous Petri net", Performance Evaluation, 33(3), pp. 175-199, August 1998.
- [4] Trivedi, K. S. and Kulkarni, V. G. 1993. FSPNs: Fluid stochastic Petri nets. *14th International Conference on Application and Theory of Petri Nets*, Chicago.
- [5] R. David and H. Alla, "Discrete, Continuous, and Hybrid Petri Nets", Springer Berlin Heidelberg, Springer, 2010.
- [6] M. Herajy, F. Liu and C. Rohr, "Coloured hybrid Petri nets for systems biology", *Biological Process and Petri Nets*, 1159, pp. 60-76, June 2014.
- [7] D. Bera, K. van Hee and H. Nijmeijer (2015), "Modeling Hybrid Systems with Petri Nets", In: *Obaidat M., Ören T., Kacprzyk J., Filipe J. (eds) Simulation and Modeling Methodologies, Technologies and Applications*. Advances in Intelligent Systems and Computing, vol 402. Springer, Cham
- [8] R. Wieting, "Modeling and Simulation of Hybrid Systems Using Hybrid High-level Nets", In: *Proceedings of the 8th European Simulation Symposium (ESS'96)*, Vol. II, pages 158-162, October 1996, Genoa, Italy.
- [9] D. Alam and X. He, "A method to analyze high level Petri nets using SPIN model checker", in *Proceedings of the 29th International Conference on Software Engineering & Knowledge Engineering*, pp. 161-166, July 2017.
- [10] R. David, H. Alla, Continuous Petri nets, in: *Proc. 8th European Workshop on Application and Theory of Petri Nets*, Zaragoza, Spain, 1987.
- [11] R. David and H. Alla, "On Hybrid Petri Nets", *Discrete Event Dynamic Systems*, 11, pp. 9-40. January 2001, doi: 10.1023/A:1008330914786
- [12] I. Demongodin, N.T. Koussoulas, Differential Petri nets: Representing continuous systems in a discrete-event world, *IEEE Trans. Automat. Control* (1998).
- [13] M. Heiner, M. Herajy, F. Liu, C. Rohr, M. Schwarick, "Snoopy – A Unifying Petri Net Tool", *Application and Theory of Petri Nets. PETRI NETS 2012. Lecture Notes in Computer Science*, vol 7347. Springer, Berlin, Heidelberg, 2012.
- [14] F. Sessego, A. Giua, C. Seatzu, "Simulation and Analysis of Hybrid Petri Nets using the Matlab Tool HYPENS", *SMC08: 2008 IEEE Int. Conf. on Systems, Man, and Cybernetics (Singapore)*, October 2008.
- [15] J. Júlvez, C. Mahulea, and C. R. Vázquez, "SimHPN: A MATLAB toolbox for simulation, analysis, and design with hybrid Petri nets", *Nonlinear Analysis: Hybrid Systems*, 6(2), pp. 806-817, March 2012.
- [16] A. Amengual, "A Specification of a Hybrid Petri Net Semantics for the HISim Simulator", accessed from <http://www.icsi.berkeley.edu/pubs/techreports/TR-09-003.pdf>, 2009.
- [17] M. Herajy, F. Liu, C. Rohr and M. Heiner. "Snoopy's hybrid simulator: a tool to construct and simulate hybrid biological models", *BMC Systems Biology*, 11:71, July 2017.
- [18] X. He: "A Comprehensive Survey of Petri Net Modeling in Software Engineering", *International Journal of Software Engineering and Knowledge Engineering - IJSEKE*, vol. 23, no. 5, 2013, 589-626
- [19] R. Alur: "Principles of Cyber-Physical Systems", The MIT Press, 2015.
- [20] A. Platzer, "Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics", Springer-Verlag Berlin Heidelberg 2010

# Visualizing Interactions in AngularJS-based Single Page Web Applications\*

Gefei Zhang<sup>†</sup>  
Hochschule für Technik und Wirtschaft Berlin  
gefei.zhang@htw-berlin.de

Jianjun Zhao  
Kyushu University  
zhao@ait.kyushu-u.ac.jp

## Abstract

*AngularJS is a popular framework for single page web applications. In AngularJS applications, the programming logic is implemented in Javascript, while the layout is defined separately in HTML files. Due to this separation, data and control flow is usually hard to track. We propose a method to visualize the data and control flow in AngularJS-based single page web applications and separate interactions from each other. Our method helps to get a better understanding of the application's work flow, to realize the boundaries of the interactions, and to know what is updated in an interaction and what is not.*

## 1 Introduction

AngularJS [2] is one of the modern frontend-frameworks which support the Model-View-View-Model architecture (MVVM) [8]: the models provide data to the application, the views define the graphical presentation of the data, and the view-models (also called controllers) define the business logic (data and control flows) of the application. Usually, views are defined in HTML, models and controllers in Javascript.

AngularJS is widely used in single page web applications (SPA). In an SPA, the application has only one HTML page, containing an array of widgets. When the user gives some input in one widget, the application reacts and updates some other widgets. Between the widgets of the page there may or may not exist data and control flow, and a widget may or may not be influenced by another one. Since data and control flow is defined in the controller, separately from the widgets, potential interactions may be obscure; understanding of the program may be hard.

We present a method to visualize data and control flow of AngularJS-based SPA. We create an *Interaction Diagram* by translating HTML widgets, as well as functions and variables in the controller to nodes,

and the invocation, reading and writing relationships between them as edges. The interaction diagram not only visualizes possible workflows of the application, but is also a starting point for more static analysis. In this paper, we show how to calculate “slices” of interactions, that is, to isolate the widgets involved in an interaction from those that are not.

The rest of this paper is organized as follows: In the following Sect. 2, we give a brief introduction to AngularJS, and also present our running example. Section 3 introduces interaction diagrams. In Sect. 4 we show how to analyse workflows of the application and define test cases using the interaction diagram. Related work is discussed in Sect. 5. Finally, in Sect. 6, we conclude and outline some future work.

## 2 AngularJS

We first give a brief introduction to AngularJS by means of a simple example, and then define an abstract syntax for AngularJS applications. Due to space limitation, we focus on a small subset of AngularJS; it is relatively straight-forward to extend our approach to cover other features of the framework.

### 2.1 Running Example

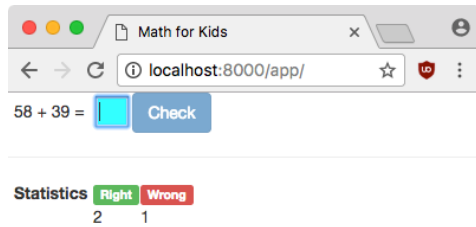
Figure 1 shows an AngularJS application to teach kids addition. In the upper part an addition problem is presented, the lower part shows statistics of how many right and wrong answers the user has given. When a new problem is shown, the user can enter her answer in an input field (Fig. 1(a)), the system then shows if the answer is right or wrong. Meanwhile, a new button appears. When the user clicks on it, a new problem is generated (Fig. 1(b)). In the lower part of the browser, a statistics is shown of how many right and wrong answers the user has given. In Fig. 1(b), the user has answered four questions in total, where three of the answers were correct, and one was wrong.

### 2.2 Data binding

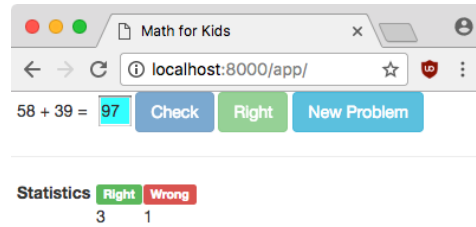
In AngularJS, the program logic is implemented in a so-called *controller* in Javascript, and the graphical

\*DOI reference number 10.18293/SEKE2018-066

<sup>†</sup>Partially supported by the EU project cAPITs and the German BMBF project deep.TEACHING (01IS17056).



(a) The system waiting for the user to enter an answer



(b) The system gives a verdict, updates statistics, and shows a button to generate a new problem

**Figure 1. Example: Math exercises with statistics**

layout of the application is defined in a *template* in the HTML syntax, with some special attributes of HTML tags, which are defined by AngularJS and called *directives*. The template of the running example is shown in Fig. 2,<sup>1</sup> where we removed styles of the HTML elements from the listing for simplicity. The controller is written in Javascript, and provides data and event handlers for the application. The controller of the running example is listed in Fig. 3.

The communication between template and controller takes place in the controller’s variable `$scope`. Just like every other object in Javascript (cf. [5]), `$scope` is also a collection of key-value pairs. In Javascript, the keys are called *properties*. The value of a key can be any object, and in particular, any function. For example, in lines 4 and 5, Fig. 3, `$scope` is extended by two properties `count_right` and `count_wrong` with initial value 0. In lines 17, the property `may_check` is assigned a function. In Javascript, function properties are usually not changed after initial assignment, while other properties are often overwritten and used as variables. We therefore call the latter *variable properties*.

The Javascript object `$scope` is the controller’s interface to the interface: its properties (e.g., `$scope.count_right` in line 4 and `$scope.check_answer` defined in lines 21 to 25) are visible to the template, and may be bound to HTML elements to provide data or event handlers. Other top-level functions and variables of the controller (e.g., `c` in line 7 and `add_problem` defined in lines 8 to 15), which are not properties of `$scope`, are only for “private” use in the controller and not visible to the template.

Data flow between template and controller is defined by *data binding*: an HTML element in the template may be *bound* to a property of the object `$scope` of the controller, and gets updated automatically when the value of the property changes. Data binding is defined in so-called directives; directives are included in the template as attributes of HTML elements.

<sup>1</sup>The whole project is available under <https://bitbucket.org/gefei/angularjs-example>

AngularJS supports both *one-way* and *two-way* data binding. The directive `ng-bind` or double braces `{{}}` define one-way data binding, that is, the HTML element automatically presents the up-to-date value of the bound property of `$scope`, whilst changes of the value presented by the HTML element, if any, would not be propagated from the GUI back to the controller (the `$scope`).<sup>2</sup> For instance, in Fig. 2, line 5, `{{a}}` and `{{b}}` will be replaced by the values of the variables `$scope.a` and `$scope.b` at runtime, respectively. The values of the two variables are assigned in function `add_problem` of the controller (Fig. 3, lines 11 and 12). A little more examination of the controller reveals that this function is called to generate a new addition problem, and `$scope.a` and `$scope.b` hold the values of the two summands.

Other directives defining one-way data binding include `ng-if` and `ng-disabled`. In `ng-if`, the value of the bound property is not shown; instead it regulates the visibility of the HTML element: if and only if the Javascript expression bound to `ng-if` is valued to `true` is the HTML element visible in the GUI. For example, both of the two buttons defined in lines 8 and 9 of Fig. 2 are guarded with `ng-if`. The conditions for the buttons being visible are the variable `$scope.right` being `true` and `false`, respectively. Figure 3 shows that the variable is set in line 22, in function `$scope.check_answer`, and is `true` iff the value of variable `c` equals to the value of `$scope.answer` parsed as an integer.

Directive `ng-disabled` is used to set HTML widgets disabled (i.e., the user cannot enter her input). In line 6 of Fig. 2, the button is disabled when the negation of the result of function `$scope.may_check()`, which is defined in the lines 17–19 of Fig. 3. The function returns `true` iff the input field is not empty (`!!$scope.answer`) and `$scope.right` is still undefined (i.e., the button `Check` has not been clicked yet, see Sect. 2.3). When this is the

<sup>2</sup>There is some subtle difference between the semantics of `ng-bind` and double braces, see <https://stackoverflow.com/a/16126174>. However, the difference is not relevant for our discussion. In this paper, we consider `ng-bind` and double braces as equivalent.

```

1 <html>
2 <body ng-app="app">
3 <div ng-controller="controller">
4   <form>
5     {{a}} + {{b}} = <input ng-model="answer">
6     <button ng-disabled=!may_check()
7       ng-click="check_answer()">Check</button>
8     <button ng-if="right===true">Right</button>
9     <button ng-if="right===false">Wrong</button>
10    <button ng-click="new_problem()">New Problem</button>
11  </form>
12  <hr>
13
14  <table>
15    <tr>
16      <th>Statistics</th>
17      <th><span>Right</span></th>
18      <th><span>Wrong</span></th>
19    </tr>
20    <tr>
21      <td></td>
22      <td><span ng-bind="count_right"></span></td>
23      <td><span ng-bind="count_wrong"></span></td>
24    </tr>
25  </table>
26 </div>
27 </body>
28 </html>

```

**Figure 2. HTML template**

case, the function returns `true`, `ng-disabled` receives the value `false`, then the button is disabled, and vice versa.

We call the property of `$scope` that an HTML element is bound to the *target* of the data binding, and the HTML element the *source*. While `ng-bind` and `{{}}` show the value of the target in the GUI, `ng-if` and `ng-disabled` do not output the value in the textual form. It rather *influences* the appearance of the source by setting its visibility or enabled/disabled status. In both cases, we say the source of a data binding *presents* the value of the target.

The directive `ng-model` defines *two-way* data binding, that is, the widget automatically updates when the value of its bound property changes, and any change of the widget's value will be propagated to the bound variable. Therefore, a bi-directional data flow is defined. For instance, in line 5 of Fig. 2, the `input` element has an attribute `ng-model=answer`. The input field is therefore bound to the variable `$scope.answer`: the value of the input field is hold in `$scope.answer`, changes are propagated automatically in both directions. In Fig. 3, the correct answer of the current problem is stored in variable `c` (line 10), and `$scope.right` is calculated by a comparison with `$scope.answer` in line 22.

The target of a one-way data binding may be a variable property or a function property of `$scope`. If it is a variable property, the source presents the target's value. If it is a function property, the widget presents the return value of the function. On the other hand, the target of a two-way data binding must be a variable property, since the target must store the value of the user input, and only a variable, as opposed to a

```

1 var app = angular.module('app', []);
2
3 app.controller('controller', function($scope){
4   $scope.count_right = 0;
5   $scope.count_wrong = 0;
6
7   var c;
8   function add_problem() {
9     var max = 100;
10    c = Math.floor(Math.random() * (max - 1)) + 1;
11    $scope.a = Math.floor(Math.random() * (c - 2)) + 1;
12    $scope.b = c - $scope.a;
13    $scope.answer = undefined;
14    $scope.right = undefined;
15  }
16
17  $scope.may_check = function() {
18    return !!$scope.answer && $scope.right === undefined;
19  }
20
21  $scope.check_answer = function() {
22    $scope.right = c === parseInt($scope.answer);
23    $scope.count_right += ($scope.right) ? 1 : 0;
24    $scope.count_wrong += ($scope.right) ? 0 : 1;
25  };
26
27  $scope.new_problem = function() {
28    add_problem();
29  }
30
31  add_problem();
32 })

```

**Figure 3. Controller**

function, can be assigned a value.

The template of the statistics is defined in an HTML table (lines 14 to 25 in Fig. 2), where two `td` cells (lines 22 and 23) present the values of `$scope.count_right` and `$scope.count_wrong` by one-way data binding.

## 2.3 Event Handling

HTML elements may also be provided with event handlers. Upon the given event, the specified function is executed. For example, in line 7 of Fig. 2, the directive `ng-click` binds the function `$scope.check_answer` to the button. Therefore, the function is invoked when the user clicks the button. The function is defined in Fig. 3, lines 21 to 25. When invoked, it first (line 22) checks if `c` has the same value as `$scope.answer` (which we know is the current value of the user input in the text field, see Sect. 2.2), parsed as an integer, and assigns the result to `$scope.right`. Then, depending on if `$scope.right` is true or not, the value `$scope.count_right` or `$scope.count_wrong` is incremented by one. Therefore, the statistics gets updated when the user clicks the Check button (Fig. 2, lines 22 and 23).

The button New Problem also has an event handler: in Fig. 2, line 10, the directive `ng-click="new_problem()"` binds the function `$scope.new_problem` to handle the event of the button being clicked. The function calls another function `add_problem` to generate a new problem by updating the values of `$scope.a`, `$scope.b`, setting `c`, to be undefined to clear the field for the user to in-

put her answer (recall: this field has a two-way data binding, as defined in line 5 of Fig. 2), setting both `scope.right` and `scope.answer` to be undefined. Therefore, when `New Problem` is clicked, `scope.may_check` will return true, and the button `Check` will get enabled (Fig. 2, line 6).

Variable `c` and function `new_problem` are not defined as `scope`'s properties. Therefore, they are local to the controller, and not exposed to the template.

## 2.4 Abstract Syntax

An AngularJS-based SPA is a tuple  $(T, C, D, E)$ .  $T$  is a template, written in HTML and consisting of a set of HTML tags  $(T = \{h\})$  which define HTML widgets,<sup>3</sup>  $C$  is the definition of a controller, written in Javascript,  $D$  is a set of data bindings, and  $E$  is a set of event handler bindings.

The controller definition  $C$  is modeled as a tuple  $(V, F, scope)$ , where  $V$  is a set of top-level variables,  $F$  is a set of top-level functions, and  $scope \in V$  is a distinguished element of  $V$ . We write  $V(scope)$  for the set of `scope`'s variable properties, and  $F(scope)$  for the set of `scope`'s function properties. We also define  $W = V \setminus \{scope\}$  to be the set of top-level variables defined in the controller other than `scope`.

$D$  is the set of data binding relations between HTML tags and variable properties of `scope`:  $D \subseteq \{(h, V(scope) \cup F(scope))\}$ . Given  $d = (n, o) \in D$ , we define  $source(d) = n$  and  $target(d) = o$ . Two-way data bindings build a subset  $D' \subseteq D$ , and  $\forall d \in D'$ , it holds that  $target(d) \in V(scope)$ .

$E$  is the set of event handler bindings between HTML tags and function properties of `scope`:  $E \subseteq \{(h, F(scope))\}$ .

Additionally, for each  $f \in F \cup F(scope)$ , we define  $R(f) \subseteq V \cup V(scope)$  and  $W(f) \subseteq V \cup V(scope)$  to be the set of the variables  $f$  reads from and writes to, respectively. We also define  $Inv(f) \subseteq F$  to be the functions invoked by  $f$ .<sup>4</sup> In this paper, we take these sets for granted. Since Javascript is an "extremely dynamic" [7] language, this is in general not always the case. However, in modern software development, *readability first* is considered best practice, and it is reasonable to assume that at least reading, writing, and invocation relationships can be obtained by simple analysis.

## 3 Interaction Diagram

In order to understand the workflow of the application, it is necessary to study both its HTML and its

<sup>3</sup>Precisely, HTML tags may be nested, thus the HTML template is a tree. In this paper, we do not consider nesting tags, and just view the template as a set.

<sup>4</sup>Although it is also possible for a `scope` function to call another `scope` function, it is usually not necessary and not a good idea.

Javascript code, as well as their interactions. We now present Interaction Diagrams to visualize the overall behavior, combining the logic defined in HTML and Javascript. Figure 4 shows the interaction diagram for our running example.

An Interaction Diagram ID is a directed graph  $(N, E)$ . The set of nodes is defined as the union of three sets:  $N = N_H \cup N_{scope} \cup N_{js}$ , where, using the notations introduced in Sect.2.4,

- for each  $(h, v) \in D$ , we generate a node  $n_h$ , and  $N_H = \{n_h \mid (h, v) \in D\}$ . Graphically, we label  $n_h$  with the label of  $h$ , or, if  $h$  does not have a label, the name of the target of the data binding (without `scope`), extended by the position of the definition of  $h$  in the template;
- for each  $(h, v) \in D$ , we create a node  $n_v$ , and for each  $(h, e) \in E$ , we create a node  $n_e$ .  $N_{scope}$  is then defined as  $N_{scope} = \{n_v \mid (h, v) \in D\} \cup \{n_e \mid (h, e) \in E\}$ . Graphically, we label each  $n_f \in N_{scope}$  with the name of the  $f$  (with `scope`), but do not include the position of  $f$ 's appearances in the controller.
- for each  $v \in W$ , we create a node  $n_v$ ; for each  $f \in F$ , we create a node  $n_f$ , and  $N_{js} = \{n_v \mid v \in W\} \cup \{n_f \mid f \in F\}$ . Graphically, we label these nodes with the name of the variable or function.

For example, Fig. 4 shows the interaction diagram for our running example. The upper compartment shows  $N_H$ , which contains a node for each of the text fields `a`, `b` (which show the two summands to the user), `scope.answer` (where the user may enter her answer), the buttons `Check` (which the user may click to check if the answer is correct), the labels `Right` and `Wrong` (which show the result of the check to the user), the labels `count.right` and `count.wrong` (which show the current success statistics of the user), and the button `New Problem` (which the user may click to generate a new problem).

The lower compartment models components of the controller. It contains the elements of  $N_{scope}$  and  $N_{js}$ . That is, it contains a node for each of the variables `scope.a`, `scope.b` (which hold the values of the summands), `scope.answer` (which holds the answer inputted by the user), `scope.right` (which holds the value `scope.check_answer` returns, see below), `scope.count.right` and `scope.count.wrong` (which hold the current number of right and wrong answers the user has given), and the functions `scope.check_answer` (which checks if the answer inputted by the user is correct), and `scope.new_problem` (which generates the summands and key of a new problem). The lower compartment also contains a node for `c`, which holds the key of the new problem.

The edges  $E$  of the interaction diagram are used to model data and control flow. We define  $E$  as the union of six subsets:  $E = E_{data} \cup E'_{data} \cup E_{event} \cup E_W \cup E_R \cup E_{Inv}$ , where

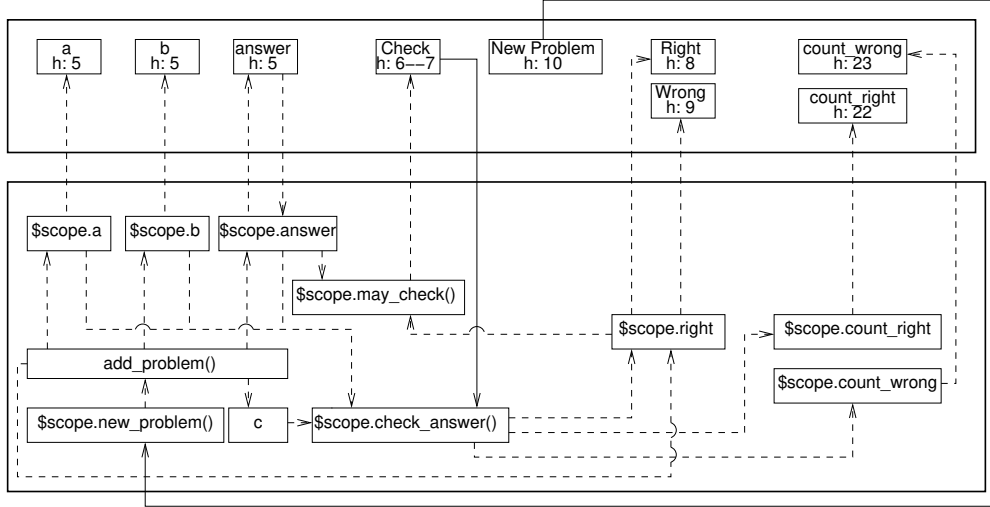


Figure 4. Interaction diagram

- for each  $d \in D$ , we create an edge  $e_d = (\text{target}(d), \text{source}(d))$ . The set of all data-flow edges from the controller to HTML widgets is then modeled as  $E_{\text{data}} = \{e_d \mid d \in D\}$ . If  $d \in D'$ , we additionally create an edge  $e'_d = (\text{source}(d), \text{target}(d))$  to model the HTML widget reading value from its bound variable. The set of all data-flow edges from HTML widgets to the controller is then modeled as  $E'_{\text{data}} = \{e'_d \mid d \in D'\}$
- for each  $(h, f) \in E$ , we define an event-handling edge  $e_h = (n_h, n_f)$ . The set of all event-handling control flow is then modeled as  $E_{\text{event}} = \{e_h \mid (h, f) \in E\}$
- for each pair  $(f, v)$ ,  $f \in F \cup F(\text{\$scope})$ ,  $v \in W(f)$ , we create an edge  $e_{f,v}$ . The set of writing relations is then modeled as  $E_W = \bigcup_{f \in F \cup F(\text{\$scope})} \{e_{f,v} \mid v \in W(f)\}$ . For each pair  $(v, f)$ ,  $f \in F \cup F(\text{\$scope})$ ,  $v \in R(f)$ , we create an edge  $e_{v,f}$ . The set of reading relations is then modeled as  $E_R = \bigcup_{f \in F \cup F(\text{\$scope})} \{e_{v,f} \mid v \in R(f)\}$
- for each  $f \in F \cup F(\text{\$scope})$  and each  $v \in \text{Inv}(f)$ , we create an edge  $e_{f,v}$ . The relations of a function writing a variable is then modeled by  $E_{\text{Inv}} = \bigcup_{f \in F \cup F(\text{\$scope})} \{e_{v,f} \mid v \in \text{Inv}(f)\}$ .

Graphically, we use a dashed-line arrow to represent each  $e \in E_{\text{data}} \cup E'_{\text{data}} \cup E_W \cup E_R \cup E_{\text{Inv}}$ , and a solid-line arrow to represent each  $e \in E_{\text{event}}$ . Note for event handling, the exact event is not modeled, and it does not need to be modeled in an Interaction Diagram.

In our example (see Fig. 4), the elements of  $E_{\text{data}}$  model one-way data binding:  $(\text{\$scope.a}, a)$ ,  $(\text{\$scope.b}, b)$ ,  $(\text{\$scope.count\_right}, \text{count\_right})$ ,  $(\text{\$scope.count\_wrong}, \text{count\_wrong})$ ,  $(\text{\$scope.right}, \text{Right})$ ,  $(\text{\$scope.wrong}, \text{Wrong})$ , and  $(\text{\$scope.may\_check}(), \text{Check})$ .  $E'_{\text{data}}$  contains the two

edges between  $\text{\$scope.answer}$  and  $\text{answer}$ , modeling the only two-way data binding in the example.  $E_{\text{event}}$  contains the two edges  $(\text{Check}, \text{\$scope.check\_answer}())$  and  $(\text{New Problem}, \text{\$scope.new\_problem}())$ .

Furthermore,  $E_W$  contains the four edges leaving  $\text{add\_problem}()$ , and the edges leaving  $\text{\$scope.check\_answer}()$ .  $E_R$  contains the edges from  $c$  to  $\text{\$scope.check\_answer}()$ , and from  $\text{\$scope.answer}$  to  $\text{\$scope.may\_check}()$ .  $E_{\text{Inv}}$  contains the edge from  $\text{\$scope.new\_problem}()$  to  $\text{add\_problem}()$ .

#### 4 Interactions between Widgets

SPAs are interactive: the user makes some input, the system reacts and makes updates to some widgets, then the user makes another input, and so on. We define an interaction to be a round of user giving input, and the system updating widgets accordingly. An interaction can be triggered explicitly by the user invoking an event handler, or implicitly while the user is updating data, which is bound by `ng-model`. Starting from interaction diagrams, it is easy to “slice” interactions, i.e., to find out the widgets that get updated upon a certain piece of user input.

A widget  $t$  reacts to another widget  $s$  iff in the interaction diagram  $t$ 's representation  $n_t$  is reachable from  $s$ 's presentation  $n_s$ , and the only event-handling edge, if any, on the path from  $n_s$  to  $n_t$  event-handling edge is the very first edge (leaving  $n_s$ ) on the path. This edge models the interaction being explicitly triggered.

Formally, given a node  $n \in \mathbb{N}_H$ , we say a node  $m \in \mathbb{N}_H$  reacts to  $n$  iff

1.  $\exists n_0, n_1, n_2, \dots, n_k \in \mathbb{N}, n_0 = n, n_k = m$  such that for each  $0 \leq i < k$ ,  $(n_i, n_{i+1}) \in E$ , and
2.  $\forall n_p, 1 < p \leq k$  and  $\forall e \in E, \text{target}(e) = n_p$  it holds that  $e \notin E_{\text{event}}$ .

We write  $I(n)$  for the set of all nodes representing the widgets that react to  $n$ . This set contains the widgets that are automatically updated upon user input, and thus constitute an interaction. To analyze interactions in the SPA, we calculate for each *input widget*, i.e., widget with an edge leaving it in the interaction diagram. Based on Fig. 4, we can calculate the following three interactions for our running example:

- $I(\text{answer}) = \{\text{Check}, \text{answer}\}$ , which means when the user is entering her answer, `answer` (which is trivial) and `Check` (enabled) are updated. Note that updating the answer does not trigger `$scope.check.answer()`, since this function needs explicit triggering via `Check`,
- $I(\text{Check}) = \{\text{Check}, \text{Right}, \text{Wrong}, \text{count\_right}, \text{count\_wrong}\}$ , which means when the user clicks on `Check`, these widgets get an update: the button itself (disabled), one of `Right` and `Wrong` is displayed, indicating whether the user-inputted answer is correct, and one of the counts also gets updated.
- $I(\text{New Problem}) = \{a, b, \text{answer}, \text{Check}, \text{Right}, \text{Wrong}\}$ , which means when the user clicks `New Problem`, the widgets `a`, `b` (showing the summands of the new problem), `answer` (emptied), `Check` (enabled), as well as `Right` and `Wrong` (both made invisible), are updated.

This analysis shows us the boundary of the interactions. For instance, according to the analysis, when the user is updating an answer, the label `Right` or `Wrong` is not shown, nor does the statistics get updated. Instead, these widgets are only updated when the user clicks `Check`.

## 5 Related Work

Analysis of Javascript programs is a very dynamic research field. Due to the very dynamic nature of the language, its analysis is not an easy task, see [7] for an overview of recent publications. One of the challenges is the interactions between browser, DOM and Javascript. In [6, 3], a unified API is given to formalise the browser behavior. However, large-scale libraries are still difficult to analyze, and therefore their functionalities are often modeled manually. Our work is also along this line in that we also take the semantics of AngularJS for granted.

For the analysis of such frameworks, it is essential to understand their interactions [7]. This is exactly where our work is positioned. Other interesting publications in this area include [4], which provides a method of checking name and type consistency between template and controller, and [1], where the authors present a hybrid method for change impact analysis, and its focused on plain Javascript, without frameworks. Compared with these approaches, our focus is on the visu-

alization and analysis of boundaries of interactions in applications based on a complex framework.

## 6 Conclusions and Future Work

We presented a method for visualization and analyzing AngularJS-based single page web applications. Based on the interaction diagrams, it is easy to calculate the interactions, and to understand which widgets react to certain user input and which do not. Our approach is helpful for understanding AngularJS programs, and thus for more powerful analysis.

In the future, we plan to automate this approach, to extend the analysis by more AngularJS directives, and based on this work, to investigate techniques for automatic test generation for AngularJS programs.

## References

- [1] Saba Alimadadi, Ali Mesbah, and Karthik Pattabiraman. Hybrid DOM-Sensitive Change Impact Analysis for JavaScript. In *Proc. 29<sup>th</sup> Eur. Conf. Object-Oriented Programming (ECOOP 2015)*, pages 321–345, 2015.
- [2] Google. AngularJS. <https://angularjs.org/>.
- [3] Simon Holm Jensen, Magnus Madsen, and Anders Møller. Modeling the HTML DOM and Browser API in Static Analysis of JavaScript Web Applications. In *Proc. 19<sup>th</sup> ACM SIGSOFT Symp. Foundations of Software Engineering and 13<sup>th</sup> Eur. Software Engineering Conf. (FSE/ESEC 2011)*, pages 59–69, 2011.
- [4] Frolin S. Ocariza Jr., Karthik Pattabiraman, and Ali Mesbah. Detecting Inconsistencies in JavaScript MVC Applications. In *Proc. 37<sup>th</sup> Int. Conf. Software Engineering (ICSE 2015), Volume 1*, pages 325–335, 2015.
- [5] Mozilla. JavaScript object basics. <https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/Basics>, 2018.
- [6] Changhee Park, Sooncheol Won, Joonho Jin, and Sukeyoung Ryu. Static Analysis of JavaScript Web Applications in the Wild via Practical DOM Modeling. In *Proc. 30<sup>th</sup> Int. Conf. Automated Software Engineering (ASE 2015)*, pages 552–562, 2015.
- [7] Kwangwon Sun and Sukeyoung Ryu. Analysis of JavaScript Programs: Challenges and Research Trends. *ACM Comput. Surv.*, 50(4):59:1–59:34, 2017.
- [8] Wikipedia. Model View ViewModel. [https://en.wikipedia.org/w/index.php?title=Model\\_View\\_ViewModel&oldid=675433955](https://en.wikipedia.org/w/index.php?title=Model_View_ViewModel&oldid=675433955), 2015.

# Software Visualization Using Topic Models

Sandeep Reddivari  
School of Computing  
University of North Florida  
Jacksonville, FL, USA 32224

William Hackney  
School of Computing  
University of North Florida  
Jacksonville, FL, USA 32224

## ABSTRACT

Latent Dirichlet Allocation (LDA) is a statistical topic modeling approach that has been used to support several software engineering activities. The main assumption is that LDA offers a unique insight into the semantic content of software systems, thus revealing otherwise unseen relations between software artifacts. However, a main problem when dealing with LDA is the complexity of its output. In particular, the numerical probabilistic distributions produced by LDA to represent topics and documents are not intuitive to understand and rationalize. To address this problem, in this paper we present a topic modeling based approach to visualize software systems based on LDA. We also present several visualizations to represent the basic elements of LDA including words, topics, and documents. These different basic views are combined through a set of integration links to enable users to effectively explore software systems by supporting knowledge discovery at different levels of abstraction. We also demonstrate how the topic modeling based visualization approach can provide support to several software engineering activities such as program comprehension, software clustering, and code evolution analysis.

## Keywords

Software visualization, program comprehension, topic modeling

## 1. INTRODUCTION

Software visualization can be defined as the mapping from software artifacts to graphical representations [16]. The main assumption is that using graphical representations of code structures reduces the cognitive effort required for understanding and navigating large and complex software systems [22]. However, extracting and visualizing meaningful information from source code is a non-trivial task [14, 18]. This can be explained based on the fact that the lexicons and syntax of programming languages is inherently more constrained than natural language [21]. Therefore, to be effectively visualized, the complex textual content of software systems has to be first reduced down to lower dimensional representations, while retaining as much of the original meaning of the text as possible [25]. These reduced representations can then be mapped into basic graphical objects to produce views of the system at higher levels of abstraction. A reductive transformation that has gained a considerable attention in Natural Language Processing (NLP)

DOI reference number: 10.18293/SEKE2018-194

related tasks is Latent Dirichlet Allocation (LDA) [5]. Using LDA, the dimensionality of a large text corpus can be reduced down into a set of meaningful latent topics, where each topic consists of a group of words collectively representing a cohesive domain concept. In particular, LDA is an unsupervised probabilistic approach for estimating a topic distribution over a text corpus. The main assumption is that documents in the text collection are generated using a certain statistical generative model as random mixtures over latent topics. LDA has been successfully applied to several software engineering activities. However, a main problem when dealing with LDA is the complexity of its output. In particular, the numerical probabilistic distributions produced by LDA are not intuitive to understand and rationalize. This has motivated researchers to start looking for alternative ways to represent LDA's output [6, 13, 24].

Motivated by these observations, in this paper we propose a collection of visualization techniques, which are often used to represent topic models in NLP, to visualize software systems. These visualizations include combinations of basic and integrated views that facilitate effective navigation and comprehension of software systems. The rest of the paper is organized as follows. Section 2 briefly introduces LDA and our preprocessing analysis. Section 3 the various suggested visualization techniques of LDA. Section 4 presents visualization supports for clustering evaluation and code evolution analysis. Finally, Section 5 presents the conclusion and future work.

## 2. LATENT DIRICHLET ALLOCATION

LDA takes the documents collection  $D$ , the number of topics  $K$ , and  $\alpha$  and  $\beta$  as inputs. Each document in the corpus is represented as a bag of words  $d = \langle w_1, w_2, \dots, w_n \rangle$ . Since these words are observed data, Bayesian probability can be used to invert the generative model and automatically learn  $\phi$  values for each topic  $t_i$ , and  $\theta$  values for each document  $d_i$ . In particular, using algorithms such as Gibbs sampling [23], an LDA model can be extracted. This model contains for each  $t$  the matrix  $\phi = \{\phi_1, \phi_2, \dots, \phi_n\}$ , representing the distribution of  $t$  over the set of words  $\langle w_1, w_2, \dots, w_n \rangle$ , and for each document  $d$  the matrix  $\theta = \{\theta_1, \theta_2, \dots, \theta_n\}$ , representing the distribution of  $d$  over the set of topics  $\langle t_1, t_2, \dots, t_n \rangle$ .

## 3. CODE VISUALIZATION USING LDA

This section describes the various visualization techniques used to represent the different aspects of the latent topic structure of software systems produced by LDA. We start



**Table 1: Experimental Datasets**

Dataset	VER.	NO. CLASS	LANG.	LOC	COMMENTS
<i>iTrust</i>	15.0	299	Java	20.7K	9.6K
<i>Apache Ivy</i>	2.3.0	451	Java	49.9K	16.7K
<i>WDS</i>	3.5.1	521	Java	44.6K	10.7K



**Figure 1: Tag cloud for a sample topic from the *iTrust* system**

by describing basic views used for representing the basic units of LDA including words, topics, and documents. We then describe more sophisticated views where multiple basic views are integrated to represent the whole system at higher levels of abstraction.

### 3.1 Datasets

To start our analysis, we used three software systems from different application domains. Table 1 describes the characteristics of these systems including: the size of the system in terms of lines of source code (SLOC), lines of comments (CLOC), implementation language (LANG.) version (VER.) and number of classes (CLS).

### 3.2 The Topic View

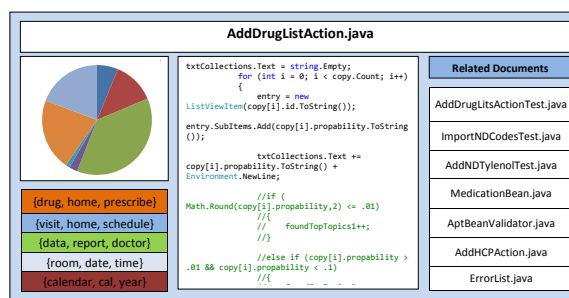
This view emphasizes topics as the main unit of visualization. The topic view allows users to visually explore a topic as a collection of weighted words organized in a *tag cloud*. Tag clouds are visually-weighted renditions of collections of words (tags), extracted from a certain corpus, where more important words are depicted in a larger font size than less important words [17]. Importance can be quantified based on different schemes such as words counts, or the TFIDF weights of words. Tag clouds are widely used as an effective method to quickly find relevant information on the Web [3].

In our analysis, each topic is represented as a separate tag cloud. Each tag in the cloud represents a word from the topic-word distribution matrix. In particular, the size of the word in the cloud is proportional to its probability in the topic word matrix  $P(w_i, t)$  i.e., words with higher probability are shown in larger font. Fig. 1 shows a tag cloud generated for a topic from the *iTrust* dataset. The topic is represented by 20 terms. The size of the different words  $\langle$ appointment, schedule, duration, ..., etc. $\rangle$  in the tag cloud in Fig. 1 shows that this particular topic describes the domain concept of scheduling a patient appointment.

### 3.3 The Document View

This view emphasizes artifacts as the main unit of visualization. In particular, each artifact ( $d$ ) in the system is represented using a combination of graphical and textual components including:

- Topics chart: A standard pie chart which shows the topic distribution of each document. In particular,



**Figure 2: Document view of AddDrugListAction.java**

each sector of the pie represents a topic in the document-topic matrix. The size of the sector is proportional to the  $P(t_i, d)$  of that topic. A unique color is used to represent each topic [6].

- Topics list: A list of color-coded topics arranged in a descending order based on their  $P(t_i, d)$  value. The top three words of each topic are used as representatives of the topic; 3-5 terms were found to decently convey the main theme of the topic [7].
- Document text: A main window that shows the actual artifact (i.e., the actual source code of the class).
- Related artifacts: A list of other artifacts in the system ranked in a descending order based on their topical similarity to the main document. In particular, The similarity between documents  $d_1$  and  $d_2$  can be measured by the similarity between their corresponding topic distributions, using techniques such as the cosine similarity, or approaches such as the Kullback-Leibler (KL) divergence measure [26].

Fig. 2 shows a document view window for the class `AddDrugListAction.java` from the *iTrust* dataset. The view shows that the topic  $\langle$ drug, home, prescribe $\rangle$  is the dominant topic in this class. It also shows that the class `AddDrugListActionTest.java` is the most topically similar class to `AddDrugListAction.java`.

### 3.4 The Document-Topic View

This visualization provides an in-depth look into the topic distribution of each artifact in the system. In particular, stacked charts are used to represent the document-topic matrix of each document. Using stacked charts, the contribution of several data items into a total are represented as bars stacked one on top of, or next to, each other. In LDA, the width/height and color of each bar represents the  $P(t_i, d)$  of each topic representing the document  $d$ . Stacked charts are known to be effective and intuitive for comparing data distributions [11]. In our analysis, the main objective of this particular view is to enable the comparison of the documents at topic level. Comparing the topic distribution of documents is essential in several software engineering activities such as code evolution analysis [2] and traceability recovery [27].

### 3.5 The System-Topic View

This view is used to represent the whole system in a single view. We adopt distribution maps as the main visualization

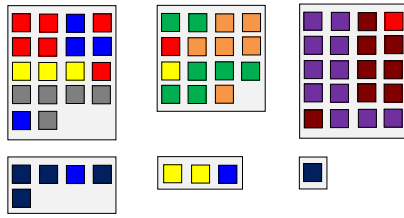


Figure 3: A topic distribution map of *Apache Ivy*

for this view [10]. This visualization is composed of large rectangles containing small squares. Each rectangle represents a system folder or a package, and each square represents an artifact within that folder. The color of the square represents the dominant topic of the artifact. Distribution maps are suitable for showing the conceptual content of a corpus. In particular, they are good for visually representing metrics as such *focus*, or how well-encapsulated or cross-cutting a topic is, and *spread*, or the parts of the system in which this topic is present as a dominant topic [10].

Fig. 3 shows a partial distribution map for the *Apache Ivy* dataset. In particular, the map shows how some topics are spread over multiple folders, dominating multiple artifacts in each folder. It also shows how some other topics are focused (well-encapsulated) within one folder only.

### 3.6 The Integrated View

The integrated view combines several basic views to produce a full picture of the system, or one integrated visualization experience that allows users to navigate through the system’s multiple views at different levels of abstraction. This view is enabled through a number of integration links that connect the different basic views of the system. Technically, such links could refer to a mouse click; in other implementations such as Web platforms, hyperlinks could be adopted. To integrate our views, several links have been added to the system. These links are shown in Fig 4. Such links include:

- In the system-topic views (i.e, distribution map), there is a link between each artifact’s graphical object (square) and the document view of that artifact. An additional secondary link (right-click) is also available to show the tag cloud of the artifact’s dominant topic.
- In the document view, a click on a topic in the list of document’s topics lunches the tag cloud of the topic. Also, a click on any of the artifacts in the list of topically similar documents will lunch that artifact’s document view.
- Links have also been added to the stacked charts views, where there is a link between each bar in the chart and the tag cloud of the topic represented by that bar.

The main objective of the integrated view is to facilitate effective program comprehension by enabling different comprehension strategies including top-down and bottom-up comprehension [19]. In particular, under the top-down strategy, software developers utilize their knowledge about the domain to build a set of expectations that are mapped onto the source code [19]. The top-down comprehension process starts from the distribution map view of the system, where developers who are usually familiar with the domain,

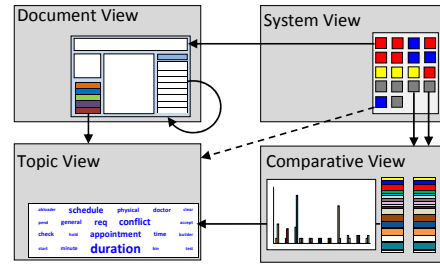


Figure 4: Integrated View

can explore the system as one unit. Such knowledge can then be propagated through the integration links down to the basic views of individual artifacts and topics, where users can investigate the source code of individual classes in the document view and the word distribution of each topic through the topic’s tag cloud.

On the other hand, a bottom-up comprehension approach adopts a divide and conquer strategy. In particular, the comprehension process starts from the source code, where developers who are often unfamiliar with the domain can then gradually build a full understanding of the system. This process is also enabled by moving to higher abstraction levels through the integration links, starting from basic document’s and topic’s views, up to the system’s distribution map to produce a full understanding of the system from the bottom-up.

## 4. VISUALIZATION SUPPORT

### 4.1 Clustering Evaluation

LDA has been intensively used as a mechanism for clustering software systems. In particular, the topic distributions of artifacts are used as main features to group topically related artifacts into smaller, more conceptually cohesive, and thus, easier to understand subsystems [12, 21]. However, finding best LDA clustering settings (e.g., number of topics,  $\alpha$ ,  $\phi$ , clustering algorithm to use, number of clusters, and the distance function) that best fit a certain task is often described as an NP-complete problem [28].

Several internal and external methods have been proposed in the literature to estimate optimal clustering parameters. Internal methods use a fitness function that captures the twin objectives of high cohesion and low coupling to help determine the boundaries between clusters [8]. On the other hand, external measures (e.g., MoJo [29]) use an authoritative decomposition as a reference point to assess the quality of generated clusters. However, such methods reduce the overall evaluation into a single number, hiding valuable information about the nature of the problem. To that end, visualizing clustering results can help to assimilate such information, providing insights into the operation of the different clustering algorithms, and their sensitivity to different clustering parameters, such as number of clusters, membership, and boundaries. [9]

In the following example, we demonstrate how topic modeling based visualizations can be used to visually compare clustering algorithms. In particular, we experiment with Complete Linkage (CL) and Single Linkage (SL), two Hierarchical Agglomerative Clustering (HAC) algorithms that have been showing consistent performance in several soft-

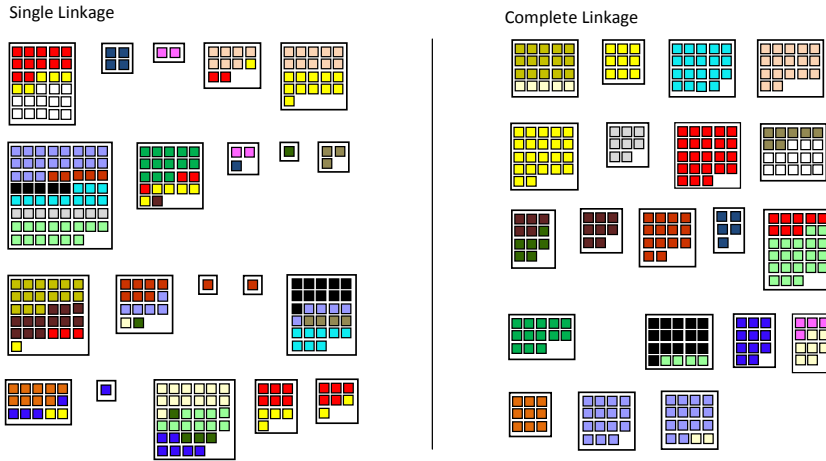


Figure 5: Visually comparing the clustering results of SL and CL algorithms on *iTrust*

ware engineering activities [1]. Technically, both SL and CL start from individual items in the cluster space, where each item is treated as a separate cluster, iteratively merging items based on a certain update rule, until the whole system is grouped into a single cluster. This process produces a hierarchy of clusters, or a dendrograph, of the whole process. Formally, SL and CL can be defined as follows:

$$SL = \min\{d(a, b) : a \in A, b \in B\}. \quad (1)$$

$$CL = \max\{d(a, b) : a \in A, b \in B\}. \quad (2)$$

In these equations,  $d(a, b)$  is the distance between data objects  $a$  and  $b$ , defining the linkage (merging) criteria for clusters  $A$  and  $B$ . For instance, SL merges the two clusters with the smallest minimum pairwise distance, and CL merges the two clusters with the smallest maximum pairwise distance. Cosine similarity is used to measure the topical distance between the different artifacts.

We apply LDA using  $K = 20$  to our *iTrust* dataset. We then cluster the system’s artifacts using both CL and SL algorithms. We set both algorithms to produce 20 clusters. We then use MoJoFM to assess the performance of both algorithms by comparing their output to the optimal decomposition [29]. An optimal decomposition is the one that groups documents that match in their dominant topics into separate clusters. Since we produced 20 topics for the system, the optimal decomposition consists of 20 clusters.

MoJo measures the distance between two decompositions of a software system by computing the number of Move and Join operations to transform one to the other. MoJoFM is basically a normalized MoJo that produces a number in the interval  $[0, 1]$ . MoJoFM value of 1 means that the algorithm is optimal. Applying MoJoFM on our clustering output returns values of  $< 87.71\%, 43.1\% >$  for CL and SL respectively, giving a clear indication of the superiority of CL over SL; however, no other information is provided. To reveal such information, we refer to our distribution maps to visually compare the output of the two algorithms. Results are shown in Fig. 5. Each map shows how each clustering algorithm divided the 299 artifacts of the *iTrust* system among the 20 clusters.

Fig. 5 shows that SL produced unbalanced clusters, scattering some of the topically related artifacts all over the clustering space. This behavior of SL can be explained based on its update rule, SL uses the smallest minimum pairwise

Algorithm	MoJo	Avg. Spread	Avg. focus
<i>Complete Linkage</i>	87.71%	1.35	82%
<i>Single Linkage</i>	43.1%	2.35	48%

distance between cluster items as the new distance in the newly formed clusters. Therefore, when SL is used, bigger clusters tend to be grouped together rather than incorporating singletons (clusters with only one element). As a result, SL tends to create a small number of large, isolated clusters, in addition to a number of singletons. In constant, CL uses the smallest pairwise maximum distance to merge clusters, thus pushes clusters apart, creating smaller, more balanced, and highly cohesive clusters.

To confirm our visual assessment, we refer to the *focus* and *spread* metrics associated with distribution maps. Such values, averaged over all the topics in *iTrust*, are shown in Table 2. The *spread* values show that using SL, artifacts sharing same dominant topic are spread over an average of 2.35 clusters, while in CL, the average spread is 1.35, giving an indication of the high encapsulation of topically related artifacts in CL clusters. Similarly, we calculate the focus values of both algorithms using the following formula:

$$focus(t, P) = \sum touch(t, p_i) * touch(p_i, t) \quad (3)$$

where  $touch(t, p_i)$  is the ratio of the number of artifacts in  $p_i$  with dominant topic  $t$  to the total number of artifacts in  $p_i$ , and  $touch(p_i, t)$  is the ratio of the number of artifacts in  $p_i$  with dominant topic  $t$  to the total number of artifacts in the system with a dominant topic  $t$  [10]. Focus values show that SL was less successful in producing cohesive clusters ( $focus = 48\%$ ), producing larger clusters that contain multiple sets of topically related artifacts. In contrast, CL produced more cross-cutting clusters ( $focus = 82\%$ ).

In general, using our distribution map view, it can be visually concluded that CL was more effective than SL in dividing the system’s artifacts into more topically cohesive and more balanced clusters. In particular, visualization helps users to visualize the different metrics and the meaning of these numbers, providing an effective alternative for illustrating the impact of the various clustering parameters, and facilitating a real-time comparison of different clustering re-

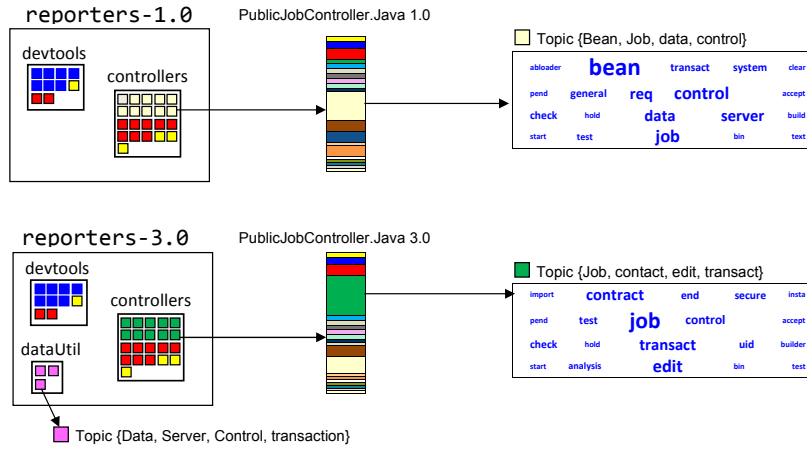


Figure 6: Visually analyzing the change in one of the *WDS* system’s folders

sults. For instance, the concentration of colors inside boxes gives an indication of high encapsulation and clear cross-cutting. In addition using the integrating links, the user can get a better understanding of the results, for example why certain artifacts with different dominant topics tend to be grouped together.

## 4.2 Code Evolution Analysis

Several studies have reported that topic models can be effectively used for the purposes of describing software evolution [20]. The main assumption is that the changes in the latent topic structure of a software system are reflective of actual changes made by developers, for example feature integration or refactoring milestones [27]. In particular, such events can be detected by applying LDA to different releases of the system and monitoring the changes in a system’s topic structure over time.

For instance, the sensitivity of LDA to software evolution can be quantified through matching individual artifacts based on their document-topic distribution, or comparing how the topic-word matrices of the different topics in the system have changed over time using well-defined evolution metrics [27]. However, similar to software clustering, such analysis can reveal only a little about the nature of the change [15]. For instance, other evolution-related information such as what specific types of change took place, and which artifacts have been affected, is often invisible to such metrics. To that end, visualization can help developers and software engineers to uncover such information by providing a more in-depth look into the different releases of the system overtime. In particular, to visualize a software change, a time dimension is integrated into the different views of the software system [15]. For example, to visualize system artifacts’ evolution, we add a time dimension to our document-topic view. Similarly, a time dimension is integrated into the system-topic view of the system to visualize changes in the system as a whole.

To demonstrate this process we run LDA over two releases of the *WDS* dataset. In particular, we work with releases 1.0 and 3.0 of the system as milestone changes have been reported between these two particular releases. In order to keep a consistent color assignment, if two topics generated for different releases of the system share the top five dominant words, then the same color is used. Using our zooming feature, we produce distribution maps of one the system’s sub-folders (*reporting*) in both releases of the system. In

release 1.0, this particular folder contains two sub folders including *controllers* and *devtools*. In release 3.0 the same folder now includes the additional folder *dataUtilities*.

Distribution maps, shown in Fig. 6, show that artifacts in folder *devtools* remain unchanged between releases. However, a drastic change in a portion of the artifacts in folder *controllers* has happened. To understand this change we take a look at the stacked chart of the system class *PublicJobController.Java* in both releases. The charts show that in release 1.0 the topic  $\langle \text{bean}, \text{job}, \text{data}, \text{control} \rangle$  was dominating this class. This topic describes functionalities related to job requirements and database connections. However, in release 3.0 this class is now dominated by the topic  $\langle \text{job}, \text{contact}, \text{edit}, \text{transact} \rangle$ , and the data related terms are no longer present. In an attempt to understand this change we inspect the new folder *dataUtilities* that has emerged in the folder *reporting* in release 3.0. All artifacts in this folder are dominated by the same topic. It can be inferred from the word distribution of this topic  $\langle \text{data}, \text{server}, \text{control}, \text{transaction} \rangle$  that this folder basically contains database related functionalities. This suggests that probably EXTRACT CLASS [4] refactoring has taken place somewhere between releases 1.0 and 3.0, where most of the database related features have been moved and encapsulated into the new folder *dataUtilities*, leaving artifacts in folder *controllers* with only job-controlling related functions.

This example demonstrates how a change in the system was made immediately obvious by our views, allowing users to not only identify the change, but also provide insight into the nature of the change.

## 5. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a set of visualization techniques to represent source code using topic models. The main assumption is that using graphical representations to represent complex source code structures helps to reduce the cognitive load when comprehending a software system. In particular, we used LDA as an effective dimensionality reduction technique to reduce the inherently complex textual content of source code into a set of semantically cohesive topics that can be effectively visualized. Such topics are represented through several views that have been used in a wide range of NLP applications. These views provide graphical representations for the different numerical distributions produced by LDA including words, topics, and documents. These dif-

ferent basic views are integrated through a set of links to enable users to quickly browse through the system modules, exploring relationships between artifacts that might otherwise go unnoticed. In addition, we presented how the topic modeling based visualization can provide support to software engineering scenarios such as program comprehension, code clustering, and evolution analysis. In the future we plan to implement our proposed visualizations through a working prototype which provides several options to adjust the visualization settings and to navigate through the different views of the system.

## 6. REFERENCES

- [1] N. Anquetil, C. Fourrier, and T. Lethbridge. Experiments with clustering as a software modularization method. In *Working Conference on Reverse Engineering*, pages 235–255, 1999.
- [2] H. Asuncion, A. Asuncion, and R. Taylor. Software traceability with topic modeling. In *International Conference on Software Engineering*, pages 95–104, 2010.
- [3] S. Bateman, C. Gutwin, and M. Nacenta. Seeing things in the clouds: The effect of visual features on tag cloud selections. In *ACM Conference on Hypertext and Hypermedia*, pages 193–202, 2008.
- [4] G. Bavota, A. D. Lucia, and R. Oliveto. Identifying extract class refactoring opportunities using structural and semantic cohesion measures. *Journal of Systems and Software*, 84(3):397–414, 2011.
- [5] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.
- [6] A. Chaney and D. Blei. Visualizing topic models. In *AAAI Conference on Social Media and Weblogs*, 2012.
- [7] J. Chang, J. Boyd-Graber, S. Gerrish, C. Wang, and D. Blei. *Reading tea leaves: How humans interpret topic models*, pages 288–296. Curran Associates, 2009.
- [8] J. Davey and E. Burd. Evaluating the suitability of data clustering for software modularization. In *Working Conference on Reverse Engineering*, pages 268–277, 2000.
- [9] I. Davidson. Visualizing clustering results. In *SIAM International Conference on Data Mining*, 2002.
- [10] S. Ducasse, T. Girba, and A. Kuhn. Distribution map. In *International Conference on Software Maintenance*, pages 203–212, 2006.
- [11] S. Eick. Visualizing software changes. *IEEE Transactions on Software Engineering*, 28(4):396–412, 2002.
- [12] S. Grant and J. Cordy. Estimating the optimal number of latent concepts in source code analysis. In *International Working Conference on Source Code Analysis and Manipulation*, pages 65–74, 2010.
- [13] B. Gretarsson, J. O’Donovan, S. Bostandjiev, T. Höllerer, A. Asuncion, D. Newman, and P. Smyth. Topicnets: Visual analysis of large text corpora with topic modeling. *Journal of Visual Languages and Computing*, 3(2):2157–6904, 2012.
- [14] M. Hearst. Tilebars: Visualization of term distribution information in full text information access. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 59–66, 1995.
- [15] R. Holt and J. Pak. GASE: Visualizing software evolution-in-the-large. In *Working Conference on Reverse Engineering*, pages 163–167, 1996.
- [16] R. Koschke. Software visualization in software maintenance, reverse engineering, and re-engineering: A research survey. *Journal of Software Maintenance*, 15(2):87–109, 2003.
- [17] B. Kuo, T. Hentrich, B. Good, and M. Wilkinson. Tag clouds for summarizing web search results. In *International Conference on World Wide Web*, pages 1203–1204, 2007.
- [18] R. Laramee. Using visualization to debug visualization software. *IEEE Computer Graphics and Applications*, 30(6):67–73, 2003.
- [19] S. Letovsky. Cognitive processes in program comprehension. In *workshop on empirical studies of programmers on Empirical studies of programmers*, pages 58–79, 2011.
- [20] E. Linstead, C. Lopes, and P. Baldi. An application of Latent Dirichlet Allocation to analyzing software evolution. In *International Conference on Machine Learning and Applications*, pages 813–818, 2008.
- [21] E. Linstead, P. Rigor, S. Bajracharya, C. Lopes, and P. Baldi. Mining concepts from code with probabilistic topic models. In *International Conference on Automated Software Engineering*, pages 461–464, 2007.
- [22] J. Maletic, A. Marcus, and M. Collard. A task oriented view of software visualization. In *International Workshop on Visualizing Software for Understanding and Analysis*, pages 32–40, 2002.
- [23] I. Porteous, D. Newman, A. Ihler, A. Asuncion, P. Smyth, and M. Welling. Fast collapsed gibbs sampling for Latent Dirichlet Allocation. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 569–577, 2008.
- [24] T. Savage, B. Dit, M. Gethers, and D. Poshyvanyk. TopicXP: Exploring topics in source code using latent dirichlet allocation. In *IEEE International Conference on Software Maintenance*, pages 1–6, 2010.
- [25] K. Sparck-Jones. Automatic summarising: Factors and directions. In *Advances in Automatic Text Summarization*, pages 1–12, 1998.
- [26] M. Steyvers and T. Griffiths. *Probabilistic topic models*, pages 427–448. Psychology Press, 2007.
- [27] S. Thomas, B. Adams, A. Hassan, and D. Blostein. Validating the use of topic models for software evolution. In *IEEE Working Conference on Source Code Analysis and Manipulation*, pages 55–64, 2010.
- [28] K. Tian, M. Revelle, and D. Poshyvanyk. Using Latent Dirichlet Allocation for automatic categorization of software. In *International Working Conference on Mining Software Repositories*, pages 163–166, 2009.
- [29] Z. Wen and V. Tzerpos. An effectiveness measure for software clustering algorithms. In *International Workshop on Program Comprehension*, pages 194–203, 2004.

# How Many Versions Does A Bug Live in? An Empirical Study on Text Features for Bug Lifecycle Prediction

Chuanqi Wang<sup>1,2</sup>, Yanhui Li<sup>1,2,\*</sup>, Baowen Xu<sup>1,2,\*</sup>

1. State Key Laboratory for Novel Software Technology, Nanjing University, China

2. Department of Computer Science and Technology, Nanjing University, China

\* Corresponding authors: {yanhuili, bwxu}@nju.edu.cn

**Abstract**—During the software system’s maintenance and evolution, finding and removing software bugs is a very important part that consumes a large amount of money and effort. To analyze different bugs’ character, it is very essential to know how long or which period of versions does the bug live in.

In this study, we define version-based bug lifecycle and propose a text features based classification model to predict the version-length of bug lifecycle. We collect 57000+ bugs from 10 well-know Apache Software Foundation projects to construct our dataset, and use the tf-idf method to collect our text features from bug report’s summary and description.

Our experimental results show that the text feature based method performs better than other baseline methods on 10 projects. The text feature based Naive Bayes classifiers outperforms all other methods with different features and classifiers.

## I. INTRODUCTION

Finding and removing Software bugs is a very important part of software evolution and maintenance that consumes a large amount of money and effort [1]. An extensive body of bug-related studies [2]–[6] have been proposed to help programmers to predict, detect and fix bugs. In all bug-related research areas, the **bug lifecycle** (the time difference between bug introduced time and bug fixed time) is an important time indicator, which is useful for many applications, such as predicting fault-proneness of code region [7] or identifying the origins of bugs [8]. The recent empirical study shows that some long-lifecycle bugs may remain alive for a very long time and in multiple versions [9], [10].

Some researchers have paid attention to bug lifecycle and their studies could be divided into two parts: **Fixing Period**, (the righthand side of bug lifecycle, from reported time to fixed time) and **Dormant Period** (the lefthand side, from introduced time to reported time). In Fixing Period part, the lengths of Fixing Period are usually considered as the bug fixing effort (BFE). To investigate which factor impacts the BFE, the correlation analysis was conducted [11]. Zhang et al. focused on the correlation of different factors and BFE by logistic regression models [12]. Furthermore, based on different datasets and classifiers, researchers have proposed prediction models to predict the BFE when the bug was reported. Song et al. proposed the association mining rules to

build BFE prediction model on NASA’s data sets [13]. Zhang et al. proposed k-Nearest Neighbors based method to construct the BFE prediction model on commercial projects [14]. In Dormant Period part, Chen et al. introduced affected version as an indicator for bug introduced time, then calculated the dormant period from introduced time to reported time [9].

Our work differs from existing studies in three important ways. **Firstly**, we consider the bug lifecycle as a whole in our study. Previous studies usually focus on the righthand or lefthand side of lifecycle. **Secondly**, we define version-based bug lifecycle and predict the version-length of bug lifecycle. Previous studies are mainly based on real time interval. We observe that in most projects, interval time between versions are different, so we believe that version could be considered an effect time units of measurement. **Thirdly**, we focus on **text** features from the summary and description in bug reports and construct the bug lifecycle prediction model. Our model aims to help the project manages and bug fixers trace the bug fixing back to the bug introducing commit.

Our main contribution consists of the following steps:

- We collect the datasets with 57000+ bugs on 10 Apache Software Foundation Projects from Jira and Github. Each projects have more than 2000 bugs.
- We use tf-idf [15], a statistic method to calculate a single word’s importance, to construct our text features.
- Our evaluation results show that our text features significantly improves the performance of the version-based bug lifecycle prediction model.

The rest of this paper is organized as follows. Section II defines bug lifecycle. Section III shows our prediction experiment setups. Section IV evaluates the performance of our text features and prediction models. Threats to validity is presented in Section V. Section VI concludes our work.

## II. VERSION-BASED BUG LIFECYCLE

This section describes our data collection approach, the studied projects, and defines bug lifecycle.

### A. Linking Jira and Github

Our study mainly focuses on data from two sources: bug related data from Jira and code evolution data from Github. What makes Jira becomes our bug reports database is not only

TABLE I: An Overview of 10 Studied Apache Projects.

Project	Description	All	AV	AV%
CXF	Web services framework.	6431	1423	22.13%
Flink	Stream processing framework.	6783	895	13.19%
Flume	Service for efficiently dealing log data.	3167	834	26.33%
Groovy	Object-oriented programming language for Java.	8100	3697	45.64%
Hadoop	Software framework for distributed storage.	12723	6766	53.18%
NiFi	Enables data flow between systems.	4446	959	21.57%
OpenJPA	Implementation of the Java Persistence API specification.	2718	1654	60.85%
PDFBox	Pure-Java library.	2952	2055	69.61%
Tuscan	Developing and running software applications.	4,082	1420	34.79%
Wicket	A lightweight component-based web application framework.	6458	2169	33.59%
Totals		57860	21872	37.80%

the popular it is, but also it provides the **affect-version** field. It is filled by the bug fixing developers and is an indicator of bug introducing time estimated by the development team [16]. Github is a web-based Git and Version Control System(VCS). We can get several collaboration features such as bug tracking, feature requests and task management from it [17].

To synchronize the bug reports from Jira and commits from Github, bug id with some key words such as “bug”, “fix”, “defect” are regarded as the link between a bug reports and commits [18], [19]. Although there are many new technique to link the bug reports with commits such as ReLink [20] and MLink [21], in this paper, a bug report can be linked with a commit only when their bug id is same which ensured the correctness of our data.

### B. The Studied Projects

We choose 10 Apache Software Foundation (ASF) projects as our studied projects. First, they are well-known ASF projects available in Jira and Github. Second, these projects have enough proportion and number of bug reports with affect version. Third, these projects have the enough number of bug reports matching from Jira to Github.

Table I summarizes these 10 ASF projects with their description and the numbers of bug reports. The third column (ALL) is the number of all the bug reports recorded in Jira. The forth column (AV) is the bug reports that have affect version item filled in by developers and the last column (AV%) is the percentage of them comparing to all bug reports.

### C. Definition of bug lifetime

As discussed in Introduction, we define the version-based life cycle  $LC(x)$  of bugs  $x$ , as the version sequence from the introducing version  $V_i$  to the fixing version  $V_f$ :

$$LC(x) = V_i, V_{i+1}, \dots, V_f$$

and the version-length of life cycle  $LLC(x)$  as version difference from  $V_i$  to  $V_f$ .

In our experiment, we use the affect version in Jira as the the introducing version of bugs. Costa et al. [16] applied the affect version on evaluating the approach of identifying

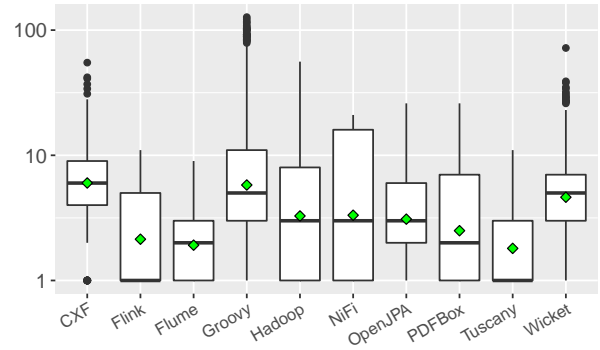


Fig. 1: The boxplots of 10 Projects distribution of the version-length from the affect version to the fixing version.

bug-introducing changes. If a bug has multiple affect versions filled in, we use the earliest affect version as bug introducing version. Jira also provides the fixing version in most bug reports, thus we use the fixing version as the endpoint version of bug lifecycle. Correspondingly, if a bug has multiple fixing versions, we use the latest one.

The AV% column in Table I shows that about 38% bugs have complete lifecycle information (affected version and fixing version). We use these 38% bugs as the studied dataset to construct and evaluate our prediction models. Figure 1 presents the bug’s lifetime statistic in the chosen 10 projects. We observe that:

- in all projects, many bugs live in lifecycle with multiple version-length ( $LLC(x) > 1$ ).
- the boxplots have much difference from project to project, and the quartile are also different which could be used to define the cutoff points in sectionIII.

## III. PREDICTION METHODOLOGY

In this section, we will discuss the approach and evaluation metrics used in the following experiment.

### A. Text Feature Extraction

It is three reasons that we use the text features in the bug lifecycle prediction scenario. First, text features are included in bug reports descriptions or summary and most of the bug reports have these items filled in. Second, there are numerous information that exists in bug report descriptions written by developers. We can extracted features from it to build the prediction model. Third, text feature based approaches have been used in many former researches [22]–[24] in other fields such as defect fixing effort prediction and software constructive cost prediction. Kikas et al. used the number of text comments or size of text comments as text features to predict when a bug will be closed in the future [25].

**Text Feature.** In our work, to generate text features from text descriptions of historical defect reports. We utilize tf-idf (short for term frequency-inverse document frequency) api of Scikit-learn package in Python language to extract word tokens from the textual descriptions.

The tf-idf method consists of two parts tf and idf. The full name of tf is term frequency which is the number of times a term appears in a document. The  $n$  word tokens we extract from a text descriptions can be defined as  $w[0, \dots, n]$ , where  $w_i$  means the  $i$ -th word in the text. tf can be defined as:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}},$$

where the  $n_{i,j}$  means the number of the word  $w_i$  appears in document  $j$ . The  $\sum_k n_{k,j}$  means the summation of document  $j$ 's words.

The full name of idf is inverse document frequency, which gives us the word's frequency across the documents. The formula of idf can be defined as:

$$idf_i = \log \frac{|D|}{|\{j: w_i \in d_j\}|},$$

where the  $|D|$  means the number of documents included in the corpus. The  $|\{j: w_i \in d_j\}|$  means the number of documents the  $w_i$  appears in the corpus.

Finally, the value of tf-idf can be calculated as:

$$tf - idf = tf_{i,j} \times idf_i$$

In information retrieval, tf-idf is a statistic method that is intended to perform a single word's importance of a certain document in a corpus [15]. It is also widely used in information retrieval or text mining as a weighting factor.

**Basic Feature.** We also use the basic metrics from Jira's bug reports as our baseline prediction features. There are many items that Jira provides for developers to fill in. But some of the items most developers (over 99%) did not fill in when they reported a bug, such as Components, Time Spent, Work Ratio, and Security Level. And there are also some items that can not be directly used as features, such as Assignee, Reporter, Creator and Environment. Thus, from our investigation, there are Priority, Votes and Watchers items that can be directly used as features to build the basic feature prediction model.

### B. Prediction Settings

**Two Classification Problem Definition.** Numerous researchers have used the classification model in bug fixing effort prediction [5], [26], [27], which inspires us to build a classification prediction model for version-based bug lifecycle.

In the view of every project's distribution of length of bug lifecycle (LLC) in Figure 1, the boxplots have much difference from project to project, and the three cutoff points (25%,50%,75%) of boxplots are also different. By the cutoff point, we transfer the bug lifecycle prediction to a two classification problem. Figure 2 shows how the transformation executes. In Figure 1, the boxplots of project CXF have three cutoff points: 4(25%), 6(50%), 9(75%). For 4 as 25% cutoff point, we group the bugs whose LLC less than 4 into a class ( $<$  cutoff class) and greater equal than 4 into another class ( $\geq$  cutoff class), then we do the two classification. For each projects, we will do the two classification at different three cutoff points (25%,50%,75%). There are two special cases in Figure 1 that Flink and Tuscany's cutoff points have some

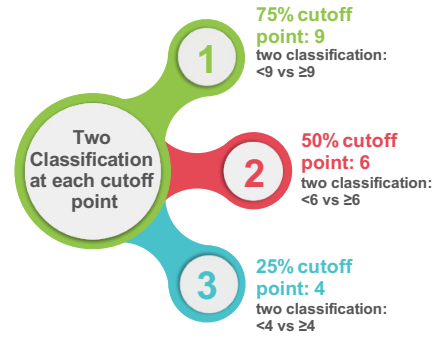


Fig. 2: An example of CXF Project's two classification at 3 Cutoff Points: 75%(9), 50%(6) and 25%(4).

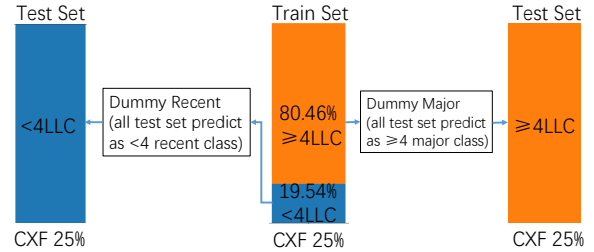


Fig. 3: An example of dummy recent and dummy major classifiers at CXF 25% cutoff point.

overlaps. Thus, if a project's cutoff point 25% equals to 50% or cutoff point 50% equals to 75%, we plus 1 LLC for the latter cutoff point.

**Cross Validation.** Two-fold cross validation is used to train and test the prediction model in the 10 studied projects. For a certain project, we break its data into two folds, one for train set and another for test set. In this experiment, we run two-fold cross validation for 50 times and totally get 100 prediction results for each cutoff point of each project.

**Prediction Classifiers.** To build our bug lifecycle prediction model, we adopt the following four commonly-used supervised classifiers: Naive Bayes(NB), Support Vector Machine(SVM), Logistic Regression(LR), and Random Forest(RF). In our experiments, we use the implementations of these classifiers in scikit-learn, a free software machine learning library for the Python programming language [28].

**Dummy Classifiers.** The dummy classifiers are used as the baseline classifiers in this experiment. A dummy random classifier is that all prediction classes are randomly guessed, which can also achieve a certain prediction accuracy. Another two dummy classifiers are the dummy recent classifier for predicting all bug introduced in recent version class, and the dummy major classifier for predicting all bug reports introduced in the major class of train set. Figure 3 presents an example of dummy recent and dummy major classifiers at CXF 25% cutoff point. For the dummy recent classifier, it predict all the test set as LLC<4 because the recent class is LLC<4. For the major recent classifier, it predict all the test set as LLC $\geq$ 4 because the major class of train set is LLC $\geq$ 4.



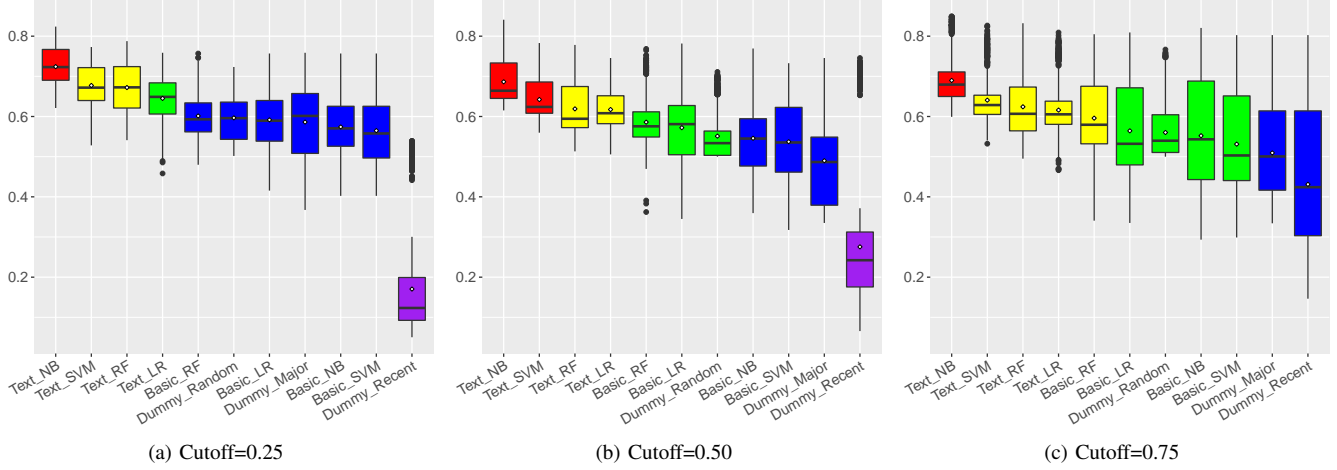


Fig. 4: The boxplots of Weighted Average F-measure values of all studied combination of features and classifiers. Different colors represents different ranks (red>yellow>green>blue>purple) calculated from double Scott-Knott Tests.

**Combination of Features and Classifiers.** Here we make a brief conclusion of different features and classifiers mentioned above: there are 11 combinations of features and classifiers we will use in the following experiment. First, 3 dummy prediction models: Dummy\_Random for randomly guess predict class, Dummy\_Recent for guess all predict class as recent class, Dummy\_Major for guess all predict class as major class. Moreover, using basic metrics as prediction features, there are 4 basic feature based prediction models: Basic\_NB, Basic\_SVM, Basic\_LR, Basic\_RF. Finally, using textual metrics as prediction features, there are 4 text based prediction models: Text\_NB, Text\_SVM, Text\_LR, Text\_RF.

### C. Evaluation

**F-measure.** To evaluate the performance, we use the macro-average measurement [14], [29]. It is also commonly known as the metric weighted average F-measure. The F-measure can be calculated by the Precision and the Recall. For project  $j$ , the F-measure of  $m$ -th class  $F_{m,j}$  can be defined as:

$$F_{m,j} = 2 \times \frac{\text{precision}_{m,j} \times \text{recall}_{m,j}}{\text{precision}_{m,j} + \text{recall}_{m,j}}.$$

Considering the class size, the weighted average F-measure of project  $j$  can be calculated as:

$$F_j = \frac{\sum_m F_{m,j} \times M_{m,j}}{\sum_m M_{m,j}}.$$

where  $M_{m,j}$  is the whole number of bug reports in  $m$ -th class and the  $m$  equals to the class number.

**Scott-Knott Test.** To compare the performance of dummy, basic feature and text feature bug lifecycle prediction models, we use the Scott-Knott test [30]. The Scott-Knott test uses hierarchical cluster analysis to recursively group classification techniques into statistically distinct ranks. In this paper, we use weighted F-measure as the performance measure. If two

groups have statistically significant difference of weighted F-measure, the Scott-Knott will execute again to further divide the ranks. The test terminates when there is no statistically distinct groups can be created [30].

**Cliffs Delta  $\delta$ .** To quantify the improvement of performance on our textual feature bug lifecycle prediction model compared with the baseline models (dummy model and basic feature model), we introduce Cliff's delta  $\delta$  [31]. The improvement magnitude is usually assessed by the thresholds:  $|\delta| < 0.147$  negligible,  $0.147 \leq |\delta| < 0.330$  small,  $0.330 \leq |\delta| < 0.474$  medium,  $|\delta| \geq 0.474$  large.

## IV. PREDICTION RESULT

This section gives the result of our proposed textual feature based bug lifecycle prediction model comparing with the basic feature prediction model and dummy prediction model. First, we do a Scott-Knott test to show the performance of different prediction models in 3 cutoff points 25%, 50% and 75%. Moreover, we give the detail table of the result and calculate the Cliffs delta  $\delta$  to compare the performance of the best method (from SK test it is text Naive Bayes method) with the others in each run of 50 times cross-validation.

**Scott-Knott Test Result.** To address our prediction result, Figure 4 presents an overview of our Scott-Knott test approach in dummy, basic feature and text feature bug lifecycle prediction models in 3 cutoff points 25%, 50% and 75%. We performed a double Scott-Knott test [32] to achieve the goal of generating statistically distinct groups. In the first run, the Scott-Knott test is run over each project and get a rank value for each project of 50 times cross-validation runs. In the second run, we put the Scott-Knott ranks of each project into another Scott-Knott test to get the final statistically distinct ranks of different prediction models.

Figure 4 shows that in each cutoff point, our proposed 4 text feature prediction models (Text\_NB, Text\_SVM, Text\_LR, Text\_RF) performs better than other baseline models.

TABLE II: F-measure means and Effect Sizes for each cutoff point in different projects. Numbers without parentheses are means and those with parentheses are Cliff’s Delta  $\delta$  comparative to Text Naive Bayes Method.

	Project	Dummy			SVM		LR		RF		NB		
		Random	Recent	Major	Basic	Text	Basic	Text	Basic	Text	Basic	Text	
25%	CXF	0.688(-1.00)	0.063(-1.00)	0.721(-1.00)	0.714(-1.00)	0.735(-1.00)	0.714(-1.00)	0.722(-1.00)	0.714(-1.00)	0.753(-0.95)	0.714(-1.00)	<b>0.779</b>	
	Flink	0.535(-1.00)	0.199(-1.00)	0.488(-1.00)	0.450(-1.00)	0.626(-0.93)	0.497(-1.00)	0.569(-0.99)	0.544(-1.00)	0.602(-1.00)	0.453(-1.00)	<b>0.674</b>	
	Flume	0.539(-1.00)	0.499(-1.00)	0.499(-1.00)	0.630(-0.98)	0.655(-0.93)	0.651(-0.97)	0.581(-1.00)	0.615(-1.00)	0.622(-0.99)	0.610(-0.99)	<b>0.722</b>	
	Groovy	0.560(-1.00)	0.161(-1.00)	0.542(-1.00)	0.542(-1.00)	0.653(-1.00)	0.560(-1.00)	0.648(-1.00)	0.584(-1.00)	0.664(-1.00)	0.558(-1.00)	<b>0.696</b>	
	Hadoop	0.572(-1.00)	0.147(-1.00)	0.563(-1.00)	0.480(-1.00)	0.644(-0.88)	0.597(-0.99)	0.594(-1.00)	0.624(-0.99)	0.634(-0.97)	0.618(-0.99)	<b>0.693</b>	
	NiFi	0.690(-1.00)	0.062(-1.00)	0.723(-0.99)	0.635(-1.00)	0.740(-0.88)	0.643(-1.00)	0.723(-0.99)	0.652(-1.00)	0.754(-0.63)	0.635(-1.00)	<b>0.773</b>	
	OpenJPA	0.626(-1.00)	0.100(-1.00)	0.644(-1.00)	0.527(-1.00)	0.691(-1.00)	0.529(-1.00)	0.656(-1.00)	0.542(-1.00)	0.700(-1.00)	0.533(-1.00)	<b>0.753</b>	
	PDFBox	0.507(-1.00)	0.272(-1.00)	0.399(-1.00)	0.607(-0.98)	0.625(-0.92)	0.614(-0.97)	0.616(-0.96)	0.597(-1.00)	0.580(-1.00)	0.570(-1.00)	<b>0.650</b>	
	Tuscany	0.616(-1.00)	0.107(-1.00)	0.631(-1.00)	0.496(-1.00)	0.728(-0.98)	0.541(-1.00)	0.681(-1.00)	0.556(-1.00)	0.723(-0.99)	0.485(-1.00)	<b>0.778</b>	
	Wicket	0.633(-1.00)	0.095(-1.00)	0.653(-1.00)	0.565(-1.00)	0.675(-0.99)	0.570(-1.00)	0.662(-1.00)	0.585(-1.00)	0.689(-0.97)	0.565(-1.00)	<b>0.722</b>	
	Ave.		0.597	0.170	0.586	0.565	0.677	0.592	0.645	0.601	0.672	0.574	<b>0.724</b>
	50%	CXF	0.542(-1.00)	0.186(-1.00)	0.506(-1.00)	0.490(-1.00)	0.607(-1.00)	0.490(-1.00)	0.578(-1.00)	0.545(-1.00)	0.597(-1.00)	0.491(-1.00)	<b>0.661</b>
Flink		0.523(-1.00)	0.223(-1.00)	0.457(-1.00)	0.452(-1.00)	0.610(-0.91)	0.503(-1.00)	0.566(-0.99)	0.550(-1.00)	0.587(-1.00)	0.455(-1.00)	<b>0.655</b>	
Flume		0.672(-1.00)	0.701(-0.99)	0.701(-0.99)	0.673(-1.00)	0.728(-0.90)	0.712(-0.95)	0.702(-0.99)	0.704(-0.98)	0.732(-0.89)	0.705(-0.99)	<b>0.787</b>	
Groovy		0.501(-1.00)	0.357(-1.00)	0.357(-1.00)	0.547(-1.00)	0.610(-1.00)	0.586(-1.00)	0.607(-1.00)	0.572(-1.00)	0.574(-1.00)	0.553(-1.00)	<b>0.643</b>	
Hadoop		0.507(-1.00)	0.274(-1.00)	0.397(-1.00)	0.585(-0.99)	0.618(-0.84)	0.593(-0.99)	0.595(-0.94)	0.577(-1.00)	0.565(-1.00)	0.580(-1.00)	<b>0.653</b>	
NiFi		0.650(-1.00)	0.084(-1.00)	0.675(-0.99)	0.635(-1.00)	0.696(-0.89)	0.641(-1.00)	0.675(-0.99)	0.650(-1.00)	0.709(-0.75)	0.635(-1.00)	<b>0.737</b>	
OpenJPA		0.548(-1.00)	0.177(-1.00)	0.519(-1.00)	0.447(-1.00)	0.622(-1.00)	0.544(-1.00)	0.576(-1.00)	0.560(-1.00)	0.617(-1.00)	0.546(-1.00)	<b>0.694</b>	
PDFBox		0.501(-1.00)	0.312(-1.00)	0.355(-1.00)	0.628(-0.74)	0.631(-0.84)	0.634(-0.69)	0.628(-0.89)	0.609(-0.98)	0.576(-1.00)	0.572(-1.00)	<b>0.649</b>	
Tuscany		0.565(-1.00)	0.156(-1.00)	0.550(-1.00)	0.486(-1.00)	0.699(-0.98)	0.536(-1.00)	0.655(-1.00)	0.552(-1.00)	0.677(-1.00)	0.478(-1.00)	<b>0.747</b>	
Wicket		0.504(-1.00)	0.286(-1.00)	0.383(-1.00)	0.428(-1.00)	0.601(-0.99)	0.482(-1.00)	0.590(-1.00)	0.540(-1.00)	0.557(-1.00)	0.442(-1.00)	<b>0.637</b>	
Ave.			0.551	0.276	0.490	0.537	0.642	0.572	0.617	0.586	0.619	0.546	<b>0.686</b>
75%		CXF	0.502(-1.00)	0.303(-1.00)	0.365(-1.00)	0.380(-1.00)	0.591(-1.00)	0.424(-1.00)	0.587(-1.00)	0.587(-1.00)	0.550(-1.00)	0.394(-1.00)	<b>0.633</b>
	Flink	0.501(-1.00)	0.349(-1.00)	0.350(-1.00)	0.443(-1.00)	0.604(-0.90)	0.498(-1.00)	0.593(-0.93)	0.508(-1.00)	0.549(-1.00)	0.419(-1.00)	<b>0.641</b>	
	Flume	0.721(-1.00)	0.756(-0.96)	0.756(-0.96)	0.751(-0.96)	0.767(-0.90)	0.760(-0.94)	0.756(-0.96)	0.753(-0.96)	0.775(-0.85)	0.772(-0.90)	<b>0.825</b>	
	Groovy	0.612(-1.00)	0.624(-1.00)	0.624(-1.00)	0.655(-1.00)	0.648(-1.00)	0.675(-0.98)	0.639(-1.00)	0.683(-0.96)	0.683(-1.00)	0.707(-0.19)	<b>0.710</b>	
	Hadoop	0.513(-1.00)	0.426(-1.00)	0.426(-1.00)	0.444(-1.00)	0.592(-0.98)	0.497(-1.00)	0.549(-1.00)	0.564(-1.00)	0.562(-1.00)	0.515(-1.00)	<b>0.652</b>	
	NiFi	0.554(-1.00)	0.169(-1.00)	0.530(-1.00)	0.457(-1.00)	0.624(-0.99)	0.474(-1.00)	0.563(-1.00)	0.523(-1.00)	0.613(-1.00)	0.457(-1.00)	<b>0.687</b>	
	OpenJPA	0.530(-1.00)	0.477(-1.00)	0.477(-1.00)	0.530(-1.00)	0.623(-0.99)	0.543(-1.00)	0.586(-1.00)	0.565(-1.00)	0.602(-1.00)	0.579(-1.00)	<b>0.677</b>	
	PDFBox	0.512(-1.00)	0.423(-1.00)	0.423(-1.00)	0.539(-0.99)	0.623(-0.94)	0.601(-0.99)	0.603(-1.00)	0.595(-0.99)	0.576(-1.00)	0.564(-1.00)	<b>0.652</b>	
	Tuscany	0.553(-1.00)	0.170(-1.00)	0.529(-1.00)	0.440(-1.00)	0.693(-0.99)	0.482(-1.00)	0.656(-1.00)	0.495(-1.00)	0.668(-1.00)	0.423(-1.00)	<b>0.747</b>	
	Wicket	0.605(-1.00)	0.615(-1.00)	0.615(-1.00)	0.674(-0.70)	0.640(-1.00)	0.690(-0.29)	0.628(-1.00)	0.686(-0.41)	0.664(-0.95)	0.692(-0.19)	<b>0.697</b>	
	Ave.		0.560	0.431	0.509	0.531	0.641	0.564	0.616	0.596	0.624	0.552	<b>0.692</b>

Moreover, the text Naive Bayes classifier performs the best across every cutoff point. In cutoff points 25% and 75%, text Naive Bayes method has been divided into a single class (the best class) by double Scott-Knott test. Although in 50% cutoff point, text Naive Bayes method is not grouped into a class individually, but it also performs best comparing to the other methods. Thus, in the following table, we use text Naive Bayes method as the chosen method in Cliffs delta  $\delta$  calculation.

**Detail Result with Cliffs Delta  $\delta$ .** The detail result of weighted F-measure is presented in table II, which is calculated by 50 times 2-fold cross-validation. The first column is the class cutoff point. The second column is the 10 projects we used in this experiment. The rest columns present the performance of the bug lifecycle prediction model in 3 dummy classifiers and 4 traditional classifiers (text and basic features).

In general, all text feature based classifiers can lead a weighted F-measure higher than 0.6. The text Naive Bayes method outperforms the related methods in every project and average value. For example, for predicting if the bug lifecycle is greater or less than 25% cutoff point of CXF’s LLC, our text based Naive Bayes method achieves the weighted F-measure 0.779, which is higher than the other methods’ results. In the view of all average results of 3 cutoff points, text based methods are all better than correlated basic methods and dummy methods. Although the basic feature methods perform better than other classifiers with text features in some projects, they can not exceed the text Naive Bayes method. In the view of Cliffs delta  $\delta$ , most of the methods performs large magnitude ( $|\delta| \geq 0.474$ ) compared to text Naive Bayes method, which means the text Naive Bayes method performs

significantly better than other methods in each run of 50 times cross-validation of each project.

**We observe that the text based methods perform better than basic methods and dummy methods in average of 10 studied projects. The text based Naive Bayes bug lifecycle prediction model outperforms all other prediction models.**

## V. THREATS TO VALIDITY

This study provides a text feature based prediction model to predict the bug lifecycle. However, there are some threats to validity that should be taken into consideration.

**Internal validity.** All the ten data sets used in our experiments are well-known open source projects of Apache Software Foundation in Jira. The performance of bug lifecycle prediction in commercial projects may be different from the open source projects. Moreover, the ten open source projects are all Java projects where other programming language may have some differences in the character of text feature. We will study more projects in the future, including the commercial projects and the projects in other languages.

**External validity.** Our bug lifecycle prediction model requires that the projects have sufficient historical bug reports with affect version to train the prediction model. However, these information could be limited in some projects. Another external validity of our experiment is that the bug reports provide is the affect version not the exact bug introducing time. The commercial data sets may be more accurate in this bug introducing time compared to the open source projects.

**Reliability validity.** All the ten projects are publicly available in Apache Software Foundation of Jira. Any researchers who intend to replicate this study can get the data sets in

Apache Software Foundation of Jira. Moreover, the python implementation of our experiment will be provided online.

## VI. CONCLUSION

Bug lifecycle prediction aims to know how long does a bug exists in software during it first be introduced and finally be fixed. It can help developers to efficiently revisit the bug introducing code. The former work mainly focus on fixing period or dormant period, and use features obtained from defect ranking, source code and CVS log to do the analysis. In this study, we consider the bug lifecycle as a whole and focus on text features (summary and description of bugs) in bug reports to construct prediction model.

Our evaluation on 10 well-know Apache Software Foundation projects with 57000+ bugs shows that our model achieves a good performance in predicting the bug lifecycle. We examine the results with 3 evaluation measurements: Weighted F-measure, Scott-Knott test, and Cliffs Delta  $\delta$ . The Scott-Knott test and Cliffs Delta  $\delta$  present that the text Naive Bayes method outperforms all the rest methods. In general, all text feature based classifiers can lead a weighted F-measure higher than 0.6. In the future, we intend to extend our text feature based model to more projects in bug lifecycle prediction and introduce more methods to train the prediction model.

## ACKNOWLEDGEMENT

This work was supported by the Natural Science Foundation of Jiangsu Province of China (Grant No. BK20140611), the National Natural Science Foundation of China (Grant No. 61403187, 61772259, 61472175, 61472178, 61772263, 61472077).

## REFERENCES

- [1] L. Erlikh, "Leveraging legacy system dollars for e-business," *It Professional*, vol. 2, no. 3, pp. 17–23, 2000.
- [2] S. Kim, T. Zimmermann, E. J. W. Jr., and A. Zeller, "Predicting faults from cached history," in *Proceedings of the 1st Annual India Software Engineering Conference*, 2008, pp. 15–16.
- [3] T. Lee, J. Nam, D. Han, S. Kim, and H. P. In, "Micro Interaction Metrics for Defect Prediction," in *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*, Szeged, Hungary, 2011, pp. 311–321.
- [4] J. Wang, B. Shen, and Y. Chen, "Compressed C4.5 models for software defect prediction," in *Proceedings of 12th International Conference on Quality Software*, 2012, pp. 13–16.
- [5] W. Abdelmoez, M. Kholief, and F. M. Elsalmy, "Bug fix-time prediction model using naive bayes classifier," in *International Conference on Computer Theory and Applications*, 2012, pp. 167–172.
- [6] Y. Liu, Y. Li, J. Guo, Y. Zhou, and B. Xu, "Connecting software metrics across versions to predict defects," in *25th IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, 2018, pp. 232–243.
- [7] E. J. Weyuker, T. J. Ostrand, and R. M. Bell, "Using developer information as a factor for fault prediction," in *Third International Workshop on Predictor Models in Software Engineering*, 2007, pp. 8–8.
- [8] V. S. Sinha, S. Sinha, and S. Rao, "Buginnings: identifying the origins of a bug," in *India Software Engineering Conference*, 2010, pp. 3–12.
- [9] T. H. Chen, M. Nagappan, E. Shihab, and A. E. Hassan, "An empirical study of dormant bugs," in *Working Conference on Mining Software Repositories*, 2014, pp. 82–91.
- [10] R. K. Saha, S. Khurshid, and D. E. Perry, "An empirical study of long lived bugs," in *Software Evolution Week - IEEE Conference on Software Maintenance, Reengineering and Reverse Engineering*, 2014, pp. 144–153.
- [11] P. Bhattacharya and I. Neamtii, "Bug-fix time prediction models: Can we do better?" in *Working Conference on Mining Software Repositories*, 2011, pp. 207–210.
- [12] F. Zhang, F. Khomh, Y. Zou, and A. E. Hassan, "An empirical study on factors impacting bug fixing time," in *19th Working Conference on Reverse Engineering*, 2012, pp. 225–234.
- [13] Q. Song, M. Shepperd, M. Cartwright, and C. Mair, "Software defect association mining and defect correction effort prediction," *IEEE Transactions on Software Engineering*, vol. 32, no. 2, pp. 69–82, 2006.
- [14] H. Zhang, L. Gong, and S. Versteeg, "Predicting bug-fixing time: An empirical study of commercial software projects," in *International Conference on Software Engineering*, 2013, pp. 1042–1051.
- [15] A. Rajaraman and J. D. Ullman, *Mining of massive datasets*. Cambridge University Press, 2012.
- [16] D. A. D. Costa, S. Mcintosh, W. Shang, U. Kulesza, R. Coelho, and A. Hassan, "A framework for evaluating the results of the szz approach for identifying bug-introducing changes," *IEEE Transactions on Software Engineering*, vol. 43, no. 7, pp. 641–657, 2017.
- [17] G. Gousios, B. Vasilescu, A. Serebrenik, and A. Zaidman, "Lean gitorrent: Github data on demand," in *Working Conference on Mining Software Repositories*, 2014, pp. 384–387.
- [18] T. Zimmermann and A. Zeller, "When do changes induce fixes?" in *International Workshop on Mining Software Repositories*, 2005, pp. 1–5.
- [19] S. Kim, T. Zimmermann, K. Pan, and E. J. Whitehead, "Automatic identification of bug-introducing changes," in *IEEE/ACM International Conference on Automated Software Engineering*, 2006, pp. 81–90.
- [20] R. Wu, H. Zhang, S. Kim, and S. C. Cheung, "Relink: recovering links between bugs and changes," in *ACM Sigsoft International Symposium on the Foundations of Software Engineering*, 2011, pp. 15–25.
- [21] A. T. Nguyen, T. T. Nguyen, H. A. Nguyen, and T. N. Nguyen, "Multi-layered approach for recovering links between bug reports and fixes," in *ACM Sigsoft International Symposium on the Foundations of Software Engineering*, 2012, pp. 1–11.
- [22] C. Weiss, R. Premraj, T. Zimmermann, and A. Zeller, "How long will it take to fix this bug?" in *International Workshop on Mining Software Repositories*, 2007, pp. 1–1.
- [23] U. Raja, "All complaints are not created equal: text analysis of open source software defect reports," *Empirical Software Engineering*, vol. 18, no. 1, pp. 117–138, 2012.
- [24] A. Said, M. Borg, and D. Pfahl, "Using text clustering to predict defect resolution time: a conceptual replication and an evaluation of prediction accuracy," *Empirical Software Engineering*, vol. 21, no. 4, pp. 1437–1475, 2016.
- [25] R. Kikas, M. Dumas, and D. Pfahl, "Using dynamic and contextual features to predict issue lifetime in github projects," in *Mining Software Repositories*, 2016, pp. 291–302.
- [26] E. Giger, M. Pinzger, and H. Gall, "Predicting the fix time of bugs," in *International Workshop on Recommendation Systems for Software Engineering*, 2010, pp. 52–56.
- [27] L. Marks, Y. Zou, and A. E. Hassan, "Studying the fix-time for bugs in large open source projects," in *International Conference on Predictive MODELS in Software Engineering*, 2011, pp. 1–8.
- [28] F. Pedregosa, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, and J. Vanderplas, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. 10, pp. 2825–2830, 2012.
- [29] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques (Third Edition)*. Morgan Kaufmann Publishers Inc., 2011.
- [30] E. Jelihovschi, J. Faria, and I. Allaman, "Scottknott: a package for performing the scott-knott clustering algorithm in r," *Tema*, vol. 15, no. 1, pp. 3–17, 2014.
- [31] J. Romano, J. D. Kromrey, J. Coraggio, J. Skowronek, and L. Devine, "Exploring methods for evaluating group differences on the nsse and other surveys: Are the t-test and cohen's d indices the most appropriate choices?" in *Annual meeting of the Southern Association for Institutional Research*, 2006, pp. 1–24.
- [32] B. Ghotra, S. McIntosh, and A. E. Hassan, "Revisiting the impact of classification techniques on the performance of defect prediction models," in *Proceedings of the 37th International Conference on Software Engineering-Volume 1*, 2015, pp. 789–800.

# Bayesian Logistic Regression for software defect prediction

Jinu M Sunil  
BITS Pilani Hyderabad Campus  
Hyderabad, India  
f20130423@hyderabad.bits-pilani.ac.in

Lov Kumar  
BITS Pilani Hyderabad Campus  
Hyderabad, India  
lovkumar@hyderabad.bits-pilani.ac.in

N L Bhanu Murthy  
BITS Pilani Hyderabad Campus  
Hyderabad, India  
bhanu@hyderabad.bits-pilani.ac.in

**Abstract**—Timely identification of bugs plays an important role in delivering quality software. Defect prediction models help to detect or rank the defect prone files so that the project management team can allocate resources diligently or may seek help from external sources to enable rigorous quality assurance activities on defect prone files. Though defect prediction models have been built using several machine learning algorithms, Bayesian approach of these models is not explored. We propose Bayesian logistic regression with non-informative and informative priors to build defect prediction models. We seek to study if there are any advantages of using Bayesian logistic regression over logistic regression and the role of priors in the performance of Bayesian logistic regression. A comparative study of the performance of Bayesian logistic regression with other widely known classifiers is also presented.

**Index Terms**—Bayesian regression, Informative priors

## I. INTRODUCTION

Delivering quality software is one of the most important goals of any IT vendor to survive in the highly competitive market. Bugs<sup>1</sup> or defects surfaced during any time in the evolution of a software, especially in the post-production phase, will be detrimental to the software quality. In practice, defects get uncovered during post-production phase in spite of expending good number of man hours for quality assurance activities like different kinds of testing and code reviews. Researchers and practitioners proposed the methodologies to curtail these defects. Software Defect Prediction is one such technique to predict defects, preferably before rolling to production environment, where in the deeper assurance activities like code reviews and testing by Subject Matter Experts (SMEs) can be performed on these defect prone files. The defect prediction models have been developed using machine learning algorithms like logistic regression, Naive Bayes classifier and Random forest etc. Logistic Regression is a probabilistic discriminative model where in probabilities of positive and negative class are modeled by sigmoid function.

$$P(Y = 1|w, x) = 1/(1 + e^{-w^T x}) \quad (1)$$

It gives the probability that a class/file is defective, given the feature vector of a file  $x$  and parameter vector  $w$ . Suppose there are  $n$  observations where the  $i^{th}$  observation is a tuple  $(x_i, y_i)$ ,  $x_i$  is a  $m$  dimensional feature vector and  $y_i$  is boolean valued representing whether the file is defective (1) or not (0).  $D = \{(x_1, y_1), (x_2, y_2) \dots (x_n, y_n)\}$  is the set of all training instances.

The following equality is obtained from bayes theorem.

$$P(w|D) = P(D|w)P(w)/P(D) \quad (2)$$

where  $P(w|D)$  is posterior distribution,  $P(D|w)$  is likelihood and  $P(w)$  is the prior distribution.

$P(D)$  is a constant for a given  $D$  and hence

$$P(w|D) \propto P(D|w)P(w) \quad (3)$$

<sup>1</sup>bug and defect are used synonymously throughout this paper

In logistic regression, it is assumed that  $w$  follows uniform distribution and hence any tuple in  $R^{m+1}$  ( $m$  is the size of input vector) is a candidate solution and hence

$$P(w|D) \propto P(D|w) \quad (4)$$

$y_i$ 's are conditionally independent hence we can write equation 4 as:

$$P(w|D) \propto \prod_1^n P(y_i|x_i, w) \quad (5)$$

$$\text{Let } p_i = P(y_i = 1|x_i, w) = \frac{1}{1 + e^{-w^T x_i}}$$

$$\therefore P(w|D) \propto \prod_1^n P(y_i|x_i, w) = \prod_1^n p_i^{y_i} * (1 - p_i)^{1 - y_i} \quad (6)$$

The optimal  $w$  for logistic regression is determined as the maximum likelihood estimator (MLE) of the given training data.

$$w_{MLE} = \text{argmax}_w \left( \prod_1^n p_i^{y_i} * (1 - p_i)^{1 - y_i} \right) \quad (7)$$

There are some drawbacks of logistic regression when applied to defect prediction problem and they are listed below.

- 1) MLE's are asymptotically unbiased, i.e  $E(w_{mle}) \approx w_{true}$  as  $n$  (number of instances/samples) becomes very large. It has been shown that regression coefficients are biased for small and moderate sample sizes [1]. Long et al. [2] offers a rough heuristic about appropriate sample sizes: it is risky to use MLE with samples smaller than 100, while samples larger than 600 seem adequate. For 44 datasets that are used in our study, the average number of files in a project are 400.
- 2) Logistic regression works well when both non-event (no bug) and event (bug) occur in the same ratio and the bias is substantial for small and medium samples with skewed ratio [3]. For defect prediction problem, generally the files are skewed towards non-event (non-defective) and the number of files in each of the project considered in our study is not high.

Logistic regression as stated above assumes uniform priors, but in practice priors can follow any distribution. In Bayesian Logistic regression a prior for  $w$  is ascertained from previous data or expert opinion. Consider the case where the prior is a normal distribution  $w \approx N(\mu, C)$  with mean  $\mu$  and covariance  $C$ . The posterior distribution is.

$$P(w|D) \propto P(D|w) * P(w) \quad (8)$$

$$P(w|D) \propto \left( \prod_1^n p_i^{y_i} * (1 - p_i)^{1 - y_i} \right) * \frac{\exp\left(\frac{-(w-\mu)^T C^{-1} (w-\mu)}{2}\right)}{\sqrt{\det(2\pi C)}} \quad (9)$$

It has been shown that Bayesian Logistic Regression improves prediction accuracy of learning models [4], [5]. Though there are

several learning models developed to predict defects, the Bayesian perspective of the models have not been studied so far. We study Bayesian Logistic Regression and attempt to answer the following research questions:

- **RQ1:** Is there any significant difference between Logistic Regression and Bayesian Logistic Regression?
- **RQ2:** Study informative and non-informative priors of  $w$  and test whether there is any significant difference between the posterior probabilities of the classifiers developed using these different priors?
- **RQ3:** Is there significant difference between the Bayesian Logistic Regression and other widely used classifiers for defect prediction problem?

The rest of the paper is organized as follows. In section II related work is discussed. The metrics and datasets used in this study are discussed in section III. A description of the bayesian logistic regression model is given in section IV. The experimental setup is described in section V. The results are discussed in section VI. Threats to validity of the experiments are discussed in section VII. Conclusions are presented at the end.

## II. RELATED WORK

Bug Prediction models have been built using various metrics of a software. Chidamber and Kermer (CK), Halstead metrics are well known metrics used in defect prediction. However it has been shown that lines of code (LOC) have strong correlation with proneness to defect. Zhang et al. [6] concluded that LOC increases bug proneness. CK metrics have been useful in detecting bugs ([7], [8]). Thomas et al. [9] explored the relationship between code complexity and defect proneness and arrived at the conclusion that complexity metrics are very useful in defect prediction and also code bases that undergo a lot of evolution are prone to defects. These metrics are used in classification algorithms.

Bug detection algorithms are well studied. L. Guo et al. [10] applied random forests to predict fault prone modules and found that random forests were better when applied to large datasets. Neural networks for bug prediction was explored by Rajni Jindal et al. [11] they used ROC to interpret the results obtained. Czibula et al. [12] used unsupervised algorithms for bug prediction and their rule mining based approach resulted in as good results as other algorithms. Relevant to this study Rakesh et al. [13] experimented with a bayesian approach to predict number of bugs in a software and showed that incorporating prior information gave beter results.

A study of benchmarking various algorithms was done by Stefan et al. [14]. They conducted a freidman neymni with AUC-ROC as their measure and showed that random forest outperformed all other models but not significantly. Also around 17 models including random forest, logistic regression, naive bayes and multi layer perceptron were shown to have no significant difference.

## III. METRICS AND DATASET

**TABLE II** gives a description of software projects considered. Only software projects with three or more versions are considered. The data of these 12 projects is extracted from the Promise repository [15]. The average number of classes over all projects and versions is 400. The metrics considered for this study are given in **TABLE I**. Jurecko and Madeyski [21] give a detailed description of all the metrics.

**TABLE I**  
**METRICS**

Author	Metric
Chidamber and Kemerer [16]	Weighted methods per class (WMC) Depth of Inheritance Tree (DIT) Depth of Inheritance Tree (DIT) Coupling between object classes (CBO) Response for a Class (RFC) Lack of cohesion in methods (LCOM)
Bansiy and Davis [17]	Number of Public Methods (NPM) Data Access Metric (DAM) Measure of Aggregation (MOA) Measure of Functional Abstraction (MFA) Cohesion Among Methods of Class (CAM)
Tang et al. [18]	Inheritance Coupling (IC) Coupling Between Methods (CBM) Average Method Complexity (AMC)
Martin [19]	Afferent couplings (Ca) Efferent couplings (Ce)
McCabe [20]	cyclomatic complexity (CC)

## IV. METHODOLOGY

As discussed in the first section the prior  $p(w)$  need not necessarily be a unifrom distribution but can be any distribution. There are two kinds of priors informative and non-informative priors. Informative priors can be derived from historical data or domain expertise whereas Non-informative priors do not rely on previous data but can be approximated by making use of training data.

In this study we make use of a Non-informative prior—Jeffery prior and also propose couple of informative priors that follow a multivariate normal distribution.

### A. Choice of Priors

- **Bayes-1 prior** - This is an **informative** prior. The prior of the current version is the posterior of the previous version of the same software. For example consider the project ivy as described in table II. Ivy has 3 versions, initially assume an isotropic normal distribution  $w \approx N(0, \alpha^{-1}I)$  as prior, and obtain posterior with ivy 1.1 as the data

$$p(w|D_{ivy1.1}) = p(D_{ivy1.1}|w) * N(0, \alpha^{-1}I)$$

. A normal approximation of the above posterior distribution will be used as prior for ivy 1.4

$$p(w|D_{ivy1.4}) = p(D_{ivy1.4}|w) * p(w|D_{ivy1.1})$$

- **Bayes-2 prior** - This is also an **informative** prior and is best explained with an example. Consider again ivy described in **TABLE II**,

- 1) Choose 66% instances of ivy 1.1 randomly let it be  $D_{ivy1.1} = 66\%$  of  $ivy1.1$ . Train a logistic regression with this data, the resulting parameters are  $w_1 = \text{argmax}_w(P(D_{ivy1.1}|w))$
- 2) Choose 66% instances of ivy 1.4 randomly let it be  $D_{ivy1.4} = 66\%$  of  $ivy1.4$ . Train a logistic regression with this data, the resulting parameters are  $w_2 = \text{argmax}_w(P(D_{ivy1.4}|w))$

TABLE II  
12 PROMISE PROJECTS

Project	Version	No. of classes	% of Bugs
Ant	1.3	125	16
	1.4	178	22.4
	1.5	293	10.9
	1.6	351	26.2
	1.7	745	22.2
Camel	1	339	3.8
	1.2	608	35.5
	1.4	872	16.6
	1.6	965	19.48
Ivy	1.1	111	56.7
	1.4	241	6.6
	2	352	11.3
Jedit	3.2	272	33.09
	4	306	24.53
	4.1	312	25.31
	4.2	367	13.08
	4.3	492	2.23
Log-4j	1	135	25.18
	1.1	109	33.94
	1.2	205	92.19
Lucene	2	195	46.67
	2.2	247	58.3
	2.4	340	59.7
	2.5	385	64.41
Poi	2	314	11.78
	2.5	385	64.41
	3	442	63.57
Synapse	1	157	10.19
	1.1	222	27.03
	1.2	256	33.59
Velocity	1.4	196	77
	1.5	214	66.35
	1.6	229	34.06
	1.6	229	34.06
Xalan	2.4	723	15.21
	2.5	803	48.19
	2.6	885	46.44
	2.7	909	98.79
Xerces	1.2	440	16.14
	1.3	453	15.23
	1.4	588	74.32
PC	1	735	8.3
	2	1493	1.2
	3	1099	12.5
	4	1379	12.9

- 3) Repeat the above 2 steps around 30 times, giving 60 parameter vectors—30 from ivy 1.1 and 30 from ivy 1.4.
- 4) Find the best fit multivariate normal for these 60 points
- 5) Use this multivariate normal as prior for ivy 2.0

- **Jefferey Prior**—The jefferey prior is a 'non-informative' prior (uniform priors are also non-informative).  $p(w) \propto \sqrt{\det(I(w))}$ , where  $I(w)$  is the fisher information matrix. The fisher information matrix in the case of logistic regression is  $I_{i,j}(w) = -E\left[\frac{\delta^2 \ln(\text{Likelihood})}{\delta w_i \delta w_j}\right]$  ( $w_i$  and  $w_j$  are the  $i^{\text{th}}$ ,  $j^{\text{th}}$  component of vector  $w$ .  $E[\cdot]$  is expected value). The posterior distribution becomes.

$$P(w|D) \propto P(D|w) \sqrt{\det(I(w))} \quad (10)$$

It was shown by Firth [22] that this choice of prior reduces first order bias in the case of logistic regression.

### B. Sampling from the Posterior

In logistic regression we get a point estimate  $w_{MLE}$  and in the bayesian approach a set of samples are drawn from the posterior

$P(w|D)$ . For any testing example, average of all probabilities would be taken into consideration to classify the testing instance. Suppose  $k$  samples are drawn  $w_1, w_2 \dots w_k$  from  $P(w|D)$  then the probability that a file with attribute  $x$  is defective would be.

$$P(y = 1|x, D) = \frac{1}{k} \sum_1^k \frac{1}{1 + e^{-w_i^T x}} \quad (11)$$

We discuss below the two ways of drawing samples from posterior  $p(w|D)$ .

**Laplace Approximation:** The posterior is approximated to a multivariate normal distribution with mean  $w_{max}$  and covariance matrix  $K$  where  $w_{max} = \text{argmax}_w(p(w|D))$  and  $K = H^{-1}$ ,  $H$  is the hessian of  $p(w|D)$  computed at  $w_{max}$ . Samples are drawn from this approximation. Laplace approximation works well when the pdf is unimodal otherwise it might be a bad approximation [23] and hence we have not used it in this study.

**MCMC:** The other way of drawing samples from the posterior is the famous Markov Chain Monte Carlo (*MCMC*) which involves constructing a markov chain with desired distribution as its equilibrium distribution. In this study we use gibbs sampling which is a MCMC method, to draw samples from the posterior.

Among these prior choices using jefferey priors did not improve performance the other two priors improved performance over simple logistic regression significantly.

## V. EXPERIMENTS

### A. Cost Model

Normalized Expected Misclassification Cost (NEMC) is used to compare performance of the models. Type 1 error is predicting a non-defective file as defective and Type 2 error is predicting a defective file as non-defective. For the defect prediction problem it is evident that type 2 errors are costlier than type 1 errors. The cost ratio  $\beta$  is defined as.

$$\beta = \frac{\text{cost of Type 2 error}}{\text{cost of Type 1 error}} \quad (12)$$

False Positive Rate and False Negative Rate are defined as.

$$E_1 = \frac{FP}{TN + FP} \quad (13)$$

$$E_2 = \frac{FN}{TP + FN} \quad (14)$$

where FP-false positive, TN-true negative, FN-false negative and TP-true positive

$$NEMC = \beta * (E_2 * P_{df}) + (E_1 * P_{ndf}) \quad (15)$$

Where  $P_{df}$  and  $P_{ndf}$  are prior probabilities of defective and non-defective files in the dataset respectively. We refer the reader to Khoshgoftaar et al. [24] for a detailed derivation of NEMC.

### B. Implementaion settings

To answer RQ1 and RQ2, we implement logistic regression, Bayesian logistic regression with Bayes-1 Prior, Bayes-2 Prior and Jeffery Prior. We also implement Random Forest, Nave Bayes and SVM for comparative study of these classifiers with Bayesian logistic regression to answer RQ3. All models are implemented in R-programming [25] and details are discussed below:

- Bayes 1 and Bayes 2: Logit function of the Bayes Logit package [26] is used to implement both the bayesian logistic regression methods. The function requires Data, prior mean and

prior variance as input. Logit function returns samples from the posterior – 1000 points were sampled after a burn-in of 1000.

- Jeffrey Prior (JP): logistf [27] package is used to implement logistic regression with jeffrey prior. The package computes the fisher matrix and also returns a set of samples from the posterior.
- Random forest (RF): randomforest [28] function is used to implement random forest, the ntree variable was set to 100 after experimentation.
- Naive bayes (NB): naiveBayes [28] function is used to implement naive bayes algorithm, this function requires data to be un-normalized.
- Logistic regression (LR): glmnet [29] is used to implement logistic regression algorithm, the regularization parameter was computed using cross-validation.
- svm, l-svm: svm [28] function is used to implement both the svm's. l-svm kernel is linear whereas the kernel for svm is radial.

### C. Training and Testing

Each model is trained on 66% of the last version of each of the software projects mentioned in TABLE II and tested on the remaining 34% of the same project. For the bayesian methods the priors come from the previous version of a project. Consider the ivy project for elaboration of the training-testing process.

- Train RF, LR, NB, svm and l-svm on 66% of ivy 2.0 (last version of ivy)
- Train JP, Bayes 1 and Bayes 2 on 66% of ivy 2.0 and the priors are approximated from ivy 1.1 and ivy 1.4.
- Test all the models on the remaining 34% of ivy 2.0, Compute NEMC on the testing data of ivy 2.0 for all models.

NEMC is used to compare performance of classifiers as discussed in the previous section. NEMC for each model is averaged over several runs (R). For example consider random forest (RF) and ivy 2.0

**for**  $i$  in  $1 \rightarrow R$  **do**

$D_{train} = 66\%$  of ivy2.0

$D_{test} = ivy2.0 - D_{train}$

train RF on  $D_{train}$

compute NEMC for RF over  $D_{test}$ , let it be  $np_i$

**end for**

avg NEMC for RF =  $np_{RF} = \frac{\sum_1^R np_i}{R}$

The above algorithm is used to compute average NEMC for all 8 models.

### D. Statistical Tests for comparing classifiers

Classifiers are compared based on average NEMC and statistical tests used in our study are discussed below:

- **Two sample test – Wilcoxon signed rank test and rank test**  
In both cases the null hypothesis is

$H_0$  : The two models have the same performance

and the alternative

$H_a$  : The two models have different performance

$H_0$  is rejected if  $p_{value} < 0.05$ . For example, consider TABLE III to compare Bayes 2 and Logistic Regression (LR) at  $\beta = 5$ . Here two samples are NEMC values of Bayes-2 (column 2) and LR (column 3)

- **K sample test – Friedman test**

Here the null hypothesis is

$H_0$  : all classifiers perform alike

and the alternative

$H_1$  : atleast 2 classifiers differ in performance

Friedman test assigns a mean rank to each classifier. And if the null hypothesis is rejected a **Nemenyi** test is used to compare all classifiers. The mean rank of two classifiers have to differ by a critical difference (CD) for them to be considered significantly different.

$$CD = q_{\alpha; \infty; L} \sqrt{\frac{L(L+1)}{12K}} \quad (16)$$

where L is the number of classifiers and K is the number of projects.

## VI. RESULTS AND DISCUSSION

Type 2 error is much costlier than Type 1 Error for defect prediction problem and the ratio of two costs ( $\beta$ ) is an important parameter in the performance measure NEMC. The cost ratio ( $\beta$ ) varies with the type of project and organization. We consider the cost ratio to be 5, 10, 20, 40 and conduct experiments to answer RQ1 through RQ3.

For each of the project and cost ratio, learning models are developed using logistic regression and Bayesian logistic regression with Bayes-1 Prior, Bayes-2 Prior and Jeffery prior. The average NEMC over 1000 runs for each of the project are tabulated in Table III, IV, V and VI respectively. Wilcoxon signed rank test and rank test is performed to check whether there is any significant difference between logistic regression and Bayesian logistic regression (RQ1).

For Bayesian logistic regression with Bayes-1 prior and Bayes-2 prior, the null hypothesis for Wilcoxon signed rank test (WSR) and rank test (RT) is rejected as  $p_{values}$  are considerably less than 0.05. Hence the Bayesian logistic regression with informative priors is significantly better than logistic regression. The  $p_{values}$  for different values of  $\beta$  are tabulated in Table VIII. The average number of data points, 400 for our study, is low and could lead to biased estimates and this might be one reason for lower performance of logistic regression. Also, the imbalance of defective files and non-defective files could be another reason.

However it is found out that there is no significant difference between the logistic regression and Bayesian logistic regression with Jeffery prior. Jeffery prior is non-informative prior like uniform prior and it is not generated from historical data and this could be one of the reasons for downplay of Bayesian logistic regression with Jeffery prior. Supporting the argument, it is also observed that there is a significant difference between the performances of Bayesian logistic regression with informative priors and that of non-informative priors (RQ2).

After observing significantly better performance of Bayesian Logistic regression (with informative priors) as compared to logistic regression, the obvious next step is to rank its performance with other widely used classifiers like Random Forest, Nave Bayes, SVM etc (RQ3). We have not considered Bayesian logistic regression with Jeffery prior in this comparative study as it is not significantly different from Logistic Regression (LR). We have conducted Friedman test to check whether there is any significant difference between these classifiers - Bayes 1, Bayes 2, RF, LR, NB, L-SVM, SVM. It is observed that there is significant difference between the performances of these classifiers. Neymni test has been conducted to rank these classifiers and the results are reported in Table VII. Random Forest

is ranked higher than all classifiers and this result is in tune with other comparative studies [14], [30], [31]. The random forest (RF) is an ensemble classifier and for reasons mentioned in [14], it ranks higher than other classifiers. Bayesian logistic regression with Bayes-2 prior and Bayes-1 prior rank 2<sup>nd</sup> and 3<sup>rd</sup> respectively but with no significant difference from Random Forest (RF).

TABLE III  
NORMALIZED PENALTY AT  $\beta = 5$

$\beta = 5$	Bayes1	Bayes2	LR	L-SVM	NB	RF	SVM
POI	0.6505	0.7140	0.8952	0.7919	1.9743	<b>0.597</b>	0.6954
ANT	0.7197	0.6946	0.8072	0.7737	<b>0.609</b>	0.6569	0.8243
CAMEL	0.9143	0.8947	0.9551	0.9962	<b>0.8192</b>	0.8540	0.9916
XALAN	0.0268	0.0360	0.0560	0.0114	0.9149	<b>0.0102</b>	0.0110
JEDIT	0.1403	0.1423	0.1525	0.1308	0.2900	0.1301	<b>0.1276</b>
PC	0.0050	<b>0.0047</b>	0.0055	0.0057	0.0056	0.0058	0.0070
IVY	0.7132	0.7169	0.7107	0.7310	<b>0.6004</b>	0.7160	0.8256
LOG4J	0.2809	0.4614	0.5059	0.1459	2.3786	0.0971	<b>0.0732</b>
LUCENE	0.9301	<b>0.7724</b>	1.1920	1.0087	1.8144	0.7732	0.9008
SYANAPSE	0.9720	0.9738	1.1104	1.0479	0.9182	<b>0.812</b>	1.0251
VELOCITY	0.9461	0.9618	1.1267	1.0334	1.2760	<b>0.8941</b>	1.2530
XERCES	0.3294	0.3415	0.5492	0.3436	1.5594	<b>0.1671</b>	0.4893

TABLE IV  
NORMALIZED PENALTY AT  $\beta = 10$

$\beta = 10$	Bayes1	Bayes2	LR	L-SVM	NB	RF	SVM
POI	1.1383	1.3174	1.6982	1.4682	3.9051	<b>1.0939</b>	1.2691
ANT	1.3936	1.3215	1.5855	1.5138	<b>1.12</b>	1.2534	1.6114
CAMEL	1.8034	1.7376	1.8962	1.9867	<b>1.5753</b>	1.6650	1.9760
XALAN	0.0451	0.0654	0.1027	0.0118	1.8270	0.0151	<b>0.011</b>
JEDIT	0.2590	0.2625	0.2732	0.2553	0.3729	<b>0.2532</b>	0.2551
PC	0.0099	<b>0.0093</b>	0.0110	0.0114	0.0111	0.0116	0.0140
IVY	1.3894	1.3707	1.3932	1.4399	<b>1.1048</b>	1.4087	1.6490
LOG4J	0.4954	0.8631	0.9555	0.2199	4.7415	0.1307	<b>0.0732</b>
LUCENE	1.6974	<b>1.3707</b>	2.2785	1.8765	3.5797	1.3730	1.6320
SYANAPSE	1.8489	1.8511	2.1542	2.0187	1.7318	<b>1.5253</b>	1.9739
VELOCITY	1.7829	1.8181	2.1713	1.9712	2.4785	<b>1.6755</b>	2.4385
XERCES	0.6008	0.6251	1.0521	0.6393	3.0898	<b>0.2866</b>	0.9008

TABLE V  
NORMALIZED PENALTY AT  $\beta = 20$

$\beta = 20$	Bayes1	Bayes2	LR	L-SVM	NB	RF	SVM
POI	2.1137	2.5242	3.3042	2.8209	7.7667	<b>2.0876</b>	2.4164
ANT	2.7414	2.5753	3.1420	2.9941	<b>2.1419</b>	2.4464	3.1856
CAMEL	3.5816	3.4233	3.7783	3.9676	<b>3.0873</b>	3.2871	3.9448
XALAN	0.0819	0.1243	0.1963	0.0127	3.6511	0.0250	<b>0.011</b>
JEDIT	<b>0.4964</b>	0.5028	0.5146	0.5043	0.5388	0.4994	0.5102
PC	0.0197	<b>0.0186</b>	0.0219	0.0229	0.0220	0.0231	0.0281
IVY	2.7418	2.6782	2.7581	2.8575	<b>2.1136</b>	2.7941	3.2958
LOG4J	0.9246	1.6665	1.8548	0.3678	9.4673	0.1981	<b>0.0732</b>
LUCENE	3.2318	<b>2.5673</b>	4.4516	3.6120	7.1101	2.5724	3.0943
SYANAPSE	3.6026	3.6058	4.2419	3.9603	3.3592	<b>2.9517</b>	3.8717
VELOCITY	3.4565	3.5309	4.2605	3.8468	4.8836	<b>3.2383</b>	4.8097
XERCES	1.1436	1.1924	2.0578	1.2307	6.1507	<b>0.5256</b>	1.7239

## VII. THREATS TO VALIDITY

The various threats to validity that may impact the analysis of the proposed approach, the experimental study and conclusions are presented here.

### A. Internal Validity

NEMC as performance measure has been adopted by previous studies. Performance measures of similar kind which give more weightage to one kind of error is common in defect prediction [30],

TABLE VI  
NORMALIZED PENALTY AT  $\beta = 40$

$\beta = 40$	Bayes1	Bayes2	LR	L-SVM	NB	RF	SVM
POI	<b>4.0646</b>	4.9378	6.5161	5.5262	15.4899	4.0750	4.7109
ANT	5.4371	5.0829	6.2550	5.9547	<b>4.1858</b>	4.8324	6.3339
CAMEL	7.1380	6.7948	7.5424	7.9295	<b>6.1114</b>	6.5313	7.8823
XALAN	0.1553	0.2419	0.3834	0.0143	7.2993	0.0448	<b>0.011</b>
JEDIT	0.9713	0.9835	0.9974	1.0021	<b>0.8706</b>	0.9917	1.0204
PC	0.0394	<b>0.0371</b>	0.0437	0.0457	0.0439	0.0462	0.0561
IVY	5.4465	5.2932	5.4879	5.6929	<b>4.1313</b>	5.5648	6.5895
LOG4J	1.7829	3.2732	3.6534	0.6637	18.9188	0.3327	<b>0.0732</b>
LUCENE	6.3008	<b>4.9606</b>	8.7976	7.0829	14.1711	4.9713	6.0189
SYANAPSE	7.1101	7.1153	8.4172	7.8437	6.6138	<b>5.8047</b>	7.6672
VELOCITY	6.8036	6.9563	8.4389	7.5979	9.6938	<b>6.3639</b>	9.5520
XERCES	2.2292	2.3269	4.0693	2.4136	12.2726	<b>1.0037</b>	3.3701

TABLE VII  
FRIEDMAN NEMENYI  $CD=2.62$

$\beta = 5$	Mean rank
RF	2.083
Bayes2	3.167
Bayes1	3.500
SVM	4.417
L-SVM	4.667
NB	4.833
LR	5.333
$\beta = 10$	Mean rank
RF	2.250
Bayes2	3.167
Bayes1	3.250
SVM	4.417
L-SVM	4.583
NB	4.833
LR	5.500
$\beta = 20$	Mean rank
RF	2.333
Bayes2	2.917
Bayes1	3.083
L-SVM	4.667
SVM	4.667
NB	4.833
LR	5.500
$\beta = 40$	Mean rank
RF	2.583
Bayes2	2.917
Bayes1	3.083
NB	4.333
L-SVM	4.833
SVM	4.833
LR	5.417

TABLE VIII  
WILCOXON SIGNED RANK TEST  
(WSR) AND RANK TEST (RT)

$\beta = 5$	RT	WSR
Bayes2-LR	0.006	0.001
$\beta = 10$	RT	WSR
Bayes2-LR	0.0004	0.0005
$\beta = 20$	RT	WSR
Bayes2-LR	0.0004	0.0005
$\beta = 40$	RT	WSR
Bayes2-LR	0.0003	0.00004
$\beta = 5$	RT	WSR
Bayes1-LR	0.005	0.001
$\beta = 10$	RT	WSR
Bayes1-LR	0.0005	0.0005
$\beta = 20$	RT	WSR
Bayes1-LR	0.0005	0.0005
$\beta = 40$	RT	WSR
Bayes1-LR	0.0004	0.00004



[32]. The choice of the cost factor  $\beta$  was made under the assumption that a defect uncovered in production is more costly than testing for bugs in a defect free class. Different choices of cost factor  $\beta$  could yield different results, it is up to the project team to set a suitable value for  $\beta$  based on available resources and time constraints.

### B. External Validity

The results presented holds good for the 12 Promise projects. The experiment may lead to different results if another dataset is used. Further the metrics used are a combination of several metrics proposed by different authors, these metrics have been adopted in several defect prediction studies. Different set of metrics could lead to different results.

## VIII. CONCLUSION

In this paper, we have formulated the Bayesian logistic regression with three different priors for defect prediction problem. We study the performance of Bayesian approach for logistic regression by making use of several versions of 12 promise projects. The performance of Bayesian logistic Regression with informative priors is significantly better than logistic regression. The performance of Bayesian logistic regression is very much dependent on the choice of prior. Bayesian logistic regression with informative priors outperform their counterpart, Bayesian logistic regression with non-informative prior. The comparative study of all classifiers reveal that Random Forest ranks first and Bayesian logistic regression with Bayes-2 prior, Bayesian logistic regression with Bayes-1 prior ranks second and third as compared with other widely used classifiers.

## REFERENCES

- [1] J. Van Houwelingen and S. Le Cessie, "Predictive value of statistical models," *Statistics in medicine*, vol. 9, no. 11, pp. 1303–1325, 1990.
- [2] J. Scott Long, "Regression models for categorical and limited dependent variables," *Advanced quantitative techniques in the social sciences*, vol. 7, 1997.
- [3] G. King and L. Zeng, "Logistic regression in rare events data," *Political analysis*, vol. 9, no. 2, pp. 137–163, 2001.
- [4] E. M. Schulz, D. Betebner, and M. Ahn, "Hierarchical logistic regression in course placement," *Journal of Educational Measurement*, vol. 41, no. 3, pp. 271–286, 2004.
- [5] A. Genkin, D. D. Lewis, and D. Madigan, "Large-scale bayesian logistic regression for text categorization," *Technometrics*, vol. 49, no. 3, pp. 291–304, 2007.
- [6] A. G. Koru, D. Zhang, K. El Emam, and H. Liu, "An investigation into the functional form of the size-defect relationship for software modules," *IEEE Transactions on Software Engineering*, vol. 35, no. 2, pp. 293–304, 2009.
- [7] R. Subramanyam and M. S. Krishnan, "Empirical analysis of ck metrics for object-oriented design complexity: Implications for software defects," *IEEE Transactions on software engineering*, vol. 29, no. 4, pp. 297–310, 2003.
- [8] 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, no. 10, pp. 751–761, 1996.
- [9] T. Zimmermann, N. Nagappan, and A. Zeller, "Predicting bugs from history," *Software Evolution*, vol. 4, no. 1, pp. 69–88, 2008.
- [10] L. Guo, Y. Ma, B. Cukic, and H. Singh, "Robust prediction of fault-proneness by random forests," in *Software Reliability Engineering, 2004. ISSRE 2004. 15th International Symposium on*, pp. 417–428, IEEE, 2004.
- [11] R. Jindal, R. Malhotra, and A. Jain, "Software defect prediction using neural networks," in *Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions), 2014 3rd International Conference on*, pp. 1–6, IEEE, 2014.
- [12] G. Czubala, Z. Marian, and I. G. Czubala, "Software defect prediction using relational association rule mining," *Information Sciences*, vol. 264, pp. 260–278, 2014.
- [13] R. Rana, M. Staron, C. Berger, J. Hansson, M. Nilsson, and W. Meding, "Analyzing defect inflow distribution and applying bayesian inference method for software defect prediction in large software projects," *Journal of Systems and Software*, vol. 117, pp. 229–244, 2016.
- [14] 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, 2008.
- [15] "The promise repository of empirical software engineering data," 2015.
- [16] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," *IEEE Transactions on software engineering*, vol. 20, no. 6, pp. 476–493, 1994.
- [17] J. Bansiya and C. G. Davis, "A hierarchical model for object-oriented design quality assessment," *IEEE Transactions on software engineering*, vol. 28, no. 1, pp. 4–17, 2002.
- [18] M.-H. Tang, M.-H. Kao, and M.-H. Chen, "An empirical study on object-oriented metrics," in *Software Metrics Symposium, 1999. Proceedings. Sixth International*, pp. 242–249, IEEE, 1999.
- [19] R. Martin, "Oo design quality metrics," *An analysis of dependencies*, vol. 12, pp. 151–170, 1994.
- [20] T. J. McCabe, "A complexity measure," *IEEE Transactions on software Engineering*, no. 4, pp. 308–320, 1976.
- [21] M. Jureczko and L. Madeyski, "Towards identifying software project clusters with regard to defect prediction," in *Proceedings of the 6th International Conference on Predictive Models in Software Engineering*, p. 9, ACM, 2010.
- [22] D. Firth, "Bias reduction of maximum likelihood estimates," *Biometrika*, vol. 80, no. 1, pp. 27–38, 1993.
- [23] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.
- [24] T. M. Khoshgoftaar and N. Seliya, "Comparative assessment of software quality classification techniques: An empirical case study," *Empirical Software Engineering*, vol. 9, no. 3, pp. 229–257, 2004.
- [25] R Core Team, *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013.
- [26] N. G. Polson, J. G. Scott, and J. Windle, "Bayesian inference for logistic models using polya-gamma latent variables." Most recent version: Feb. 2013., 2013.
- [27] G. Heinze and M. Ploner, *logistf: Firth's Bias-Reduced Logistic Regression*, 2016. R package version 1.22.
- [28] D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, and F. Leisch, *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, 2017. R package version 1.6-8.
- [29] J. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for generalized linear models via coordinate descent," *Journal of Statistical Software*, vol. 33, no. 1, pp. 1–22, 2010.
- [30] K. Muthukumar, A. Dasgupta, S. Abhidnya, and L. B. M. Neti, "On the effectiveness of cost sensitive neural networks for software defect prediction," in *International Conference on Soft Computing and Pattern Recognition*, pp. 557–570, Springer, 2016.
- [31] K. Dejaeger, T. Verbraken, and B. Baesens, "Toward comprehensible software fault prediction models using bayesian network classifiers," *IEEE Transactions on Software Engineering*, vol. 39, no. 2, pp. 237–257, 2013.
- [32] K. Muthukumar, A. Choudhary, and N. B. Murthy, "Mining github for novel change metrics to predict buggy files in software systems," in *Computational Intelligence and Networks (CINE), 2015 International Conference on*, pp. 15–20, IEEE, 2015.

# Revisiting the Impact of Regression Models for Predicting the Number of Defects

Man Wu<sup>1,2</sup>, Sizhe Ye<sup>4</sup>, Chunhua Li<sup>1\*</sup>, Ziyi Ma<sup>3</sup>, Zhongwang Fu<sup>2</sup>

<sup>1</sup>Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan, China

<sup>2</sup>School of Computer Science and Information Engineering, Hubei University, Wuhan, China

<sup>3</sup>School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China

<sup>4</sup>State Key Lab. of Software Engineering, Computer School, Wuhan University, Wuhan, China

\*Corresponding author email: li.chunhua@hust.edu.cn

**Abstract**— Predicting the number of faults in software modules can be more helpful instead of predicting the modules being faulty or non-faulty. Chen et al. (SEKE 397-402, 2015) and Rathore et al. (Soft Computing 21: 7417-7434, 2017) empirically investigate the feasibility of some regression algorithms for predicting the number of defects. The experimental results showed that the decision tree regression algorithm performed best in terms of average absolute error (AAE), average relative error (ARE) and root mean square error (RMSE). However, they did not consider the imbalanced data distribution problem in defect datasets and employed improper performance measures for evaluating the regression models to evaluate the performance of models for predicting the number of defects. Hence, we revisit the impact of different regression algorithms for predicting the number of defects using Fault-Percentile-Average (FPA) as the performance measure. The experiments on 31 datasets from PROMISE repository show that the prediction performance of models for predicting the number of defects built by different regression algorithms are various, and the gradient boosting regression algorithm and the Bayesian ridge regression algorithm can achieve better performance.

**Keywords**—predicting the number of defects; regression algorithm; data imbalance; Fault-Percentile-Average;

## I. INTRODUCTION

Software defect prediction is one of the most important software quality assurance techniques. Based on the investigation of historical metrics, defect prediction aims to detect the defect proneness of new software modules. Therefore, defect prediction is often used to help to reasonably allocate limited development and maintenance resources [1]. So far, many efficient software defect prediction methods using statistical methods or machine learning techniques have been proposed [2-4], but they are usually confined to predicting a given software module being faulty or non-faulty by means of some binary classification techniques.

However, predicting the defect-prone of a given software module does not provide enough logistics to software testing in practice [5-6]. Some of the faulty software modules may have comparatively vast quantities of faults compared to other

modules and hence require some additional maintenance resources to fix them. So, it may result in a waste of limited maintenance resources if simply predicting the defect-prone of a given software module and allocating the limited maintenance resources solely based on faulty and non-faulty information. If we are able to predict the accurate number of faults, software testers will pay particular attention to those software modules that have more number of faults, which makes testing processes more efficient in the case of limited development and maintenance resources. Thus, predicting the number of faults in software modules can be more helpful instead of predicting the modules being faulty or non-faulty [6].

### A. Motivation

A number of prior studies have investigated regression models on predicting the number of faults. Some researchers [7-12] have investigated genetic programming, decision tree regression, and multilayer perceptron in the context of predicting the number of defect, and found that these models achieved good performance. Chen et al. [11] performed an empirical study on predicting the number of faults using six regression algorithms and found that the prediction model built with decision tree regression had the highest prediction accuracy in terms of precision and root mean square error (RMSE). In another similar study, Rathore et al. [12] presented an experimental study to evaluate and compare six regression algorithms for predicting the number of defects. The results found that decision tree regression, multilayer perceptron, and linear regression achieved better performance in terms of average absolute error (AAE) and average relative error (ARE), measure of completeness, and prediction at level  $l$  measures.

However, the software defect datasets are imbalanced. In other words, the number of defects in the majority of modules is zero, and the minority of modules have one or more defects. Using imbalanced defect data to derive a regression model and then estimate the error value of the resulting learned model can result in misleading over-optimistic estimates due to over-specialization of the learning algorithm to the imbalanced defect data [6]. Suppose that we have trained a regression model to predict the number of defects in the project Ant 1.3 (see Table I), which contains 124 instances and 33 defects. An AAE value of, say, 0.264 ( $=33/124$ ) may make the regression model seem quite accurate. But, an AAE value of 0.264 may

not be acceptable—the regression model could predict the number of defects of all instances to be zero.

In addition, Yang et al. [13] pointed out that predicting the precise number of defects of a module is hard to do due to the lack of good quality data in practice. Actually, for those existing approaches that tried to predict explicitly the number of defects in a software module, they used these predicted numbers to rank the modules anyway, to direct the software quality assurance team in targeting the most faulty modules first [14], [15]. Actually, for software defect prediction for the ranking task, models with higher prediction accuracy (smaller AAE or ARE) might give a worse ranking. For example, assuming that there are three software modules  $M_1$ ,  $M_2$ , and  $M_3$ , which have 2, 3, and 4 defects respectively, model A predicts that  $M_1$ ,  $M_2$ , and  $M_3$  have 2, 1, and 4 defects respectively, while model B predicts that  $M_1$ ,  $M_2$ , and  $M_3$  have 0, 1, and 2 defects respectively. Although model A has a better prediction accuracy (according to AAE and ARE), model B gives tester better ranking of these three software modules and can guide the assignment of testing resources correctly. Hence, Weyuker et al. [15] proposed FPA to reflect the effectiveness of the different prediction models for predicting the number of defects.

### B. Our work

Considering on the issue that the existing studies [11-12] employed improper performance measures (e.g., RMSE, AAE and ARE) to evaluate the performance of models for predicting the number of defects, thus resulting in wrong conclusion, we revisit the impact of seven regression algorithms for predicting the number of defects by using FPA as the performance measure. These seven regression algorithms are Bayesian Ridge Regression (BRR), Decision Tree Regression (DTR), Gradient Boosting Regression (GBR), Linear Regression (LR), Nearest Neighbors Regression (NNR), Multilayer Perceptron Regression (MPR), and Support Vector Regression (SVR). The experimental study is performed on 9 software projects collected from the PROMISE repository. The FPA performances of the seven algorithms are analyzed by the one-way ANOVA test and the multiple comparison test. The experimental results show that the performance difference of seven regression algorithms for predicting the number of defects are statistically significant, and the model for predicting the number of defects built by Gradient Boosting Regression algorithm and Bayesian Ridge Regression algorithm achieve better performance.

### C. Organization

The remainder of this paper is organized as follows. Section 2 presents the related work. Section 3 describes the seven regression algorithms for predicting the number of defects. Section 4 introduces the experiment setup. Section 5 demonstrates the experimental results. Finally, Section 6 addresses the conclusion.

## II. RELATED WORK

Many researchers have proposed various models for predicting the module being faulty or non-faulty. Support vector machine [16-17], neural networks [18], decision trees

[19] and Bayesian methods [20] paved the way for classification-based methods in the field of defect prediction. These methods used software metrics to properly predict whether a module is defect-prone or not. However, these methods are confined with predicting a given software module being faulty or non-faulty.

A number of prior studies have investigated regression models on predicting the number of software faults. Graves et al. [21] presented a generalized linear regression based method for predicting the number of defects by using various change metrics datasets collected from a large telecommunication system, and found that modules age, changes made to module and the age of the changes were significantly correlated with the defect-prone. Afzal et al. [8] used genetic programming for modeling software reliability growth to help in deciding project release time and managing project resources. Rathore et al. [7] proposed an approach to predict the number of faults in the software system and developed defect prediction model using neural network and genetic programming. In the subsequent study [9], they proposed an approach to predict the number of faults in the given software system using the Genetic Programming (GP). The results showed that GP based models could produce the significant results for the number of faults prediction. In another paper [10], they explored the capability of decision tree regression (DTR) for the number of faults prediction in two different scenarios, intra-release prediction and inter-releases prediction for the given software system. The experimental results indicated that intra-project prediction produced better accuracy.

Chen et al. [11] performed an empirical study on predicting the number of faults using six regression algorithms and found that the prediction model built with decision tree regression had the highest prediction accuracy in terms of precision and RMSE in most cases. In another similar study, Rathore et al. [12] presented an experimental study to evaluate and compare six regression algorithms for the number of faults prediction. The results found that decision tree regression, genetic programming, multilayer perceptron, and linear regression achieved better performance in terms of AAE, ARE, measure of completeness, and prediction at level  $l$  measures in many cases. However, the two empirical studies employed improper performance measures (e.g., RMSE, AAE and ARE) to evaluate the performance of models for predicting the number of defects, thus resulting in wrong conclusion.

## III. REGRESSION ALGORITHMS

Figure 1 shows a typical process of predicting the number of defects, which can be divided into four stages. In the first stage, different from labeling a module defective or not for predicting a given software module being faulty or non-faulty, the number of defects in software modules is extracted. In the second stage, the features for each software module are extracted. Common features include complexity metrics, changes, or structural dependencies. Therefore, we can construct a training dataset for predicting the number of defects. In the third stage, we can build a model for predicting the number of defects with the help of some regression algorithms. In the last stage, after extracting the same features from a new

software module, we can use the model to predict the number of defects in the module.

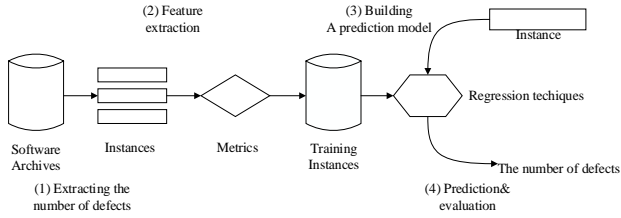


Fig. 1. The flow chart of predicting the number of defects

In this paper, we select seven regression algorithms, i.e., Bayesian Ridge Regression (BRR), Decision Tree Regression (DTR), Gradient Boosting Regression (GBR), Linear Regression (LR), Nearest Neighbors Regression (NNR), Multilayer Perceptron Regression (MPR), and Support Vector Regression (SVR). We implement these regression algorithms based on the python machine learning library *sklearn*. Unless otherwise specified, the default parameter settings for different regression models used in our experiments are specified by *sklearn*. That is, we do not perform additional optimization for each regression model. The brief introductions of the seven regression algorithms are described as follows:

(1) Bayesian Ridge Regression (BRR). It estimates a probabilistic model of the regression problem by introducing uninformative priors over the hyper parameters of the model. The prior for the parameter  $\omega$  is given by a spherical Gaussian:

$$p(\omega|\lambda) = N(\omega|0, \lambda^{-1}I_p) \quad (1)$$

The priors over  $\alpha$  and  $\lambda$  are chosen to be gamma distributions, the conjugate prior for the precision of the Gaussian. The resulting model is called Bayesian Ridge Regression, and is similar to the classical Ridge. The parameters  $\omega$ ,  $\alpha$  and  $\lambda$  are estimated jointly during the fit of the model. The remaining hyperparameters are the parameters of the gamma priors over  $\alpha$  and  $\lambda$ . These are usually chosen to be non-informative. The parameters are estimated by maximizing the marginal log likelihood.

(2) Decision Tree Regression (DTR). It predicts the value of a target variable by learning simple decision trees inferred from the data features. A decision tree is built top-down from a root node and uses a splitting criterion to partition the data into subsets that contain instances with similar values. The attribute which maximizes the expected error reduction is chosen as the root node. The process is run recursively on the non-leaf branches, until all data is processed.

(3) Gradient Boosting Regression (GBR). It produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do. It allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function.

(4) Linear Regression (LR). It is a statistical approach for modeling the linear relationship between a dependent variable  $y$  and one or more independent variables. A linear regression model can be described according to Eq.(2).

$$Y=b_0+b_1 x_1+b_2 x_2+\dots+b_n x_n \quad (2)$$

where  $Y$  is the dependent variable,  $x_1, x_2, \dots, x_n$  are the independent variables,  $b_1, b_2, \dots, b_n$  are the regression coefficients of the independent variables and  $b_0$  is the error term.

(5) Nearest Neighbors Regression (NNR). It is based on the  $k$ -nearest neighbors algorithm, and the regression value of an instance is computed based the mean of the labels of its nearest neighbors. The basic nearest neighbors regression uses uniform weights: that is, each point in the local neighborhood contributes uniformly to the classification of a query point. Under some circumstances, it can be advantageous to weight points such that nearby points contribute more to the regression than faraway points.

(6) Multilayer Perceptron Regression (MPR). It consists of a series of processing elements interconnected through the connection weights in the form of layers. A multilayer perceptron regression model can be described according to Eq. (3) and Eq. (4).

$$net_k = w_{1k}x_1 + w_{2k}x_2 + \dots + w_{nk}x_n + b_k \quad (3)$$

$$O_k = f(net_k) \quad (4)$$

where  $O_k$  is the dependent variable,  $x_1, x_2, \dots, x_n$  are the independent variables,  $w_{1k}, w_{2k}, \dots, w_{nk}$  are the weights associated with each input layer, and function  $f(\cdot)$  is a activation function.

(7) Support Vector Regression (SVR). It uses the same principles as the SVM for classification, with only a few minor differences. The main idea is always the same: to minimize error, individualizing the hyperplane which maximizes the margin, keeping in mind that part of the error is tolerated.

Training the original SVR means solving:

$$\text{minimize } \frac{1}{2} \|w\|^2 \quad (5)$$

$$\text{subject to } \begin{cases} y_i - \langle w, x_i \rangle - b \leq \varepsilon \\ \langle w, x_i \rangle + b - y_i \leq \varepsilon \end{cases} \quad (6)$$

where  $x_i$  is a training sample with target value  $y_i$ . The inner product plus intercept  $\langle w, x_i \rangle + b$  is the prediction for that sample, and  $\varepsilon$  is a free parameter that serves as a threshold: all predictions have to be within an  $\varepsilon$  range of the true predictions. Slack variables are usually added into the above to allow for errors and to allow approximation in the case the above problem is infeasible.

## IV. EXPERIMENT SETUP

### A. Data set

In this experiment, we employ 31 available and commonly used datasets which can be obtained from PROMISE [22]. Table I tabulates the details about the datasets. The 31 datasets have the same 20 features. For the complete details of the software features, please refer to [14].

### B. Performance measures

Considering  $k$  modules listed in increasing order of predicted defect number as  $f_1, f_2, f_3, \dots, f_k$ , and assuming that  $n_i$  is the actual defect number in the module  $i$ ,  $n=n_1+n_2+\dots+n_k$  is

the total number of defects, and the top predicted modules should have  $\sum_{i=k-m+1}^k n_i$  defects. The proportion of the actual defects in the top  $m$  predicted modules to the whole defects is

$$\frac{1}{n} \sum_{i=k-m+1}^k n_i. \quad (7)$$

Then the FPA is define as

$$\frac{1}{k} \sum_{m=1}^k \frac{1}{n} \sum_{i=k-m+1}^k n_i. \quad (8)$$

FPA is actually the average of the proportions of actual defects in the top modules to the whole defects. A higher FPA means a better ranking, where the modules with most defects come first.

TABLE I. DETAILS OF EXPERIMENT DATASET

Project	Release	#Instance	#Defects	%Defects	Max	Avg
Ant	1.3	125	33	16	3	1.65
	1.4	178	47	22.5	3	1.18
	1.5	293	35	10.9	2	1.09
	1.6	351	184	26.2	10	2.00
	1.7	745	338	22.3	10	2.04
Camel	1.0	339	14	3.8	2	1.08
	1.2	608	522	35.5	28	2.42
	1.4	872	335	16.6	17	2.31
	1.6	965	500	19.5	28	2.66
Forest	0.7	29	15	17.2	8	3.0
	0.8	32	6	6.2	4	3.0
Jedit	3.2	272	382	33.1	45	4.24
	4.0	306	226	24.5	23	3.01
	4.1	312	217	25.3	17	2.75
	4.2	367	106	13.1	10	2.21
	4.3	492	12	2.2	2	1.09
Log4j	1.0	135	61	25.2	9	1.79
	1.1	109	86	33.9	9	2.32
Prop	1	18471	5293	14.8	37	2.01
	2	23014	4096	10.6	27	1.68
	3	10274	1640	11.5	11	1.39
	4	8718	1362	9.6	22	1.62
	5	8516	1930	15.3	19	1.49
	6	660	79	10	4	1.2
Synaps	1.0	157	21	10.2	4	1.31
	1.1	222	99	27	7	1.65
	1.2	256	145	33.6	9	1.69
Xalan	2.4	723	156	15.2	7	1.42
	2.5	803	531	48.2	9	1.37
Xerces	1.3	453	193	15.2	30	2.8
	1.4	588	1596	74.3	62	3.65

### C. Experimental procedure

Tantithamthavorn et al. [23] investigated some model validation techniques in the domain of defect prediction and recommended out-of-sample bootstrap validation. Therefore, we employ out-of-sample bootstrap validation to perform the experiment. The out-of-sample bootstrap process is made up of two steps:

(Step 1) A bootstrap sample of size  $N$  is randomly drawn with replacement from an original dataset that is also of size  $N$ .

(Step 2) A model is trained using the bootstrap sample and tested using the rows of the original sample that do not appear in the bootstrap sample.

We repeat the out-of-sample bootstrap 10 times and the average out-of-sample performance is reported as the performance estimate.

### D. Research questions

In order to provide the guidance for the choice of various regression algorithms for predicting the number of defects, we address the following two research questions.

RQ1: Which regression algorithm can achieve the best performance for predicting the number of defects among the seven algorithms?

RQ2: Are the performances of models for predicting the number of defects built with different regression algorithms different?

## V. EXPERIMENT RESULTS

### A. Results for RQ1

The results (in terms of FPA) of the seven regression algorithms for each dataset are reported in Table II. Experimental results show that BRR and GBR produce the higher FPA values on most of the datasets compared to other considered regression algorithms for predicting the number of defects. GBR outperforms other regression algorithms on 9 datasets and achieves the highest average FPA value (0.656). LR, NNR and DTR are the third, fourth and fifth best defect prediction algorithms in terms of the average FPA value, respectively, while MPR and SVR produce relatively lower FPA values compared to other regression algorithms.

TABLE II. FPA VALUES ON 31 DATASETS USING THE SEVEN REGRESSION ALGORITHMS

Release	BRR	DTR	GBR	NNR	LR	MPR	SVR
Ant-1.3	0.708	0.698	0.675	0.639	0.618	<b>0.733</b>	0.680
Ant-1.4	0.551	0.610	<b>0.673</b>	0.590	0.601	0.479	0.551
Ant-1.5	<b>0.742</b>	0.543	0.580	0.656	0.684	0.366	0.440
Ant-1.6	0.721	0.695	<b>0.768</b>	0.721	0.715	0.207	0.449
Ant-1.7	<b>0.811</b>	0.711	0.798	0.771	0.809	0.385	0.475
Camel-1.0	<b>0.608</b>	0.603	0.600	0.597	<b>0.608</b>	0.351	<b>0.608</b>
Camel-1.2	<b>0.645</b>	0.571	0.602	0.574	0.642	0.533	0.570
Camel-1.4	0.691	0.620	0.714	0.652	<b>0.736</b>	0.615	0.453
Camel-1.6	0.704	0.635	0.693	0.486	<b>0.728</b>	0.660	0.513
Forrest-0.7	0.801	0.468	0.718	<b>0.827</b>	0.801	0.641	0.494
Forrest-0.8	0.700	0.800	<b>0.900</b>	0.700	0.700	0.500	0.800
Jedit-3.2	0.846	0.562	<b>0.853</b>	0.789	0.835	0.383	0.503
Jedit-4.0	0.787	0.710	<b>0.803</b>	0.750	0.795	0.723	0.608
Jedit-4.1	<b>0.738</b>	0.619	0.689	0.628	<b>0.738</b>	0.469	0.463
Jedit-4.2	0.787	0.690	0.787	0.743	<b>0.795</b>	0.633	0.716
Jedit-4.3	0.286	0.267	0.267	0.286	0.286	<b>0.955</b>	0.286
Log4j-1.0	0.456	0.715	0.703	0.784	0.457	<b>0.891</b>	0.695
Log4j-1.1	<b>0.735</b>	0.659	0.640	0.642	0.472	0.722	0.547
Prop-1.0	0.588	0.539	<b>0.595</b>	0.555	0.584	0.498	0.392
Prop-2.0	0.542	0.506	0.533	0.497	<b>0.543</b>	0.461	0.447
Prop-3.0	0.488	0.451	0.472	0.466	0.486	<b>0.539</b>	0.426
Prop-4.0	0.631	0.566	0.618	0.597	<b>0.632</b>	0.609	0.508
Prop-5.0	0.552	0.444	0.540	0.508	0.550	<b>0.561</b>	0.430
Prop-6.0	0.588	0.479	0.547	0.567	0.584	<b>0.607</b>	0.501
Synaps-1.0	0.282	0.517	<b>0.562</b>	0.311	0.266	0.271	0.282
Synaps-1.1	<b>0.755</b>	0.753	0.702	0.538	0.740	0.750	0.589
Synaps-1.2	0.648	0.535	0.596	<b>0.650</b>	0.623	0.428	0.567
Xalan-2.4	0.545	0.543	0.541	<b>0.595</b>	0.584	0.537	0.533
Xalan-2.5	0.631	0.597	<b>0.658</b>	0.615	0.616	0.442	0.556
Xerces1-3	0.769	0.740	0.754	0.762	<b>0.804</b>	0.230	0.442
Xerces-1.4	0.724	0.655	<b>0.752</b>	0.680	0.720	0.689	0.632
Average	0.647	0.597	<b>0.656</b>	0.619	0.637	0.544	0.521

Figure 2 shows the box-plot diagrams of FPA measure. X-axis indicates the regression algorithms under consideration, and Y-axis indicates the values of FPA measure produced by the used regression algorithms. The different parts of box-plot show the minimum, maximum, median, first quartile, and third quartile of the samples. The line in the middle of the box shows the median of the samples. SVR and MPR produce the lowest median value, DTR, LR and NNR produce moderate median FPA values, whereas BRR and GBR produce the highest median FPA values. MPR produces the lowest minimum FPA values, DTR, LR, BRR and SVR produce moderate minimum FPA values, GBR and NNR produce the highest minimum FPA values. For maximum value, SVR produces the lowest FPA values, DTR, NNR, LR and MPR produce moderate FPA values, and BRR and GBR produce the highest FPA values. For first quartile value, SVR produces the lowest value, DTR, NNR, LR and MPR produce moderate values, and BRR and GBR produce the highest values. For the third quartile, MPR and SVR produce the lowest value, DTR and NNR produce moderate values, and BRR, GBR and LR produce the highest values.

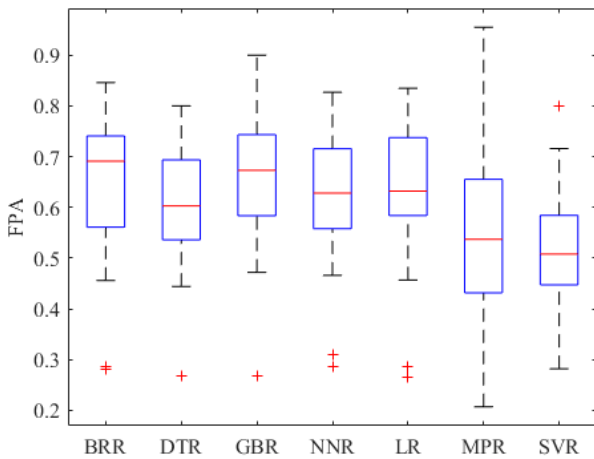


Fig. 2. Box-plot analysis for FPA measure for all the datasets

In total, Table II and Figure 2 illustrate that BRR and GBR outperform other considered regression algorithms. DTR, NNR and LR perform moderately, while MPR and SVR perform relatively poor in compared to other used regression algorithms. To answer the first question, Bayesian Ridge Regression and Gradient Boosting Regression algorithms can achieve the best performance for predicting the number of defects among the seven algorithms.

### B. Results for RQ2

To answer the second question, we carry out one-way ANOVA test on the seven regression algorithms to examine if the algorithms are statistically different or not. Analysis of variance (ANOVA) is a collection of statistical models and their associated procedures used to analyze the differences among group means. Since the two-group case can be covered by a t-test, the main idea of one-way ANOVA is to compare means of two or more groups by using the F distribution. It is more conservative than the t-test and performs well in terms of

comparing groups for statistical significance. The null hypothesis for the one-way ANOVA test is that all the group population means are the same, while the alternate hypothesis is that at least one pair of means is different. Table III shows the one-way ANOVA results.

The first row of the Table III lists five parameters, i.e., sums of squares, degrees of freedom, mean square, F and p-value. Since  $F=4.59 > 3.68$ , the results are significant at the 5% significance level. In addition, it is clearly that the p-value (0.0002) for this test is less than the typical cutoff 0.05. We would reject the null hypothesis, concluding that there is strong evidence that at least two regression algorithms are significantly different from each other.

TABLE III. ANOVA FOR THE DATASETS

Source	Sum Sq.	d.f.	Mean Sq.	F	p-value
Approach	0.507	6	0.0845	4.59	0.0002
Error	3.865	210	0.0184		
Total	4.37	216			

In the study, we further performed the multiple comparison test using Tukey's honestly significant difference criterion. Figure 3 shows the multiple comparison result for the seven regression algorithms. The figure displays graphs with each group mean represented by a symbol ( $\circ$ ) and 95% confidence interval as a line around the symbol. There are two situations in the figure: two means are significantly different, if their intervals disjoint. On the contrary, two means are not significantly different, if their intervals overlap. From Figure 3, we can summarize the following points:

- (1) MPR and SVR algorithms have significantly worse prediction performance than other algorithms.
- (2) Although the other five algorithms show similar performance (no significant difference), BRR and GBR performs slightly better than DTR, NNR and LR.

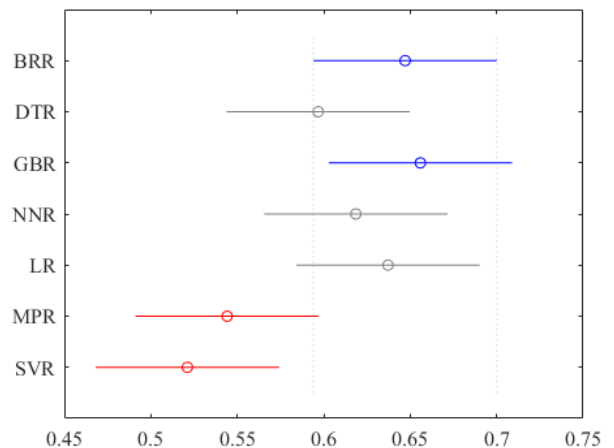


Fig. 3. Multiple comparison for seven algorithms

As seen in Table III and Figure 3, we can conclude that the models built by different regression algorithms for predicting the number of defects have different performance.

### C. Threats to Validity

In this subsection, we discuss several validity threats that may have an impact on the results of our studies. (1) Although the 31 datasets in our experiment have been widely used in many software defect prediction studies, we still cannot claim that our conclusion can be generalized to other datasets. (2) We only study the seven regression algorithms without additional optimization for a given dataset. (3) We only employ FPA as the evaluation measure. Nonetheless, other evaluation measures such as cost effectiveness graph [24] can also be considered.

### VI. CONCLUSION

The existing studies [11-12] employed improper performance measures (e.g., RMSE, AAE and ARE) to evaluate the performance of models for predicting the number of defects. Therefore, in this paper, we evaluated and compared the performance of seven regression algorithms (i.e., BRR, DTR, GBR, LR, NNR, MPR, and SVR) for predicting the number of defects in given software modules by using FPA as the performance measure. The experiments were performed on 31 datasets from the PROMISE repository. In addition, the one-way ANOVA test and the multiple comparison test are performed to assess the relative performance of the seven algorithms. The experimental results show that the performance difference of seven regression algorithms for predicting the number of defects are statistically significant, and Gradient Boosting Regression algorithm and Bayesian Ridge Regression algorithm achieve better performance for predicting the number of defects.

### ACKNOWLEDGMENT

This work is supported by the National Key R&D Program of China (2016YFB0800402), partially supported by the National Natural Science Foundation of China under Grant No.61232004 and the Fundamental Research Funds for the Central Universities(2016YXMS020).

### REFERENCES

- [1] F. Rahman, D. Posnett, P. Devanbu, Recalling the imprecision of cross-project defect prediction, Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering, 2012, 61.
- [2] X. Yu, M. Wu, Y. Jian, et al, Cross-company defect prediction via semi-supervised clustering-based data filtering and MSTRa-based transfer learning, Soft Computing, 2018: 1-12.
- [3] X. Yu, J. Liu, W. Peng, et al, Improving Cross-Company Defect Prediction with Data Filtering, International Journal of Software Engineering and Knowledge Engineering, 2017, 27(09n10): 1427-1438.
- [4] X. Yu, J. Liu, M. Fu, et al, A Multi-Source TrAdaBoost Approach for Cross-Company Defect Prediction, SEKE, 2016: 237-242.
- [5] X. Yu, J. Liu, Z. Yang, et al, The Bayesian Network based program dependence graph and its application to fault localization, Journal of Systems and Software, 2017, 134: 44-53.
- [6] Yu X, Liu J, Yang Z, et al. Learning from Imbalanced Data for Predicting the Number of Software Defects, 2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE), IEEE, 2017: 78-89.
- [7] Rathore S S, Kuamr S, Comparative analysis of neural network and genetic programming for number of software faults prediction, 2015 National Conference on Recent Advances in Electronics & Computer Engineering (RAECE), 2015: 328-332.
- [8] W. Afzal, R. Torkar, R. Feldt, Prediction of fault count data using genetic programming, Multitopic Conference, 2008. INMIC 2008. IEEE International. IEEE, 2008: 349-356.
- [9] S. S. Rathore, S. Kumar, Predicting number of faults in software system using genetic programming, Procedia Computer Science, 2015, 62: 303-311.
- [10] S. S. Rathore, S. Kumar, A Decision Tree Regression based Approach for the Number of Software Faults Prediction, ACM SIGSOFT Software Engineering Notes, 2016, 41(1): 1-6.
- [11] M. Chen, Y. Ma, An empirical study on predicting defect numbers, 28th International Conference on Software Engineering and Knowledge Engineering, 2015: 397-402.
- [12] S. S. Rathore, S. Kumar, An empirical study of some software fault prediction techniques for the number of faults prediction, Soft Computing, 2016: 1-18.
- [13] Yang, Xiaoxing, K. Tang, and X. Yao, A learning-to-rank approach to software defect prediction, IEEE Transactions on Reliability, 2015, 64(1): 234-246.
- [14] K. Gao and T. M. Khoshgoftaar, A comprehensive empirical study of count models for software fault prediction, IEEE Transactions on Reliability, 2007, 56(2): 223-236.
- [15] E. J. Weyuker, T. J. Ostrand, and R. M. Bell, Comparing the effectiveness of several modeling methods for fault prediction, Empirical Software Engineering, 2010, 15(3): 277-295.
- [16] D. Gray, D. Bowes, N. Davey, et al, Using the support vector machine as a classification method for software defect prediction with static code metrics, International Conference on Engineering Applications of Neural Networks, Springer Berlin Heidelberg, 2009: 223-234.
- [17] Z. Yan, X. Chen, P. Guo, Software defect prediction using fuzzy support vector regression, International Symposium on Neural Networks, Springer Berlin Heidelberg, 2010: 17-24.
- [18] M. M. T. Thwin, T. S. Quah, Application of neural networks for software quality prediction using object-oriented metrics, Journal of systems and software, 2005, 76(2): 147-156.
- [19] J. Wang, B. Shen, Y. Chen, Compressed C4. 5 models for software defect prediction, 2012 12th International Conference on Quality Software. IEEE, 2012: 13-16.
- [20] T. Wang, W. Li, Naive bayes software defect prediction model, Computational Intelligence and Software Engineering (CiSE), 2010 International Conference on IEEE, 2010: 1-4.
- [21] T. L. Graves, A. F. Karr, J. S. Marron, et al, Predicting fault incidence using software change history, IEEE Transactions on software engineering, 2000, 26(7): 653-661.
- [22] G. Boetticher, T. Menzies, T. Ostrand, The PROMISE Repository of Empirical Software Engineering Data, 2007 <<http://promisedata.org/repository>>.
- [23] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, et al, An Empirical Comparison of Model Validation Techniques for Defect Prediction Models, IEEE Transactions on Software Engineering, 2017, 43(1): 1-18.
- [24] T. Jiang, L. Tan, S. Kim, Personalized defect prediction, Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on. IEEE, 2013.

# A Gated Hierarchical LSTMs for Target-based Sentiment Analysis

Xiaofang Zhang<sup>†\*</sup>, Bin Liang<sup>\*</sup>, Qian Zhou<sup>\*</sup>, Hao Wang<sup>\*</sup>, Baowen Xu<sup>†</sup>

<sup>†</sup>State Key Laboratory for Novel Software Technology, Nanjing University Nanjing, China

<sup>\*</sup>School of Computer Science and Technology, Soochow University Suzhou, China

Email: xfzhang@suda.edu.cn

**Abstract**—The deep neural model combining attention mechanism has achieved remarkable success in the task of target-based sentiment analysis. In current research, the attention mechanism is more broadly combined with LSTM(Long Short-Term Memory) networks, however, such neural network-based architectures generally rely on complex computation and only focus on the single target. We propose a gated hierarchical LSTMs(GH-LSTM) model of combining regional LSTM and sentence-level LSTM via a gated operation for the task of target-based sentiment analysis. This approach can distinguish different polarities of sentiment of different targets in the same sentence through a regional LSTM, and is able to concentrate on the long dependency of target in the whole sentence via a sentence-level LSTM. The experimental results on multi-domain datasets of two languages from SemEval2016 indicate that, our approach yields better performance than SVM(Support Vector Machine) and several typical neural network models.

## I. INTRODUCTION

As a more fundamental task in the field of sentiment analysis, target-based sentiment analysis is capable of digging for subtler descriptions of polarity in terms of different targets in the sentence. It has become one of the hot areas of NLP (Natural Language Processing) during the recent years [1, 2]. Different from the conventional sentiment analysis, target-based sentiment analysis needs to identify the sentiment polarities of different targets in the sentence, which depends on the sentiment information of different targets in the text rather than the context [3]. For example, in sentence “Good food but dreadful service at that restaurant”, the sentiment polarity of target “food” is positive while the sentiment polarity of target “service” is negative. This means that even in the same sentence different targets may have completely opposite sentiment polarities.

In the past research, most of the methods based on traditional machine learning had to rely on complicated artificial rules and feature engineering in spite of their remarkable success in tasks of traditional sentiment analysis [4]. Additionally, it is difficult for this kind of methods to do feature extraction and learning for different target words efficiently, so it tended to predict that different targets from the same sentence have the same sentiment polarity.

In recent years, deep learning methods have achieved great progress in many fields. More and more researchers started to

apply deep learning methods in NLP tasks [5, 6]. Deep neural network models combined with LSTM get better results than the traditional machine learning methods in NLP tasks with target attached such as target-based relation classification [7], target-based modeling of sentence pairs [8], target-based machine translation [9] and target-based sentiment analysis [10]. This kind of deep network models combined with attention mechanism can highly focus on specific feature information of target and can adjust the parameters of the network for different targets to dig for more hidden feature information. In the task of attention-based sentiment analysis, the network model combined with LSTM can effectively save the dependency relationships between different words by receiving the input of the text sequentially. Meanwhile, through using different target attention matrixes as the input of the network, we could enable the network to pay high attention to the sentiment information of different targets during the training process.

However, LSTM network requires a sequential input of the text. The computation of every word needs to be connected with the information of previous words and the attention matrix and the input word vector need the support from weight matrix, which leads to the complexity of the model’s parameter computation [11]. On the other hand, when a sentence contains more targets with complicated dependency relationships, it is hard for the models combined with only the specific attention mechanism for target to identify the sentimental polarities of different targets.

In order to address these problems, in this paper, a gated hierarchical LSTMs model, namely GH-LSTM, for target-based sentiment analysis is proposed. This model could be able to extract both targets(pre-defined by the dataset) and sentence information in the given text. The sole gated hierarchical LSTMs aims to discriminate different polarities of different targets in the same sentence and capture the feature information of the whole sentence. The regional LSTMs receive a regional sequential input including a target, aiming to concentrate on the specific target in the sentence we are considering. The sentence-level LSTMs receive the sequential input of the whole sentence, which can explicitly reveal the important relations between the specific target and the whole sentence. After that regional LSTMs will receive the feature extracted from sentence-level LSTMs modulated by a gated operation.



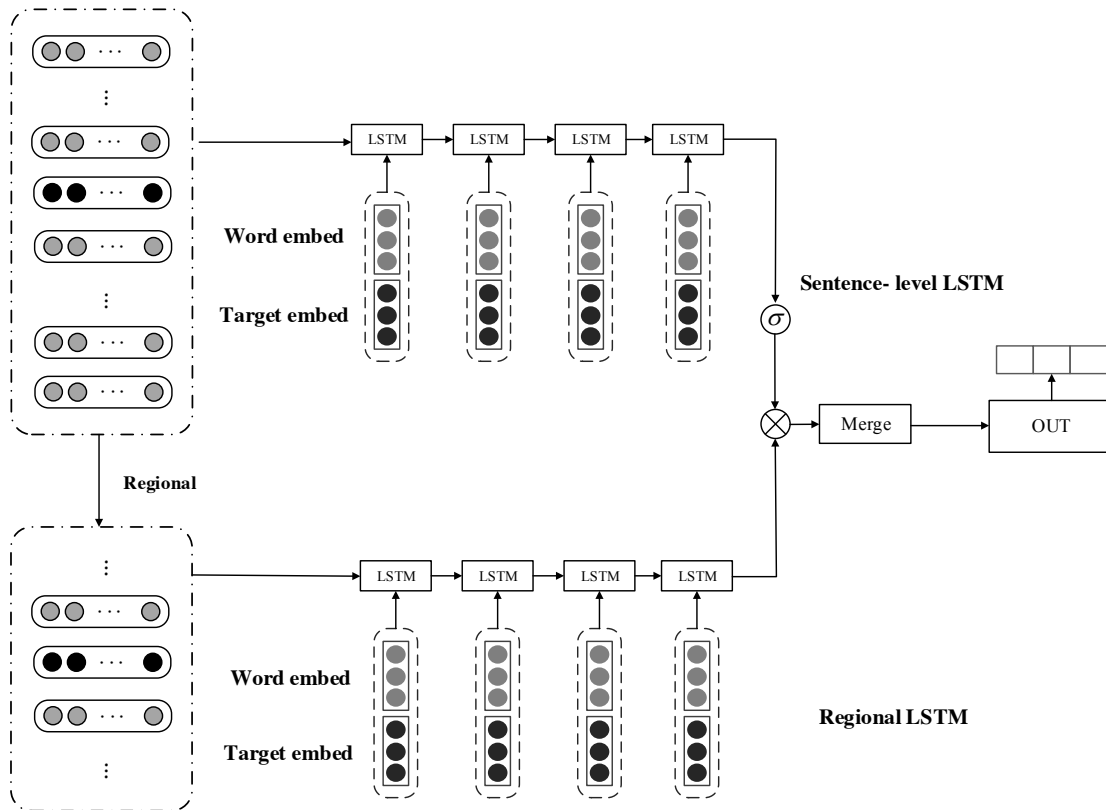


Fig. 1. The illustration of our gated hierarchical LSTMs (GH-LSTM). Regional LSTM receive a region of the sentence divided based on specific target. Sentence-level LSTM receive the whole sentence sequential input. The output of sentence-level LSTM that modulated by a gated operation and the final output of regional LSTM are fed into a merge layer to output a probability distribution of sentiment.

## II. RELATED WORK

In this section, we will introduce some related works including target-based sentiment analysis and LSTMs.

### A. Target-based sentiment analysis

Target-based sentiment analysis is sentiment analysis on a deeper level and a fine-grained text classification task, which has achieved much attention from researchers since its emergence [12]. Qiu et al. [13] propose a bidirectional back propagation algorithm to determine the sentiment polarity of target by the specific relationship between the sentiment word and the target word. Through the extension of this dictionary, the approach uses the updated dictionary and extracts the relationship between the target words to predict the dependency between the target and the sentiment words. Finally, the sentiment polarity of the target can be obtained. Kiritchenko et al. [14] use a SVM classifier which is combined with multiple features for aspect-level sentiment analysis. This method adds unigram, bigram, sentiment dictionary and other features to the libSVM model to exploit multiple types of sentiment information in the text, making the classifier capable of identifying sentiment polarities in different aspects.

As for the deep learning-based methods, Nguyen and Shirai [15] proposed a target-based sentiment analysis model

based on RNN(Recursive Neural Network) and dependency tree. This model works on a binary phrase dependency tree containing the element structure and dependency relationship tree of sentence, increasing the correct rate of target-based sentiment analysis considerably and reducing a large amount of feature projects during the task. Dong et al. [16] used an AdaRNN(Adaptive Recursive Neural Network) model to handle target-based sentiment analysis. This model makes use of an adjustable neural network model to learn the connection between the target and the words as well as the syntactic structure of the sentence. Then the model extends the sentiment information by the relation between the target and the other words to identify the sentiment polarity of the target effectively.

### B. Hierarchical model

In recent years, hierarchical models of neural networks have obtained much attention in the field of NLP. Lin et al. [17] propose a hierarchical recurrent neural network language model (HRNNLM) for document modeling by capturing relations between sentences. Li et al. [18] use a hierarchical LSTM auto-encoder to preserve and reconstruct multi-sentence paragraphs and a hierarchical LSTM is used for learning representations of text spans based on attention mechanism [19].

For the task of aspect-based sentiment classification, Ruder et al. [20] introduce a hierarchical bidirectional LSTM (HP-LSTM), which is able to leverage both intra-sentence and inter-sentence relations. This modal inspires us to use a gated hierarchical LSTMs model to leverage both the target and the sentence-level sentiment information.

### III. GATED HIERARCHICAL LSTMS MODEL

As is shown in Figure 1, our GH-LSTM model contains following components:

- **Regional LSTM:** receive a regional sequential input of sentence divided based on specific target, each LSTM unit receives a target embedding amalgamated with word embedding to focus on the specific target information in the process of training.
- **Sentence-level LSTM:** receive sequential input of sentence including word and target embeddings to extract long dependency of the specific target in the whole sentence.
- **Gated merge layer:** combine the output of regional LSTM and sentence-level LSTM through a gated operation.
- **Fully connected layer:** get the sentiment distribution of the specific target in the given sentence.

The targets in the sentence are pre-defined by the dataset. In this section, we will introduce the above components in detail.

#### A. Task Definition and Aspect Representation

Given a sentence  $s = \{w_1, w_2, \dots, t_i, \dots, t_j, \dots, w_n\}$ , where  $t_i$  and  $t_j$  are two different targets in the sentence, the task of target-based sentiment analysis is to discriminate the sentiment polarities of different targets in the same sentence. For example, the sentence “Good food but dreadful service at that restaurant”, the sentiment polarity of the target “food” is positive, while for the target “service” it is negative, so even in the same sentence, the opposite polarities can still appear since they belong to different targets. For each word in the sentence, we generate a  $m$ -dimensional embedding  $\mathbf{x} \in \mathbb{R}^m$  to represent the word and target. Let  $\mathbf{x}_i \in \mathbb{R}^m$  be the  $m$ -dimensional word vector corresponding to the  $i$ -th word, a sentence of length  $n$  concatenated with an aspect embedding can be represented as:

$$\mathbf{E}_s = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \dots \oplus \mathbf{x}_n \quad (1)$$

where  $\oplus$  is the concatenation operation.

#### B. Region division

Regional LSTM is an improvement of LSTM for handing the tasks with a long text input. We divide each sentence into one or more regions based on targets.<sup>1</sup> The regional LSTM in our work is primarily inspired by a regional CNN(Convolutional Neural Network)-LSTM model for long

<sup>1</sup>Unlike using an individual sentence as a region, dividing sentences based on the targets can focus on the most important words of different targets and distinguish the sentence with multiple targets effectively in prediction process.

text sentiment analysis [21]. We divide sentences into specific regions based on specific target, which not only preserves the important feature of specific target in the sentence, but also distinguishes the sentiment information of different targets in sentences availably. Given a sentence  $s = \{w_1, w_2, \dots, t_i, \dots, t_j, \dots, w_n\}$ , we focus on the targets and the words around the target in the sentence, and divide the sentence into different regions based on different targets. For example, in sentence  $s = \{w_1, w_2, \dots, t_i, \dots, t_j, \dots, w_n\}$ , where  $t_i$  and  $t_j$  are two different targets in the sentence, we divide the sentence into two independent regions  $r_1 = \{w_{i-1/2}, w_{i-1/2+1}, \dots, t_i, \dots, w_{i+1/2}\}$  and  $r_2 = \{w_{j-1/2}, w_{j-1/2+1}, \dots, t_j, \dots, w_{j+1/2}\}$ , where  $l$  is the length of a region.

#### C. Regional LSTM

In this part, we introduce the novel regional LSTM in our approach. As Figure 1 illustrates, the regional LSTM receives a region sequential input matrix of a sentence to focus on the most important information of the specific target. The input of each LSTM unit is composed of the previous output of hidden layer and the current word and target embeddings, defined as:

$$\mathbf{E}_s = \mathbf{W}_h \cdot \mathbf{h}_{i-1} \oplus \mathbf{W}_t \cdot \mathbf{t}_i \oplus \mathbf{W}_x \cdot \mathbf{x}_i \quad (2)$$

where  $\mathbf{h}_{i-1}$  is the last hidden output,  $\mathbf{W}_h$  is the weight matrix of hidden output,  $\mathbf{t}_i$  is the  $i$ -th target embedding,  $\mathbf{W}_t$  is the weight matrix of target embedding,  $\mathbf{x}_i$  is the  $i$ -th word embedding,  $\mathbf{W}_x$  is the weight matrix of word embedding.

#### D. Sentence-level LSTM

In order to capture in-depth sentiment information of the sentence, and extract the long-distance dependency of the target across a sentence, sequential inputs composed of word and target embedding are fed into the sentence-level LSTM, as is demonstrated in Figure 1. The input of each LSTM unit is calculated in the same way as regional LSTM, as is shown in formula (2).

#### E. Gated merge layer

We attempt to combine the information extracted by regional LSTM and sentence-level LSTM to leverage the specific sentiment features in the sentence. The final output of regional LSTM and sentence-level are fed as input to the merge layer controlled by a gated operation:

$$\mathbf{H} = \mathbf{h}_r \otimes \sigma(\mathbf{W}_s \mathbf{h}_s + b) \quad (3)$$

where  $\mathbf{h}_r$  is the final output of regional LSTM,  $\mathbf{h}_s$  is the final output of sentence-level LSTM,  $\mathbf{W}_s$  and  $b$  are learned parameters.  $\sigma$  is the sigmoid function and  $\otimes$  is the element-wise product between matrices.

#### F. Model training

Finally, the output  $\mathbf{H}_m$  of merge layer is fed into a softmax layer to predict the probability distribution of sentiment.

$$y = \text{softmax}(\mathbf{W} \mathbf{H}_m + b) \quad (4)$$

TABLE I

STATISTICS OF THE DATASETS. DOMAIN OF PHONES AND CAMERAS HAVE NO NEUTRAL SENTENCE.

Dataset	#Positive	#Negative	#Neutral
REST-Train	1640	736	98
REST-Test	582	178	37
LAPT-Train	1631	1070	180
LAPT-Test	432	261	40
PHNS-Train	748	566	0
PHNS-Test	302	204	0
CAME-Train	802	442	0
CAME-Test	328	124	0

TABLE II

DETAILS OF HYPER-PARAMETER IN OUR EXPERIMENTS.

Hyper-parameter	Value
$l_2$ constrain	3
Mini-batch	32
Dropout	0.5
Length of region	11

Where  $\mathbf{W}$  and  $b$  are the parameters for softmax layer. We use the back propagation algorithm to train the model and optimize the model by minimizing the cross entropy error of sentiment classification:

$$loss = - \sum_{i \in D} \sum_{j \in C} \hat{y}_i^j \log y_i^j + \lambda \|\theta\|^2 \quad (5)$$

where  $D$  means all training instances,  $C$  is the number of sentiment categories,  $\hat{y}$  is the correct distribution of sentiment,  $y$  is the predicted sentiment distribution, and  $\lambda \|\theta\|^2$  is  $l_2$  regularization.

## IV. EXPERIMENTS

### A. Datasets and training

We conduct experiments on four datasets of two languages<sup>2</sup> from Semeval2016 Task 5 [22]. Statistics of the datasets are shown in Table 1. We remove sentences with no target which are out of the scope of our task.

We use Glove<sup>3</sup> [23] to initialize word vectors for English and Leipzig Corpora Collection<sup>4</sup> for Chinese. We use 300-dimensional word vectors in our experiments. We train all models with a mini-batch size of 32, dropout rate of 0.5,  $l_2$  regularization weight of 0.001, and the update rule of AdaGrad. The length of regions is 11 words, and we segment Chinese data first. The details of experiments hyper-parameters are shown in Table 2.

<sup>2</sup>The four domain datasets including restaurants, laptops, phones, and cameras, and the languages are English and Chinese. Sentiment classes of restaurants and laptops are positive, negative, and neutral, and domain of phones and cameras are positive and negative.

<sup>3</sup><http://nlp.stanford.edu/projects/glove/>

<sup>4</sup><http://corpora2.informatik.uni-leipzig.de/download.html/>

TABLE III

BINARY PREDICTION ACCURACY OF OUR RLSTM AND GH-LSTM FOR TARGET-BASED SENTIMENT CLASSIFICATION ON DIFFERENT DOMAIN DATASETS IN COMPARISON TO COMPARATIVE MODELS. BEST SCORES ARE IN BOLD.

Models	REST	LAPT	PHNS	CAME
SVM	81.97	75.90	70.36	76.33
LSTM	81.58	76.62	69.57	76.11
ATT-CNN	82.89	78.64	71.15	76.99
ATT-LSTM	84.87	<b>80.81</b>	72.53	78.98
RLSTM	81.97	75.61	70.36	75.22
GH-LSTM	<b>85.53</b>	80.66	<b>73.91</b>	<b>79.65</b>

### B. Comparison models

We compare our gated hierarchical LSTMs with several typical and state-of-the-art models, including SVM [14], LSTM [24], ATT-CNN [8], ATT-LSTM [10].

**RLSTM:** Region LSTM model. This model is part of the layered network model proposed in this paper, RLSTM model only use the region LSTM and is capable to exploit sentiment information for different targets. But it cannot get enough sentiment information of the whole sentence.

**GH-LSTM:** The complete model of our work, which is able to distinguish different targets information in the same sentence and capture the long dependency of target across the review.

**SVM:** A feature-based SVM classification model. This model got a better result in experiment of target-based customer reviews classification than the previous researches, but it needs some extra features.

**LSTM:** Standard LSTM without any attention of target that cannot infer the sentiment polarity of different targets in the same sentence exactly.

**ATT-CNN:** An attention-based CNN model that achieves state-of-the-art performance on sentence pairs modeling. We use a similar model for receiving the word embedding and target embedding in the experiment that is able to highly focus on the target in the process of training. But the structure of the model is relatively complex and the effect is quite dependent on the attention matrix.

**ATT-LSTM:** An attention-based LSTM that can concentrate on different parts of a sentence for different targets. This model achieves state-of-the-art performance on aspect-level sentiment classification. But the model needs a high training time cost.

### C. Comparative results

The binary prediction (positive and negative) results are shown in Table 3. Our models achieve the second best result on laptops dataset (ATT-LSTM achieve the best result) and achieve the best results on other 3 domain datasets. In addition, the results of experiment including neutral polarity category are presented in Table 4. We can find that, similar to the results of binary prediction, our models achieve better performance than other comparative models on all domain datasets, indicating that our model can discriminate the sentiment polarity

TABLE IV  
PREDICTION ACCURACY INCLUDING NEUTRAL CATEGORY OF OUR MODELS AGAINST OTHER MODELS. BEST SCORES ARE IN BOLD.

Models	REST	LAPT
SVM	77.85	70.94
LSTM	78.80	71.62
ATT-CNN	81.18	74.49
ATT-LSTM	82.69	76.26
RLSTM	79.29	71.49
GH-LSTM	<b>83.35</b>	<b>76.81</b>

of different targets and capture sufficient feature information in more complex sentiment categories via a gated hierarchical network. Another phenomenon is that the GH-LSTM which using hierarchical input layer and gated operation can improve the accuracy in contrast to RLSTM on all experiments, revealing that the hierarchical input layer of sentence-level LSTM and gated merge layer are valid in our approach.

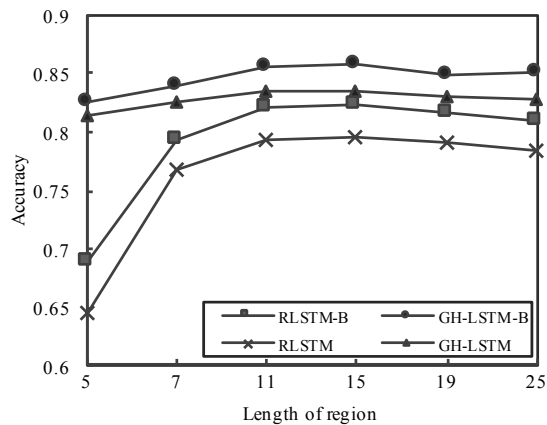


Fig. 2. Classification accuracy of our RLSTM and GH-LSTM with different length of region on restaurants dataset. “-B” represents binary prediction.

#### D. Effects of the length of region

In order to validate the effectiveness of dividing regions based on targets and the effects of using gated hierarchical LSTMs, we implement different lengths of region in GH-LSTM and RLSTM on restaurants dataset. As shown in Figure 2, before the length is 11, both GH-LSTM and RLSTM achieve better performance when the length of region is increased. But the results fluctuated when the length of the region is greater than 11. So the value of the length of the region is 11 in our experiment. We can also find that when the length is very small, GH-LSTM can also get a good prediction results, while RLSTM does bad when the length is 5, so that the gated hierarchical LSTM can improve the classification when the model lacks regional information.

#### E. Runtime analysis

We study the runtime of our approach and comparative models. We implement all these approaches based on the same

TABLE V  
RUNTIME (SECONDS) OF EACH TRAINING EPOCH ON THE RESTAURANTS DATASET.

Method	Time cost
LSTM	108
ATT-CNN	62
ATT-LSTM	324
RLSTM	84
GH-LSTM	167

neural network infrastructure, and run them on the same CPU and GPU server. As shown in Table 5, LSTM combining attention mechanism is indeed computationally expensive, the ATT-LSTM costs 324s during each training epoch. Our RLSTM is almost 4 times faster than ATT-LSTM and faster than basic LSTM, our GH-LSTM is also save half of the time contrast to ATT-LSTM.

#### V. CONCLUSION

We propose a novel deep gated hierarchical LSTMs model for target-based sentiment analysis. Our model not only can effectively identify the sentiment polarity of different targets, but also can obtain long distance dependencies of specific targets in the whole input sentence and extract more hidden information of specific targets. Finally, the experiment results on the 4 datasets of two languages illustrated the validity of our model by comparing with the excellent models in previous studies.

#### ACKNOWLEDGMENT

This work is partially supported by the National Natural Science Foundation of China (61772263, 61772014, 61572375).

#### REFERENCES

- [1] M. Pontiki, D. Galanis, J. Pavlopoulos *et al.*, “Semeval-2014 task 4: Aspect based sentiment analysis,” in *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Dublin, Ireland: Association for Computational Linguistics and Dublin City University, August 2014, pp. 27–35.
- [2] B. Pang, L. Lee *et al.*, “Opinion mining and sentiment analysis,” *Foundations and Trends® in Information Retrieval*, vol. 2, no. 1–2, pp. 1–135, 2008.
- [3] T. Nasukawa and J. Yi, “Sentiment analysis: Capturing favorability using natural language processing,” in *Proceedings of the 2nd international conference on Knowledge capture*. ACM, 2003, pp. 70–77.
- [4] E. Boiy and M.-F. Moens, “A machine learning approach to sentiment analysis in multilingual web texts,” *Information retrieval*, vol. 12, no. 5, pp. 526–558, 2009.
- [5] Y. Kim, “Convolutional neural networks for sentence classification,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, October 2014, pp. 1746–1751.

- [6] P. Liu, X. Qiu, and X. Huang, “Adversarial multi-task learning for text classification,” in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 1–10.
- [7] P. Zhou, W. Shi, J. Tian *et al.*, “Attention-based bidirectional long short-term memory networks for relation classification,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Berlin, Germany: Association for Computational Linguistics, August 2016, pp. 207–212.
- [8] W. Yin, H. Schütze, B. Xiang *et al.*, “Abcnn: Attention-based convolutional neural network for modeling sentence pairs,” *Transactions of the Association of Computational Linguistics*, vol. 4, no. 1, pp. 259–272, 2016.
- [9] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv preprint arXiv:1409.0473*, 2014.
- [10] Y. Wang, M. Huang, X. Zhu *et al.*, “Attention-based lstm for aspect-level sentiment classification,” in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, November 2016, pp. 606–615.
- [11] D. Tang, B. Qin, and T. Liu, “Aspect level sentiment classification with deep memory network,” in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, November 2016, pp. 214–224.
- [12] B. Liu, “Sentiment analysis and opinion mining,” *Synthesis lectures on human language technologies*, vol. 5, no. 1, pp. 1–167, 2012.
- [13] G. Qiu, B. Liu, J. Bu *et al.*, “Opinion word expansion and target extraction through double propagation,” *Computational linguistics*, vol. 37, no. 1, pp. 9–27, 2011.
- [14] S. Kiritchenko, X. Zhu, C. Cherry *et al.*, “Nrc-canada-2014: Detecting aspects and sentiment in customer reviews,” in *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, 2014, pp. 437–442.
- [15] T. H. Nguyen and K. Shirai, “Phrasernn: Phrase recursive neural network for aspect-based sentiment analysis,” in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, September 2015, pp. 2509–2514.
- [16] L. Dong, F. Wei, C. Tan *et al.*, “Adaptive recursive neural network for target-dependent twitter sentiment classification,” in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Baltimore, Maryland: Association for Computational Linguistics, June 2014, pp. 49–54.
- [17] R. Lin, S. Liu, M. Yang *et al.*, “Hierarchical recurrent neural network for document modeling,” in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, September 2015, pp. 899–907.
- [18] J. Li, T. Luong, and D. Jurafsky, “A hierarchical neural autoencoder for paragraphs and documents,” in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, July 2015, pp. 1106–1115.
- [19] Q. Li, T. Li, and B. Chang, “Discourse parsing with attention-based hierarchical neural networks,” in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, November 2016, pp. 362–371.
- [20] S. Ruder, P. Ghaffari, and J. G. Breslin, “A hierarchical model of reviews for aspect-based sentiment analysis,” in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, November 2016, pp. 999–1005.
- [21] J. Wang, L.-C. Yu, K. R. Lai *et al.*, “Dimensional sentiment analysis using a regional cnn-lstm model,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Berlin, Germany: Association for Computational Linguistics, August 2016, pp. 225–230.
- [22] M. Pontiki, D. Galanis, H. Papageorgiou *et al.*, “Semeval-2016 task 5: Aspect based sentiment analysis,” in *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. San Diego, California: Association for Computational Linguistics, June 2016, pp. 19–30.
- [23] J. Pennington, R. Socher, and C. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, October 2014, pp. 1532–1543.
- [24] X. Wang, Y. Liu, C. Sun *et al.*, “Predicting polarities of tweets by composing word embeddings with long short-term memory,” in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, July 2015, pp. 1343–1353.

# Does Ad-Context Matter on the Effectiveness of Online Advertising?

Caihong Sun, Meina Zhang, Meiyun Zuo\*  
Key Laboratory of Data Engineering and Knowledge Engineering,  
School of Information, Renmin University of China  
Beijing, China  
chsun@ruc.edu.cn    zhangmeina@ruc.edu.cn    zuomy@ruc.edu.cn

**Abstract**—In this paper, we focus on examining the effects of Ad-context on the click-through rate (CTR) for the online advertising. Many researches have shown that ad-context congruity is a key factor to CTR, but the features of ad-context are rarely introduced in CTR prediction algorithms. By leveraging data from various sources, and using text mining and sentiment analysis techniques, our proposed approach extracts three types of features (i.e. users, advertisements and ad-context features) to predict CTR. User features describe “Who” is browsing the webpage, advertisement features depict “How” an ad serving is, and ad-context features include “What” the product is, “Where” an ad displays, as well as the “Mood” of the context. Experiment results show that our proposed approach outperforms the benchmark models by introducing ad-context features. Novel ad-context features we proposed make good contributions to the prediction of CTR. The research highlights the power of ad-context in online advertising CTR prediction. Moreover it gives an insight into supporting effective advertising and improving users’ satisfaction.

**Keywords**—Click-Through-Rate, Ad-Context, Sentiment Analysis, Feature Engineering, Online Advertising

## I. INTRODUCTION

With the increasing of Internet users, online advertising becomes a marketing gold mine and generates huge advertising revenues. For example, the Interactive Advertising Bureau (IAB) announced on December 20th 2017 that digital advertising revenues in the US for the first half of 2017 surged to \$40.1 billion, a 23 percent year-over-year rise from the \$32.7 billion reported during the same timeframe in 2016. The huge commercial value of online advertisement in some extent depends on whether users click on the advertisement or not. The click-through rate (CTR), as a major measurement of online advertising, can be used to improve users’ satisfaction and support effective advertising. It plays an important role in online advertising. Academia and industry have also invested a lot of effort in the CTRs prediction model of advertising.

The features used to predict CTR in the existing literatures generally could be classified into three kinds: advertisement features, users’ features and ad-context features. Many researchers use the advertisement features, such as the size, the position of an advertisement, and users’ features such as demography, the equipment type and user’s behavior history for CTR prediction. But few researchers introduce ad-context features into CTR prediction algorithms since they are not easy to be obtained. There are two contrasting theories to explain the relationship between ad-context congruity and memory for ad content, where Ad-context congruity is defined as the degree to which advertising material is thematically similar to adjacent editorial content [1]. Priming theory leads to increased memory of ads that are similar to their context, but interference theory is prone to decrease memory when ads resemble their context. Although contrasting, both priming and interference theories show that ad-context matters in advertisement effectiveness [1-3]. In the study of contextual advertising, Chakrabarti et al. measured the ad-context congruity by using the cosine distance between the keywords of the advertisement and the webpage the ads placed [4]. By adding this information into CTR prediction, it gained 25% prediction accuracy improvement. But what the ad-context features are and how to extract them?

In this paper, our main research question is to extract ad-context features and introduce them into the state-of-art prediction algorithms to enhance the performance of CTR prediction. The main contributions of this study are in two areas:

First, this work demonstrates a prediction framework in which the different sources of data could be collected, fused and analyzed to train CTR prediction algorithms, such as Logistic Regression (LR), Gradient Boosting Decision Tree (GBDT) and Factorization Machine (FM). In this study, we collected a SUV automobile advertisement dataset and crawled the webpages that the ads placed to demonstrate the process and the effectiveness of our prediction framework.

Second, our study proposed several novel ad-context features: “What”, representing the attributes of a product on a web page (i.e. price, the origin of production and product type etc.), “Where”, ad serving channels and the “Mood” of the web contents. Although many features have already been studied and introduced in CTR prediction for better performance in many aspects, the effect of ad-context features is rarely considered into CTR prediction until now, especially the features in product and mood dimensions. We showed that these features make

great contributions to the system's performance and help to explain that the ad-context is a key factor to the effectiveness of online advertising.

Accuracy, logLoss and AUC (the Area under Receiver Operating Characteristics (ROC) Curve), will be employed as the evaluation criteria for the performance in our comparison study.

The paper proceeds as follows. Section 2 reviews the research literatures. Section 3 outlines the system framework and presents the process of feature engineering. Section 4 describes the data collection and shows the experiment studies. Section 5 concludes the paper with future remarks.

## II. LITERATURE REVIEW

In this study, we draw mainly on three streams of researches in online advertising: (i) CTR prediction models, (ii) features used in CTR prediction, and (iii) ad-context.

### A. CTR Prediction Models

Many researchers and advertisers to predict CTR use logistic Regression. The major difference of these studies is the features they used for prediction. Robinson et al. studied how 7 features of banner ad affect the online advertising effectiveness by using LR, and they found that in network game domain, the size of ad, promotion and game information were good for clicking, but the dynamic banners, the corporate brands or flags had no effect on clicking [5]. Chakrabarti et al. used LR and studied how the congruity between ad and web contents influences the CTR in contextual advertising [4]. Besides the classic LR model, Dave and Varma applied GBDT to estimate CTR [6], while the main idea of GBDT is inspired by probably approximately correct learning model proposed by Valiant. The GBDT algorithms are prone to over-fitting, since they ignore the correlations among features. FM algorithm proposed by Rendle [7] overcomes this drawback, and FM can handle the dependent features well. Zhang et al. used RNN (Recurrent Neural networks) to predict the CTR in search advertisements [8]. RNN outperforms LR when data volume is large. Moreover, to improve the accuracy of CTR prediction, one strategy is to fuse several models together. The simple one is to weighted add the results of each models, or stacking these models. Tian et al. use FM+GBDT, LR+GBDT to evaluate the effectiveness of feature selections [9].

### B. Features in CTR Prediction

The features used by CTR prediction could be classified into three kinds: the features of advertisement, the features of users and ad-context features. These three kinds of features constitute the feature triple<advertisement, user, context>. The advertisement features, such as the size, position and dynamics of an advertisement, are often used to predict CTR prediction [5]. The features of users i.e. the demography of users, the equipment type, user behavioral data and user ID, are introduced into CTR prediction models to improve the accuracy of CTR prediction [10]. Ad-context features are rarely used in

literatures since they are not easy to be acquired. Richardson et al. [11] introduce the degree of congruity between the search keywords and an advertisement to estimate the CTR of a new advertisement. Chakrabarti et al. [4] improve the prediction effectiveness with 25% by introducing the content congruity between an advertisement and the web page it placed. In addition, many CTR prediction literatures try to introduce more features to improve the prediction accuracy [12-13]. To sum up, the ad-context features are not well explored in CTR prediction yet.

### C. Ad-context

Although ad-context is considered to be important in CTR prediction, to the best of our knowledge, there are few literatures discussed about ad-context features engineering. In the studies of ad-context in online advertisement, many researchers focus on exploring the effects of ad-context congruity which is defined as the degree to which advertising material is similar to adjacent editorial content [1]. The research on the ad-context congruity is mainly focus on the following perspectives: structure e.g. background color [14], mood [16], semantic e.g. whether deliver an automobile ad to the auto, finance or sport website [17]. All these contextual factors will influence the effectiveness of advertising. However, articles that study contextual factors from the web product dimension are basically not available. Especially on some vertical website, the semantic and structural factors are completely the same with those of the target advertising. Since the product features such as price have impact on advertisement effects, the product information on the web page for a vertical advertising could possible affect users' click. Psychological research shows that congruity of the attitude related, has important impact on the individual cognition and attitude formation [14-15]. Mood has impact on the audience's attitude to an advertisement and does influence the effectiveness of an advertisement [16]. Some studies [4, 11] demonstrate that semantic or thematically contents influence the effectiveness of an advertisement a lot. To sum up, ad-context does have impact on advertisement effect. Since ad-context features are rarely discussed from the feature engineering aspects, in this study we try to extract ad-context features from web contents and utilize them to improve the CTR prediction.

## III. CTR PREDICTION FRAMEWORK AND FEATURE ENGINEERING

### A. CTR Prediction Framework

Figure 1 illustrates the framework of our CTR prediction. The first phase is data acquisition. We picked two complementary data sources—log data from an advertising agency and their ad serving websites. We can acquire the features of ads, the impression and click data and some user features from the log data. And the ad serving website provides the ad-context features such as serving channels, product features on the website and mood of a webpage. As for data collection methods, the two sources are different as well. We use API to extract data

from the log data provided by the advertising agency, while crawl the contents of the website via python.

In the second phase, data from both sources are cleaned, transformed, consolidated, and stored in a database. In particular, we use text mining and sentiment analysis techniques to get the mood feature of the web contents and then stored them in database.

The third phase, “feature engineering”, involves using the acquired data to construct features that will ultimately be used to train a CTR predictive model. Specifically, we classify various features into three groups, which will be described in detail later.

In the fourth phase, a predictive model can be trained. The Experimental Result Analysis section will discuss our experiments and research questions in detail.

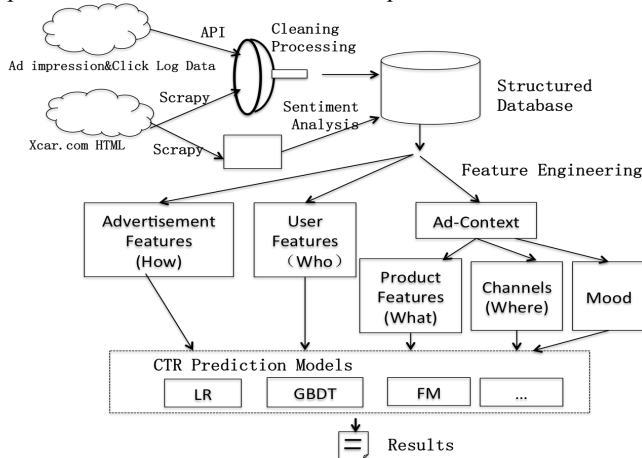


Figure 1. The Framework of CTR Prediction Feature Engineering

### B. Feature Engineering

In our study, we derived three groups of features: “who” features, “how” features, and ad-context features. Ad-context features consist of ad serving channels (Where), and product features (What) and mood features.

“Who” features consist of the features describing users who could view the ads: UserID, the number of visited pages, the number of browsed auto types, city, the number of visited channels, and type of the equipment a user used.

“How” features describe how an advertisement displays: AdID, Ad’s size, position, the number of impressions, the number of impression days and so on.

“Ad-context” features consist of ad serving channel (Where), product features (What) and mood feature. The serving channel captures the channel targeting aspect, where the ad was served in a web site. Since our study focuses on serving ads on the vertical media, different channels provide consumers with different types of information, and consumers may browse different channels at different consumption stages. Therefore, the click through rate of ads will also vary in different channels. While product features describe the semantic aspect of web contents by adding the product dimension which the ad described. The difference of product features between the

advertising target and the webpage has impact on the consumers. Here the product features are described as product ID, price, the origin of production and product type since some studies show that the price, the original site, and the automobile type are the three main factors that affect the car purchase decision [18]. Furthermore, the mood feature describes the mood of a webpage via the negative or positive attitude, which captures the emotion of the web contents.

## IV. EXPERIMENTAL DESIGN AND ANALYSIS

In this section, we conduct several experiments to validate the effects of ad-context features based on LR, GBDT and FM models. We do the experiments by adding ad-context features or not to demonstrate the effectiveness of our proposed method.

### A. Data Description

There are two data sources in our research. One is the log data provided by advertising agency. They provide us the features of advertisements, some feature of users and the log data. The other data is crawled by us from the websites where the ads delivered to access the ad-context features. In our study, we focus on a new SUV auto ads activity. The ad-serving period is two weeks, from December 23rd 2014 to Jan 5th 2015. Xcar.com is the ad-serving website, an auto vertical media where the serving channels include 4 channels: information, shopping guide, test-driving and car usage.

We extract user and advertisement features from the data provide by the advertising agency, and the ad-context features from the web contents we crawled. Serving channel feature (“Where”) has 4 categorical values according to the channels the advertisement displayed on. These four channels provide information consumers they sought for at different stages of their purchase decision. The information channel provides some basic information about cars, which satisfied the information need of consumers in the early stages of purchase decisions. The shopping guide channel provides consumers with specific information on a topic, such as price, auto type, for comparison. Test-driving channel delivers some driving experience of a certain car type. The car usage channel provides maintenance information after purchase. The source of mood factors is from the textual context of the web pages. We apply the dictionary-based sentimental word analysis algorithm to the text content of the webpages we crawled. In our experiments, we adopt the sentimental lexicon from the information retrieval laboratory of Dalian University of Technology for sentimental analysis. The mood feature is a numeric type, while a negative or positive value represents negative or positive mood respectively. The product features (“What”) consist of price, the origin of production and product type.

The price of SUV, the target product of advertising activity, in the ad activity is about 129.9-169.9 thousand RMB. The total impression of the ad is 440812, the number of clicks is 586. There are 251199 users, and 501 users click the ads, as shown in Table 1.



TABLE I. STATISTICS OF THE DATASETS

Number of Impression	440812
Number of Clicks	586
Number of users	251199
Number of clicked users	501

**B. Evaluation Metrics**

For CTR prediction, we tried 3 algorithms: LR, GBDT and FM. We evaluate the overall performance based on 3 metrics:

- 1) *Classification accuracy*, which is the percentage of correctly predicted instances;
- 2) *Logloss*, a measure defined as the negative log-likelihood of the true labels given a probabilistic classifier’s predictions, the smaller the value is, the better the prediction;
- 3) *AUC*, the curve plots the true positive rate against the false positive rate. An AUC of 1 means a perfect classification whereas 0.5 refers to a random guess. Being more robust against prior distributions, AUC is considered by many researchers to be one of the best indicators of a classifier’s performance.

**C. Three State-of-art CTR Prediction Algorithms**

To illustrate the effectiveness of our proposed method, we choose the following three state-of-the-art CTR prediction algorithms for comparisons.

- **LR** (Logistic Regression): It is a widely used method for CTR prediction, especially used as the baseline model for comparisons with other prediction methods [4][6].
- **GBDT** (Gradient Boosting Decision Tree): It is a non-linear model, an ensemble decision tree inspired by boosting. It has the advantages to discover the raw features and cross features. GBDT usually used in CTR prediction fused with other methods such as LR and FM [9][18].
- **FM** (Factorization Machine): Factorization machines (FMs) are a generic approach that allows mimicking most factorization models by feature engineering [7]. FM is able to perform a regression model whose task is to estimate a function  $y: \mathbb{R}^n \rightarrow T$  from a real valued vector  $x \in \mathbb{R}^n$  to a target domain T and it could model all single and pairwise interactions between the input variables by using factorized interaction parameters.

**D. Experimental Design and Analysis**

To evaluate the effectiveness of ad-context features, we apply the features with and without ad-context features to three algorithms respectively: LR, GBDT and FM. Since the click rate is very low, i.e. the positive and negative samples are extremely misbalanced. In our experiments, we sample the ratio of positive and negative from 1:1 to 1:9, and we found the sampling ratio of positive and negative samples mattered just a little bit in all three algorithms, When the ratio is 1:5, the performance is better. As shown in Figure 2, with the ratio of sampling in positive and negative data from 1:1 to 1:9, the performance of LR varied a little (we can find the similar results in both GBDT and FM). In the following experiments, we set

the positive and negative sampling ratio as 1:5.

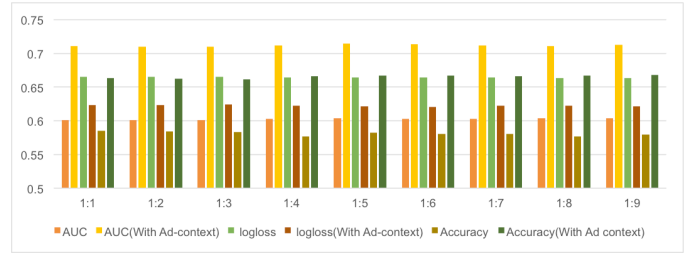


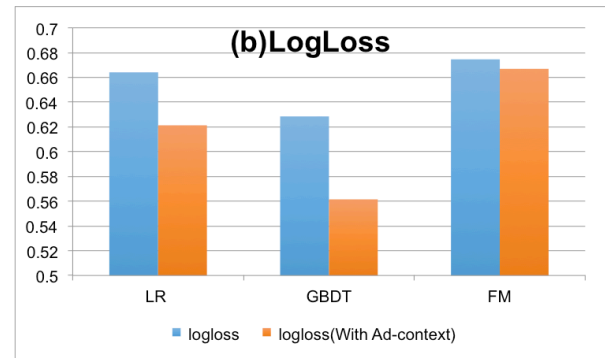
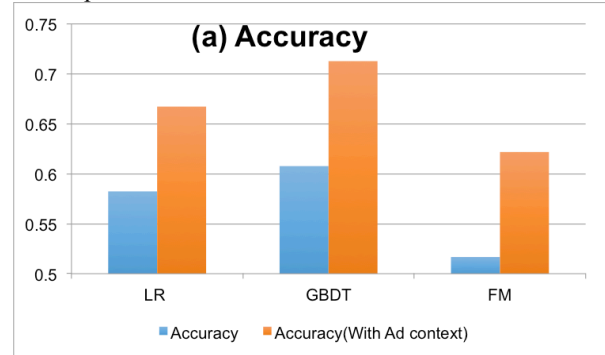
Figure 2. Performance with and without ad-context features by using LR in different sampling ratio

Experiments are designed to answer the following two research questions.

RQ1: Does Ad-context matter in CTR prediction of online advertising?

We design comparison experiments by adding ad-context features or not. The baseline is to use features of advertisement and users to predict CTR, without any ad-context features.

As shown in Figure 3, by introducing ad-context features, all the three evaluation metrics get better in LR, GBDT and FM. The accuracy improves a lot (shown in Figure 3 (a)), the logloss decreases (shown in Figure 3(b)) and the AUC increases (shown in Figure 3(c)) in all three algorithms. In our experiments, GBDT outperforms LR and FM.



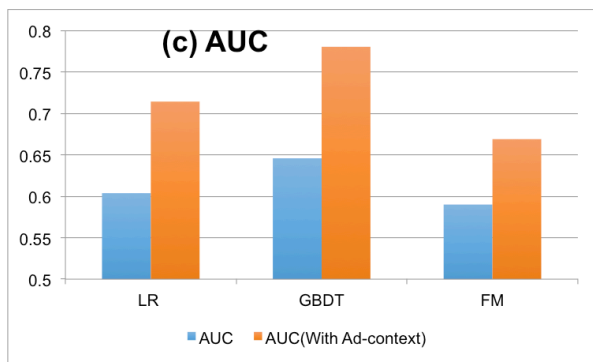


Figure 3. Comparisons in (a) Accuracy (b)LogLoss and (c) AUC with and without ad-context features

To sum up, ad-context features do matter in the prediction of CTR in our experiments.

RQ2: Are all dimensions of ad-context we proposed valuable?

To illustrate the effect of “Where”, “What” and “Mood” dimensions in ad-context, we design experiments by adding each of them into baseline in LR algorithm, and we find (shown in Figure 4) that “Where” and “What” features improve all three evaluation metrics individually, but the “Mood” feature does not. Furthermore, by introducing all three kinds of ad-context (Where, What and Mood), the algorithm gets better performance.

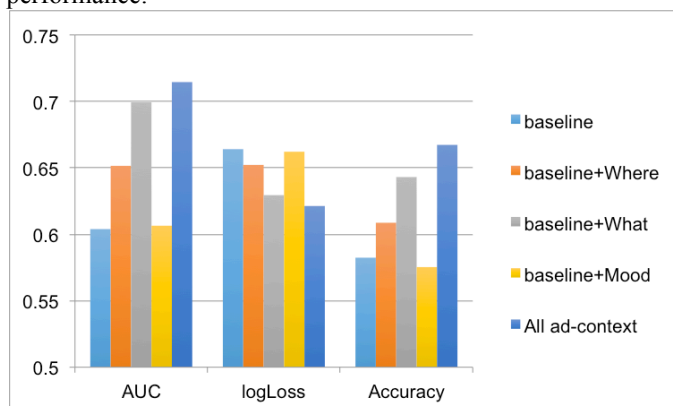


Figure 4. The Effects of “Where”, “What” and “Mood” Features in LR algorithms

As shown in Figure 4, we can see that “What” features, i.e. the features of products in our experiments, contribute a lot in CTR prediction. By adding all three kinds of ad-context features, the algorithm gets much better performance. The similar findings are found in GBDT and FM. The experiments demonstrate that our proposed ad-context features are very effective on CTR prediction.

## V. CONCLUSIONS AND FUTURE WORK

In this study, we proposed a CTR prediction framework by collecting data from different sources. By employing text mining and sentiment analyzing, we extract ad-context features including “Where” the ads displayed, “What” is the product and

“mood” of the web contents. CTR prediction improves a lot with the novel ad context features we proposed in LR, GBDT and FM algorithms. Different from the application of context factors in search advertising mentioned in the previous literature review, this article attempts to apply context factors in general banner ads. Existing studies aim to explore how ad-context congruity affects the effective of online advertising. Here, we are not considering congruity but introducing context attributes to the CTR prediction algorithm. Our study demonstrates that ad-context is valuable to CTR prediction and gives out a practical way of ad-context feature engineering. Our work gives an insight on ad-serving channels decision-making and users’ satisfaction improvement. There are also several directions for future research. Our future work includes testing our method on more datasets and CTR prediction algorithms. More ad-context features could be extracted and drawn into further study.

## ACKNOWLEDGEMENT

This work was supported in part by the Fundamental Research Funds for the Central Universities under Grant 10XNJ065, part by the National Natural Science Foundation of China under Grant 71771210, part by the Beijing Natural Science Foundation under Grant 9182008, part by National Social Science Foundation of China Major Program under Grant 13&ZD184. Meiyun Zuo is the corresponding author of this paper.

## REFERENCES

- [1] Gunter, B., et al., Children’s memory for television advertising: effects of programme–advertisement congruency. *Applied Cognitive Psychology*, 2010. 16(2): p. 171-190.
- [2] Kamins, M.A., L.J. Marks, and D. Skinner, Television Commercial Evaluation in the Context of Program Induced Mood: Congruency versus Consistency Effects. *Journal of Advertising*, 1991. 20(2): p. 1-14.
- [3] Yinon, Y. and M.O. Landau, On the reinforcing value of helping behavior in a positive mood. *Motivation & Emotion*, 1987. 11(1): p. 83-93.
- [4] Chakrabarti, D., D. Agarwal, and V. Josifovski. Contextual advertising by combining relevance with click feedback. in *International Conference on World Wide Web*. 2008.
- [5] Robinson, H.R., A. Wysocka, and C. Hand, Internet advertising effectiveness: The effect of design on click-through rates for banner ads. *International Journal of Advertising*, 2007. 26(4): p. 527-541.
- [6] Dave K S, Varma V. Learning the click-through rate for rare/new ads from similar ads[C]// *International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2010:897-898.
- [7] Rendle, S. Factorization Machines. in *IEEE International Conference on Data Mining*. 2010.
- [8] Zhang, Y., et al., Sequential Click Prediction for Sponsored Search with Recurrent Neural Networks. 2014: p. 1369-1375.
- [9] Tian, C., et al., Research and implementation of feature extraction methods of Internet CTR prediction model. *Computer Application Research*, 2017. 34(2): p. 334-338. (in Chinese)
- [10] Zhu, Z.A., et al., A novel click model and its applications to online advertising. 2010: p. 321-330.
- [11] Richardson, M., E. Dominowska, and R. Ragno. Predicting clicks:estimating the click-through rate for new ads. in *International Conference on World Wide Web*. 2007.
- [12] Ta, A.P. Factorization machines with follow-the-regularized-leader for CTR prediction in display advertising. in *IEEE International Conference on Big Data*. 2015.

- [13] Zhang, S., Q. Fu, and W. Xiao, Advertisement Click-Through Rate Prediction Based on the Weighted-ELM and Adaboost Algorithm. 2017. 2017(1): 1-8.
- [14] Chaiken, S., E.M. Pomerantz, and R. Giner-Sorolla, Structural consistency and attitude strength. *Attitude Strength Antecedents & Consequences*, 1995.
- [15] Berkowitz, L., Mood, Self-Awareness, and Willingness to Help. *Journal of Personality & Social Psychology*, 1987. 52(4): p. 721-729.
- [16] Faseur, T. and M. Geuens, Different Positive Feelings Leading to Different Ad Evaluations: The Case of Coziness, Excitement, and Romance. *Journal of Advertising*, 2006. 35(4): p. 129-142.
- [17] Gunter, B., et al., *Children's memory for television advertising: effects of programme-advertisement congruency*. *Applied Cognitive Psychology*, 2010. 16(2): p. 171-190.
- [18] Engel, J.F., R.D. Blackwell, and P.W. Miniard, *Consumer Behavior (8th Eds.)*. 1995.
- [19] Chapelle O, Manavoglu E, Rosales R. Simple and scalable response prediction for display advertising [J]. *ACM Transactions on Intelligent Systems & Technology*, 2013, 5 (4): 1-3

# Analyzing The Impact Of Feedback In GitHub On The Software Developer's Mood

Mateus Freira, Josemar Caetano, Johnatan Oliveira, Humberto Marques-Neto

Department of Computer Science  
Pontifical Catholic University of Minas Gerais (PUC Minas)  
Belo Horizonte, Brazil

{mateus.freira, josemar.caetano, johnatan.oliveira}@sga.pucminas.br,  
humberto@pucminas.br

## Abstract

*Software development depends on cooperation between people, and the way it works can define the future of the software project. Developers emotions affect their productivity and way they work, yet there is little information about how developers can influence the mood of each other. As a first step toward understanding how feedback may affect the developers' sentiment, this paper analyzes the mood variations on more than 78k pull requests and 268k pull comments on GitHub. We found that in 31.16% of the cases the developers presented a significant mood variation within one hour when receiving feedback on their pull requests. The variation reduces to 18.16% when evaluating one day before and after the commentary. In software projects with less than 34k lines of code, the number of developers that never contribute again after receiving a negative comment on the first pull request is 10.97%; this number more than doubles to 24.02% when evaluating projects with more than 197k lines of code.*

**Keywords:** Developers emotions, Developers mood, GitHub, Open source, Subjective well-being.

## 1 Introduction

Humans are one of the most valuable resources in software projects. Besides the technical knowledge needed for a successful project, it is essential to have good teamwork[7]. Staats [15] shows that increasing the teams familiarity decreases the number of defects, reduces budget deviation, and yields a 10% in performance improvement from the clients' perspective. Regarding the importance of keeping the team's members, Voices [17] warned that unhappiness could lead the developers to quit their company/project endeavors. Additionally, emotions can affect positively or negatively, the developers' productivity, creativity, and task quality[4, 6]. Understanding the mood variations of the developers on a software project, as well as the effects of feedback on the team members, is relevant to help the projects leaders to take proactive actions in increasing their team engagement and familiarity and, therefore, improving productivity.

GitHub is a social network and code hosting provider

that hosts more than 71 million projects<sup>1</sup> at present. The main GitHub feature is not only the code hosting but also project manager features, such as issues and pull request controls. These features promote discussions among the users, which is related to the reported bug and the requested feature or even connected to a code path that a developer wants to merge in a repository. We collected pull requests, their comments, and the profile of the developers, related to any of those interactions, to perform our research.

Recently, researchers have published several papers, regarding developer sentiment analysis, on GitHub interactions[5, 7, 8, 9, 12, 14]. However, what causes positive or negative variances in the developers sentiment, as well as the duration of the variation, is not entirely understood yet; many external influencing factors remain unexplored.

Our study contributes to a better understanding of developers' mood variations by analyzing the developers at the time they are submitting pull requests to a repository, and how other developers' comments may influence their mood. We highlight that we *cannot* establish a strong causal relationship between a comment and a developers' mood variation, as events outside of our data set might have influenced such variation. We empirically analyzed the 100 most popular java projects on GitHub and their more than 226k pull request comments. We applied SentiStrength [16] to calculate the sentiment expressed in each pull request and comment, and we then calculated the subjective well-being (SWB) [3] to obtain the mood variations before and after receiving a feedback from another developer. Finally, we analyzed the SWB to understand how the feedback may affect the developers' mood.

The rest of this paper is outlined as follows. We first present the techniques we used to perform the sentiment analysis and to calculate the sentiment influences in Section 2. We show the experiment design in Section 3. In Section 4 we present the results of our analysis, We present the related work in the Section 5. and finally, we conclude and present the plans for future works in Section 6.

---

<sup>1</sup><https://github.com/about>

## 2 Background

Reminding that our present goal is to investigate developers' mood variation when receiving feedback from others, we present an overview of the central concepts we have used to hit our goals. We first introduce the GitHub concepts, then we present the sentiment analysis approach we have used, then we show the changes we have made on the tool's dictionary to better address software engineering texts, and finally, we present the sentiment state and change measurement metrics we have used.

**GitHub:** Here we present some GitHub concepts that will help gain a better understanding of the paper. Next, we pick the features that relate the most to the present work.

**The Repository** represents the project itself; it contains the code, documentation, and also aggregates all the project interactions (issues, pull requests and commits). Repositories can be private or public; however, in this paper, all the repositories we have used are public, and therefore, all their information is available on the GitHub API [2].

A **pull request** is a proposed code change to a repository submitted by a developer. Once a pull request is open the project members can comment and either accept or reject it. If the pull request is accepted, the code integrates them into the repository code; if it is rejected, it is discarded. Pull requests have their discussion forum where the developers can comment on the changes or ask for improvements or changes before merging it. A pull request can contain one or many commits [2].

A **commit** consists of a change in one or many files, enabling developers to track the changes they have made. Commits usually contain a message with a short explanation about the change that it contains [2].

**Author association** is the association between the developers who are commenting or opening a pull request, and it represents the role of the developer in the repository. The author association's possible values are: collaborator (has been invited to the repository but has not committed anything yet), contributor (has been invited and has at least one commit), member (is a part of organization that owns the repository on GitHub), none (has no relationship with the repository), and owner (owner of the repository), we limited this list to the values that are present in our data set [1].

**GitHub pull request flow** consists of some developer (any GitHub user) with or without a relationship with the repository, who wants to integrate a code change to the code base. Let's suppose there is a bug and a developer wants to fix it, he/she needs to: 1) clone the project, 2) create a branch locally, 3) commit the changes to solve the bug, and 4) submit a pull request to the central repository. Figure 1 presents this process. After opening the pull request, other developers can comment on it, asking for changes or endorsing the changes made. That is why these comments

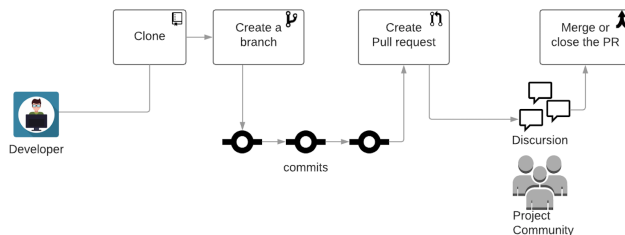


Figure 1: GitHub Pull request flow.

often become discussions among developers relating to the code change submitted. For that matter, the most critical interactions for us are after opening the pull request, when the discussions start. At this point, we compare the developer comments, before and after another developer's comment, to understand if the comment caused any mood variation on the developer who wanted to integrate his/hers code to the repository.

**Sentiment analysis:** Sentiment analysis is a common task when evaluating social network interactions. Ribeiro et al. [11] presented benchmark testing 24 sentiment analysis tools in 18 labeled tweet data sets. Their results showed that the SentiStrength tool [16] presented the best results in most of the datasets. Additionally, the previous work [7, 8, 14] have successfully used SentiStrength [16] to perform sentiment analysis of software engineering interactions, such as commits, issues, and pull requests. Therefore, in this paper, we decided to use SentiStrength to perform the sentiment analysis.

SentiStrength uses a lexicon approach based on a dictionary of words and idiomatic expressions to detect two sentiment polarizations, negative (from -1 slightly negative to -5 very negative) and positive (from 1 slightly positive to 5 very positive). By that, it provides the overall sentiment of the sentence (scale), subtracting the negative sentiment from the positive sentiment [16]. We used SentiStrength to extract the sentiment of each pull request and comment.

However, Novielli et al. [10] warned that using any sentiment tool to evaluate sentiment on software engineering artifacts without any change might result in an inadequate analysis because some terms considered as negative in other social networks are natural when analyzing technical texts. For example, the word 'static,' which in the SentiStrength default dictionary is considered as -2 negative, is used most of the time by developers to reference a method or field of a class. Additionally 'static' is a common term in some programming languages such as Java. Based on the Novielli et al. [10] suggestion, and in the previous work [14, 8], we decided to change the default SentiStrength dictionary to evaluate our data set better.

**Sentiment Analysis in Software Engineering:** Using any sentiment analysis tool without any change will, as we had introduced, result in an inadequate analysis. For that matter, we decided to change the default SentiStrength dictionary to address the software developer field better. We performed this process by checking the results of the classification manually and removing the common terms from the dictionary, classified as negative or positive. The Table 1 shows the modifications we have made in the dictionary; the words in the first column, *Words*, are the words that we have modified; the second column, *Original Value*, shows the value that the words have in the original SentiStrength dictionary; the third column shows the new values that we had set to the words, and the last column shows a short explanation why we had changed that group of words.

Words	Original Value	Change	Reason
broke*, fail	-2	0	Usually is a reference to the build status and has no sentiment
bug, defect, error, missing, mock,	-2	0	No sentiment related just reference fact
constrain*, drop, kill, static	-2	0	Common term in development with no sentiment expressed
Default, exit	-2	0	A common term in development with no sentiment expressed
garbage, vagrant, storm	-3	0	A common term in java projects with no sentiment expressed
revert	-2	0	Common term across GitHub social network
not working	-4	0	Idiomatic expression that most of the times have no sentiment expressed

Table 1: SentiStrength dictionary changes

**Mood Variation:** To evaluate the influences of others in the developer mood, we decided to use a metric called subjective well-being initially presented by Bollen et al. [3] and used in the Twitter social network to measure mood propagation. We used the technique to calculate the state of developer sentiment by analyzing a window of time. The subjective well-being ( $S(d)$ ) of a developer is given by subtracting the number of positive comments from the number of negative comments, divided by the number of positive comments plus the number of negative comments. This way, the  $S(d)$  value varies from -1 to 1; Equation 1 shows the  $S(d)$  equation. Once  $S(d)$  gives the developer sentiment on a specific window of time, we use the metric to evaluate the sentiment change by calculating the difference between before ( $S_b(d)$ ) and after ( $S_a(d)$ ) another developer comment on a pull request. The metric of mood variation ( $SC(d)$ ) varies between -2 and 2, and Equation 2 presents its equation. As previously mentioned, we *cannot* establish a strong causal relationship between a comment and the developer mood variation, given the possibility of external influences. However, by analyzing different windows of time (1, 2, 4, 8 hours and one day) we intend to reduce or mitigate such a problem.

$$S(d) = \frac{N_p(d) - N_n(d)}{N_p(d) + N_n(d)} \quad (1)$$

$$SC(d) = S_b(d) - S_a(d) \quad (2)$$

### 3 Experiment Design

This section presents the steps taken in our experiment, starting with a short presentation of our research questions, followed by data set and finally the sentiment mining.

**Goal and research questions:** As we previously mentioned the primary goal of this paper is to analyze the impact of feedback on GitHub in developers mood and how the influence behaves in time. Therefore, we formulate the following research questions (RQ):

**RQ1:** Can a developer change the sentiment of another developer with a pull request comment?

**RQ2:** Does the role of the developer in the project change the intensity of influence he/she has on the sentiment of another developer?

**RQ3:** How does the influence behave over time? Does the behavior change depending on the comment sentiment?

**RQ4:** Do negative comments on the first pull request lead to quitting the project?

**Data Set:** We collected the data from the GitHub API. We first obtained the most popular java projects from the API<sup>2</sup>. GitHub limits the search to the first 1,000 results. To get the most popular projects, we sorted the results by stars. We decided to remove all the projects with less than 1,000 lines of java code because they were probably documentation or experimental projects. After filtering out projects smaller than 1,000 lines of code, 930 projects remained on the data set. After filtering the projects, we collected the project interactions, pull requests, pull request comments, pull request reviews and commits from the GitHub API. Then we filtered the 100 most popular projects among the 930 filtered in the first filter to investigate more deeply. At the end of the data collection, our data set contained 100 projects, 555,665 commits, 78,475 pull requests, 226,446 reviews, 240,060 pull comments and 15,865 developers.

**Sentiment mining:** After collecting the data, we applied the SentiStrength [16] with the changes in the dictionary we presented previously, in all the interactions. We noted that the developers express less sentiment on the commits interactions, (15.52% of the commits have some sentiment expressed), on the other hand, 54.32% of the pull comments express some emotion Figure 2 shows the percentage for all the kinds of interactions. In all the cases the rate of positive sentiment is more significant than the rate of negative, on pull comments 40.5% are positive against 13.82% negative, and in the commits, the difference is small (7.86% positive,

<sup>2</sup><https://developer.github.com/v3/projects/>

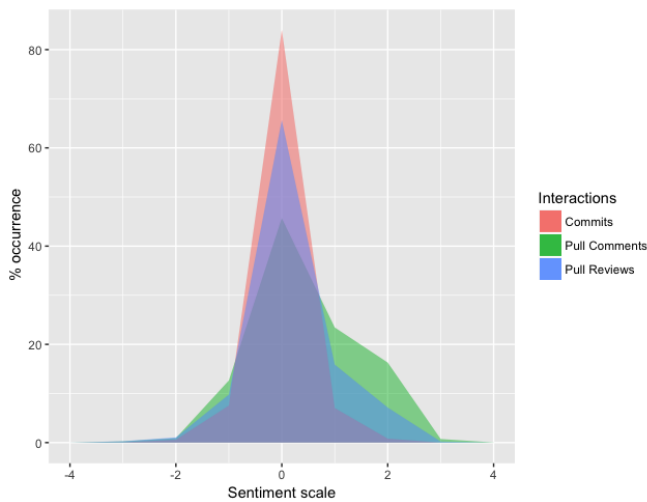


Figure 2: Sentiment distributions in different interactions.

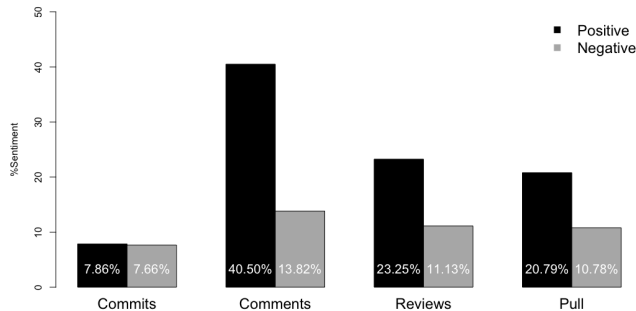


Figure 3: Sentiment by interaction.

and 7.66% negative). Figure 3 shows the percentage for all the interactions.

We noted that the most sentimental interaction is the pulls comments because this is where the users express their sentiments the most (54.32%). We expected this result, given the nature of the interaction once it represents a commentary from a developer on another developers patch of code. The commenter can agree or disagree with the change and can also request changes which sometimes starts a discussion. We believe this interaction can be a trigger for changing the humor of the developer submitting the code to integrate and we explore this hypothesis in the section 4.

## 4 Results

This section presents the results of the experiment each result is related to an RQ.

**Mood Variation** : To answer the RQ1 and RQ2, we analyzed the subjective well-being change that was previously present in the equation, for 1, 2, 4, 8 hours and one day, using a comment in a developer’s pull request as a reference; this way, we can evaluate whether or not the comment

Role	%
ANY	31.16%
COLLABORATOR	30.51%
CONTRIBUTOR	31.50%
MEMBER	32.06%
NONE	29.77%
OWNER	31.62%

Table 2: Relevant sentiment change One hour time window

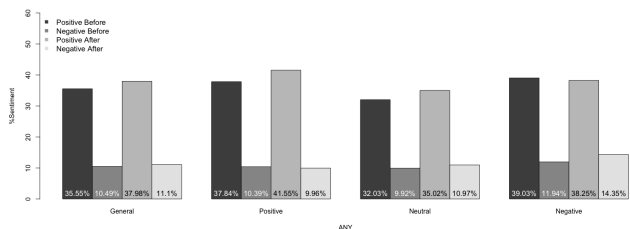


Figure 4: Sentiment variation before and after interaction with ANY developer, in the one-hour time window.

had an influence and how the influence will behave in time. Next, we discussed the results.

In a 1 hours time window, we noted 31.15% of significant mood change ( $abs(SC(d)) > 1$ ) in general. We also explored this sentiment change with the role of the commenter, but the role of the commenter does not change the influence significantly, where the smaller influence is from the role none (commenter no association with the repository) 29.77% and the highest is from member (commenter is a member of the organization that owns the repository) 32.06%. Table 2 shows the values for all the roles. We also analyzed how the sentiment changed based on the sentiment expressed by the commenter. This time, we analysed the positive and negative sentiments expressed before and after a comment. Figure 4 shows that receiving a positive comment the increased the percentage of positive interaction from 37.85% to 41.55% and receiving a negative comment increased the percentage of negative interaction from 11.94% to 14.35% ignoring the role of the commenter.

To answer **RQ3**, we studied how the influences behaved over time. We noted that as time passed, the notable possible influence reduced, as we show in Figure 5. The blue line shows the relevant mood variation ( $abs(SC(d)) > 1$ ) for each time window when receiving a neutral comment, the red shows the same when receiving negative comments, and the green line when receiving positive comments. We found that as time passed, the influence reduced in all the cases. The negative comments had the bigger influence in all analyzed time windows, with an average of 2,93 percentage points bigger than general.

To address **RQ4**, we explored the consequences of a negative comment on the first pull request of a developer.

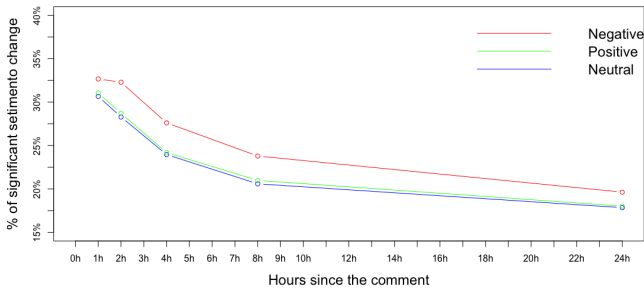


Figure 5: Relevant sentiment change by time since comment.

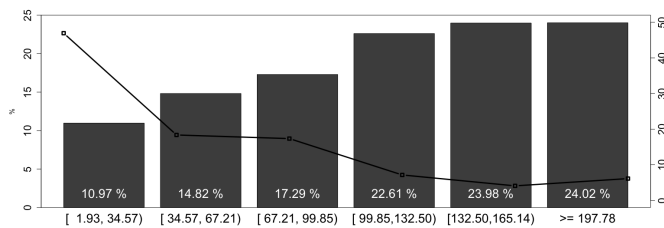


Figure 6: Percent of Once Contributor, with negative comment on the first Pull Request by KLoc

On average, 70% of the pull requests came from a contributor who would never contribute again to the repository (once-contributors). We related this to receiving a negative comment on the first contribution (pull request) and also to the project characteristics, KLoc, number of contributors, and stargazers. We found that as the project grew in KLoc(Thousand lines of code) the number on once-contributors with negative comments on the first pull request grew, from 10.97% on projects with less than 30.57 KLoc to 24.02% on projects bigger than 197.78 KLoc, we present the progression in Figure 6 in the bars, and the line shows the percent of projects in the respective KLoc range, we did not find a relationship between number of contributors or stargazers with the percentage of once-contributors.

**Research Questions:** In this paper, we addressed the following research questions:

**RQ1: Can a developer change the sentiment of another developer with a pull request comment?** The selected metric presented 31.16% of significant sentiment change when considering one hour before and after a comment.

**RQ2: Does the role of the commenter in the project change the intensity of influence he/she has on the sentiment of another developer?**

The role of the commenter had not had a significant impact on the sentiment influence, with a difference of only 1.83 percentage points between the most influencer when the commenter was the owner of the project and the least

influencer when the commenter has no relationship with the project.

**RQ3: How does the influence behave over time? Does the behavior change depending on the comment sentiment?**

In general, the influence reduces as the time pass, from 31.15% in one hour to 18.16% in one day. The behavior does not change, depending on the sentiment of the comment, but we noted the biggest influence from negative comments in all the analyzed time windows.

**RQ4: Do negative comments on the first Pull Request lead to quitting the project?**

Only 14.85% of the developers with a single Pull request received at least one negative comment on the pull request. On the other hand, we found a weak correlation with the size of the project in lines of code and the number of developer with only one contribution that received at least one negative comment, where the percentage grew from 10.97% on projects with less than 34.57k lines of code, to 24.02% on projects bigger than 197.78k lines of code.

## 5 Related Work

This section describes previous works regarding sentiment analysis on software development interactions and a short comparison with the current paper.

Robinson et al. [12] performed data analysis on open source projects looking to understand how behavior change can change developers sentiment, and they analyzed 2 points, behavior change and routine change and their relationship with sentiment change. They used a regression model to search for the relationship between developer sentiment change and developer behavior change. They hypothesized that routine change would change developer sentiment positively or negatively. They evaluated 124 GitHub projects also performing intra-project and multi-project analysis and their results shown that routine change had a positive impact on the developer sentiment when evaluation multi-project approach and negative sentiment change were related to behavior change.

Islam and Zibrán [8] performed an analysis of 50 projects with more than 490 thousand commits messages, searching for sentiment variations over the commits messages. They searched for a relationship between, weekday, day hour, commits message length, and task type related to the developer sentiment variation, they used hierarchical algorithm clustering to perform clustering and used a statistical approach to support their findings. They found relationships between the task type and the developer sentiment variation; bug fix commits have more positive sentiment than refactoring tasks. Also more significant commits message express more sentiment than small commit messages, weekdays and hours of that day did not show any significant relationship with developer sentiment. Souza and



Silva [14] studied the sentiment related to building status, evaluated 1,262 projects from GitHub, and more than 609k builds, they found that the commit message following broken builds has a week correlation with negative sentiment.

Sinha et al. [13] analyzed the sentiments of the developer in the commits messages; their research focuses on finding the sentiments variations only into commits and relate this with the day of the week and the size of the commit. They found that a low percentage of commits has sentiment expressed, and there are more negative than positive sentiment expressed (5% positive and 14% negative), they also found the worst day in sentiment level (higher volume of negative sentiment) on Tuesday.

None of the related works explored the sentiment contagion studying the influence of other developers in the developers' mood; also, few papers analyzed the sentiment over the Pull Request comments, which is the proper place to promote discussion related to code. Therefore, in this paper, we take the challenge of investigating how other developers' comments may influence the developers' mood.

## 6 Conclusions and Future Work

This paper analyzed the impact of feedback on developers' mood when submitting pull requests. Our results showed that developers feedback might influence another developers' mood; negative comments have bigger impacts on mood variation; and as project grows in lines of code, the bigger is the impact of negative comments on the first contribution, and it might result in not contributing again with the same repository. We believe our results will help project leaders and companies create conduct codes to guide developer feedback constructively.

In future works, we intend to explore deeper the consequences of politeness and impoliteness in the success of open source projects and communities' growth or decline and its relationship with the maintainer's sentiment expression. We also intend to use the relevance of developers in a community instead of the rule they have in the project they are commenting or contributing. We believe that developer relevance has a relationship with the impact it can cause, independent of the rule they have in the project they are contributing.

## Acknowledgments

This research was partially supported by CAPES, CNPq, FAPEMIG, and FIP-PUC Minas

## References

- [1] Comment author association — github developer guide. URL <https://developer.github.com/v4/reference/enum/commentauthorassociation/>.
- [2] Github glossary - user documentation. URL <https://help.github.com/articles/github-glossary/>.
- [3] Johan Bollen, Bruno Gonçalves, Guangchen Ruan, and Huina Mao. Happiness is assortative in online social networks. *CoRR*, abs/1103.0784, 2011.
- [4] Munmun De Choudhury and Scott Counts. Understanding affect in the workplace via social media. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work, CSCW '13*, pages 303–316, 2013. ISBN 978-1-4503-1331-5.
- [5] A. Fountaine and B. Sharif. Emotional awareness in software development: Theory and measurement. In *2017 IEEE/ACM 2nd International Workshop on Emotion Awareness in Software Engineering (SEmotion)*, pages 28–31, 2017.
- [6] Daniel Graziotin, Xiaofeng Wang, and Pekka Abrahamsson. Happy software developers solve problems better: psychological measurements in empirical software engineering. *PeerJ*, 2:e289, March 2014. ISSN 2167-8359.
- [7] Emitza Guzman and Bernd Bruegge. Towards emotional awareness in software development teams. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2013*, pages 671–674, 2013. ISBN 978-1-4503-2237-9.
- [8] M. R. Islam and M. F. Zibran. Towards understanding and exploiting developers' emotional variations in software engineering. In *2016 IEEE 14th International Conference on Software Engineering Research, Management and Applications (SERA)*, pages 185–192, 2016.
- [9] Mika Mäntylä, Bram Adams, Giuseppe Destefanis, Daniel Graziotin, and Marco Ortu. Mining valence, arousal, and dominance: Possibilities for detecting burnout and productivity? In *Proceedings of the 13th International Conference on Mining Software Repositories, MSR '16*, pages 247–258, 2016. ISBN 978-1-4503-4186-8.
- [10] Nicole Novielli, Fabio Calefato, and Filippo Lanubile. The challenges of sentiment detection in the social programmer ecosystem. In *Proc. of the 7th International Workshop on Social Software Engineering, SSE 2015*, pages 33–40, 2015. ISBN 978-1-4503-3818-9.
- [11] Filipe Nunes Ribeiro, Matheus Araújo, Pollyanna Gonçalves, Fabrício Benevenuto, and Marcos André Gonçalves. A benchmark comparison of state-of-the-practice sentiment analysis methods. *CoRR*, abs/1512.01818, 2015.
- [12] W. N. Robinson, T. Deng, and Z. Qi. Developer behavior and sentiment from data mining open source repositories. In *2016 49th Hawaii International Conference on System Sciences (HICSS)*, pages 3729–3738, 2016.
- [13] V. Sinha, A. Lazar, and B. Sharif. Analyzing developer sentiment in commit logs. In *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*, pages 520–523, May 2016. doi: 10.1109/MSR.2016.069.
- [14] R. Souza and B. Silva. Sentiment analysis of travis ci builds. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, pages 459–462, 2017.
- [15] Robert S. HuckmanBradley Staats. The hidden benefits of keeping teams intact, Aug 2014. URL <https://hbr.org/2013/12/the-hidden-benefits-of-keeping-teams-intact>.
- [16] Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. Sentiment in short strength detection informal text. *J. Am. Soc. Inf. Sci. Technol.*, 61(12):2544–2558, December 2010. ISSN 1532-2882.
- [17] Valley Voices. Developers don't care how much you pay them, Feb 2017. URL <https://www.forbes.com/sites/valleyvoices/2017/02/09/developers-dont-care-how-much-you-pay-them/>.

# Do Scale Type Techniques Identify Problems that Affect User eXperience?

## User Experience Evaluation of a Mobile Application

Leonardo Marques, Walter Nakamura, Natasha Valentim, Luis Rivero and Tayana Conte

Institute of Computing  
Federal University of Amazonas, UFAM  
Manaus, Brazil

{lcm, walter, natashavalentim, luisrivero, tayana}@icomp.ufam.edu.br

**Abstract**—User experience (UX) is a quality aspect of an application that considers the emotions evoked by the system. There are several types of UX evaluation methods, such as scales, interviews, user monitoring, among others methods. However, there is still not enough information regarding if and in which contexts these methods are more suitable. This paper focuses on finding more information about the feasibility of applying scale-type methods for the UX evaluation in interactive systems. Thus, we carried out a UX evaluation on the Edmodo educational mobile application employing two scale-type methods: AttrakDiff and Hedonic Utility Scale. These methods were chosen after a selection process applying inclusion and exclusion criteria. The results indicate that it is possible to evaluate the quality of an application at a low cost. However, it is necessary to adapt these methods to provide a more complete report of the UX, allowing users to subjectively report their experiences and, consequently, identify the issues that affected the UX.

**Keywords**—User Experience; UX; AttrakDiff; Hedonic Utility Scale; Edmodo; comparative study;

### I. INTRODUCTION

Technological evolution has enabled the emergence of new interaction paradigms, new technologies and new types of software [1]. This evolution has changed the way users interact and perceive interactive products. Hassenzahl and Tractinsky [2] have identified that pragmatic factors such as functionality and usability are not enough to satisfy the desire of customers seeking innovative products and products that meet their expectations. According to Bargas-Avila and Hornbæk [3], usability is too focused on the efficiency and the accomplishment of tasks, with a need for more comprehensive notions of quality. As a result, User eXperience has emerged as a concept of quality of use that, besides involving the aspects mentioned above, covers hedonic qualities or emotional aspects resulting from the use of an application [3].

Several UX evaluation methods have been proposed in the literature. Vermeeren et al. [4] carried out a survey on the UX evaluation methods used in the academy and in industry and identified 96 methods. Despite the large number of methods, there is a need to verify their applicability with regards to the resources, required skills to apply them, and the perceived ease of use of these methods [3].

This paper presents a comparison between two scale-type methods and the indications of their feasibility in the context of evaluating a mobile educational application called Edmodo. The UX evaluation of Edmodo was suggested by a University that was analyzing the possibility of using educational technologies in the classroom. Considering that UX is one of the aspects that can impact the acceptance of adopting teaching and learning technologies [5], we decided to carry out a UX evaluation to verify if Edmodo evoked a positive UX in its mobile version.

In order to select the UX evaluation methods, we carried out a selection process by means of inclusion and exclusion criteria, which resulted in two scale-type techniques: AttrakDiff [7][16] and the reduced form of the Hedonic Utility Scale (HED/UT) [8]. We compared these methods in terms of usefulness and ease of use, raising information to verify their feasibility for assessing mobile applications. The results allowed us to identify the positive and negative aspects of the methods employed, besides presenting opportunities for improvement in the application.

The remainder of this paper is organized as follows. Section II presents concepts related to UX. Section III describes the use of UX evaluation methods presenting the evaluated application, the selection process and the study carried out. Section IV presents the results obtained in this study and the discussions. Finally, Section V concludes the paper.

### II. BACKGROUND

ISO 9241-210 [9] defines UX as "*person's perceptions and responses that result from the use and/or anticipated use of a product, system or service.*" Hassenzahl and Tractinsky [2] suggest that UX is a consequence of the user's internal state (predispositions, expectations, needs, etc.), the characteristics of the projected system (complexity, usability) and the context where the interaction occurs (organization, users, among others).

UX evaluation plays an important role in the development of interactive applications. Through the UX evaluation it is possible to identify how users apply, perceive and learn these applications, allowing the applications to evolve and adapt to user's expectations [10]. Thus, it is possible to identify potential problems in the use of applications and their causes, as well as to obtain suggestions for their improvement. There are several methods for evaluating UX, which can be categorized into three

types [11]: (i) written report, which consists of the evaluation of the experience through scales, forms and questionnaires; (ii) oral report, in which participants report their experiences through interviews or verbal methods; and (iii) observation/monitoring, in which participants are observed or use sensors to monitor their responses while perform activities related to the use of the system.

Written reporting methods, such as scale-type methods, have been widely used due to their low cost, ease of use, and the possibility of collecting data from both positive and negative experiences [12]. Examples of this type of method are questionnaires that use scales with semantic differentials, i.e., scales composed of pairs of words that are opposing adjectives, such as "simple/complicated" and "pleasant/unpleasant". However, there is little information regarding the positive and negative aspects of these methods. Thus, we selected methods within this category to: (a) analyze what type of outcomes are produced by these methods; (b) analyze if scales are sufficient to identify problems that have affected the UX; and (c) verify the feasibility of these methods to evaluate a mobile application.

### III. APPLYING USER EXPERIENCE ASSESSMENT METHODS

#### A. Goals and Metrics

Table 1 presents the goal of the study according to the GQM (Goal Question Metric) paradigm, which allows to define and evaluate objectives in the stage of goal setting [13].

TABLE 1. GOAL OF THE STUDY ACCORDING TO THE GQM PARADIGM.

<b>Analyze</b>	The AttrakDiff and Hedonic Utility Scale UX evaluation methods
<b>For the purpose of</b>	characterizing.
<b>With respect to</b>	participants' perception in terms of Usefulness, Ease of Use and intention to use each method.
<b>From the point of view</b>	of users and researchers.
<b>In the Context of</b>	a UX evaluation of a real application in a Computer Science Introduction class.

The UX problems were verified through the number of tasks performed successfully in Edmodo as well as through difficulties experienced by the participants. The utility, ease of use and intention to use the methods were obtained through the TAM3 (Technology Acceptance Model) [14].

#### B. UX Method Selection Process

We applied the selection process on the UX evaluation methods list identified by Rivero and Conte [11], seeking to identify UX evaluation methods for mobile applications. This selection process consisted of two refinement steps.

The first refinement was based on the criteria described in Table 2. For each criterion, there is a description of what was considered for the exclusion. First, we considered only the methods available for consultation (EC2). After, we applied the other exclusion criteria, resulting in a set of 18 UX evaluation methods. The detailed specification of the criteria used for exclusion can be found in the technical report [6]

TABLE 2. CRITERIA FOR FILTERING THE UX EVALUATION METHODS.

Criteria	Description
<b>EC1 (Type of method)</b>	Methods characterized only as tools.
<b>EC2 (Availability)</b>	Methods not available for free or unavailable.
<b>EC3 (Data source)</b>	Methods whose data sources are not provided by users.
<b>EC4 (Location)</b>	Methods whose application is not possible in controlled environments.
<b>EC5 (Type of Assessed Product)</b>	Methods that cannot be applied to mobile app evaluation.
<b>EC6 (Type of Assessed Artifact)</b>	Methods whose evaluated artifact are not functional prototypes or final applications.
<b>EC7 (Assessed Period of Experience)</b>	Methods whose UX evaluation occurs before or during the use of the system.

Among the 18 selected methods, we queried each method in the AllAboutUX<sup>1</sup> Website in order to carry out the second refinement. This website presents, for each method, their characteristics and what is needed to use them. Some methods, such as the "Group-based expert walkthrough" [15] requires UX experts to be applied. Given that the UX of Edmodo would be assessed by users and not experts, we discarded methods that require experts to evaluate. We also discarded those developed for specific contexts, such as the "Attrak-Work Questionnaire" method [16], which was developed for the context of news and journalism. At the end of the second refinement, we selected three methods: AttrakDiff [7][16], Hedonic Utility Scale (HED/UT) [8] and Self-Assessment Manikin (SAM) [17].

We carried out a pilot study using the three selected methods to verify the data collected by each method and the outcomes that each one generated. The results indicated that SAM is a method that evaluates only the hedonic aspects of the experience, thus it would be unfair to compare it with AttrakDiff or HED/UT, which assess both pragmatic and hedonic aspects of the user experience. Thus, at the end of this process, we selected the AttrakDiff and HED/UT methods.

The AttrakDiff method consists of 28 pairs of words that evaluate pragmatic, hedonic and attractiveness aspects, whereas the HED/UT method, in its reduced form, consists of 12 pairs of words that evaluate the pragmatic and hedonic aspects. Regarding HED/UT, we applied its reduced version because the results obtained in [8] showed that its 12 pairs of words are sufficient to verify the quality, making it viable to measure the UX. The pairs of words of both methods are organized on a seven-point scale, in which the participant performs the UX evaluation by marking the closest point to the adjective that best characterizes his experience of use (see Figure 1).

#### C. Evaluated Mobile Application

We decided to evaluate the Edmodo application due to the university's suggestion to address its feasibility, while no studies related to the UX evaluation of this application were found. Edmodo is a Learning Management System (LMS) created in 2008 to manage learning activities. Its popularity among educational institutions has risen and it has more than 80 million users and more than 50 million downloads in its mobile version.

The evaluation of LMSs is a critical issue, as it can affect students' performance, making them spend more time trying to

<sup>1</sup> <http://www.allaboutux.org/all-methods>

understand how to use these environments than learning the educational content [18][19].

A	Human						X		Technical
	Isolating								Connective
	Pleasant								Unpleasant
B	Useful								Useless
	Impractical					X			Practical
	Necessary								Unnecessary

Figure 1. (A) AttrakDiff Questionnaire, (B) HED/UT Questionnaire.

#### D. Execution

We conducted the study with 38 volunteer students from the Federal University of Amazonas, who participated in an Introduction to Computing class. This class was partially online and used an LMS in the teaching/learning process.

For the execution of the study, we accommodated the participants in a laboratory and divided them into two groups, balanced according to their previous experience with the Edmodo app. One group used the AttrakDiff and the other group used HED/UT (details in the technical report [6]).

Initially, participants received a Consent Form, and then, they received: (a) a form for reporting their difficulties when using Edmodo, (b) the AttrakDiff and HED/UT, (c) the TAM3 questionnaire, and (d) a questionnaire with open questions related to the use of the UX evaluation methods.

Before performing the UX evaluation, we made a brief explanation about Edmodo and its functionalities. After, we asked the participants to download the application. All the participants used Android or iOS devices, in which a prior check was made to ensure that the execution flow of the activities in the application would not be changed.

After explaining the application, the participants received a schedule of activities. According to Nielsen [20], the basic rule for selecting a set of activities is that it must be chosen in such a way that they are as representative as possible. Thus, this script consisted of the following activities: (i) create a student account in Edmodo and enter the group through the access code, made available by the moderators during the study, (ii) update the profile photo, (iii) access the group library and download a file for reading, (iv) answer an activity containing two questions related to the text of the downloaded file and (v) attach a file and send it to the teacher.

At the end of the activities, the participants received a form about the difficulties encountered during the interaction with Edmodo in order to better understand the quantitative results of the UX evaluation. Then, each group received a UX evaluation method, in which each participant was instructed on how to perform the evaluation. The form about the difficulties and the number of tasks performed successfully in Edmodo was verified in order to identify the UX problems.

As the participants finished evaluating the application, they received the questionnaire based on the TAM3. The TAM3 is a model used to verify the acceptance of a technology that, among other dimensions, considers utility, ease of use and intention to

use. Participants were instructed to use this questionnaire to evaluate the UX evaluation method they used. We also attached to the TAM3-based questionnaire, a questionnaire containing five open questions related to their experiences with the UX evaluation method in order to better understand the aspects that made each method easy or difficult to use.

#### IV. RESULTS AND DISCUSSIONS

In order to make the comparison possible, we tried to equate both methods, since they evaluate the UX in different dimensions. AttrakDiff evaluates UX in four dimensions: Pragmatic Quality (PQ), Hedonic Quality/Stimulus (HQ/S), Hedonic Quality/Identity (HQ/I) and Attractiveness (ATT), while HED/UT evaluates two dimensions: Utility, which corresponds to the Pragmatic dimension, and Hedonic.

Given that the pragmatic dimensions of both methods are equivalent (evaluate the same experiences), we searched for definitions that qualified what was evaluated in the hedonic dimension of each method to verify if they were equivalent. According to Voss et al. [21], the hedonic dimension of HED/UT is the result of sensations derived from the experience of using products. According to Väättäjä et al. [16], the HQ/S dimension of AttrakDiff is related to personal development, that is, curiosity, personal growth, skill development, and the proliferation of knowledge, i.e., feelings and sensations caused by the use of the application. Thus, the AttrakDiff HQ/S was considered equivalent to the HED/UT hedonic dimension. Thus, we considered only the Pragmatic and QH/E dimensions of AttrakDiff, and the Utility and Hedonic dimensions of HED/UT.

To compare the methods, we organized the data by factors. Factor 1 relates to the pragmatic dimension, while Factor 2 represents the hedonic dimension. The following subsections describe the results of the UX evaluation of Edmodo and the results regarding the methods used to evaluate the its UX.

##### A. Results and Analysis of the UX Evaluation on Edmodo

Table 3 presents the score for each Factor per method. According to Distefano et al. [22], when Factors are not defined by the same number of items, which is the case of the Factors of both methods, it is recommended to calculate the mean, making it possible to compare them with each other. The mean is also recommended by Sullivan and Artino Jr [23] when measuring less concrete concepts, such as satisfaction, where a single research item is not likely to capture the assessed concept completely. In order to obtain the Factor's scores, first we calculated the mean of each participant's scores per Factor. This mean was based on the score given by each participant in each item of the method's dimension. Then, we obtained the score of each Factor through the mean of the scores of each previously calculated participant.

Regarding Factor 1, the scores indicate that participants who evaluated UX using the HED/UT evaluated more positively the experience regarding the ease of use of Edmodo compared to those using AttrakDiff. Given that the scale ranges from 1 to 7, the UX would be positive if the scores were greater than or equal to 5. Thus, the results indicated that the participants considered that Edmodo provides a positive UX, since the lowest score was close to 5. Regarding Factor 2, the group that used AttrakDiff

felt it neutral, i.e., Edmodo was not considered bad, but it did not stimulate users so much. It indicates that Edmodo needs to implement improvements to stimulate and captivate users. The group that used HED/UT considered Edmodo's UX positive.

TABLE 3. FACTORS' SCORE ASSESSED BY THE ATTRAKDIFF AND HED/UT.

Factor 1 (Pragmatic)			Factor 2 (Hedonic)	
Method	AttrakDiff	HED/UT	AttrakDiff	HED/UT
Score	4,8	5,9	4,4	5,5

The responses provided by the participants in the form about the difficulties faced when using Edmodo reflected these scores. From a total of 38 participants, 18 reported having had difficulty finding the group's Library in Edmodo. This is a problem that affects the use of the application, being reflected in the UX evaluation, where most participants considered that Edmodo is very technical (AttrakDiff) and impractical (HED/UT). Other problems were also pointed out, such as the mix of words in Portuguese and English on the interface. These problems were only possible to be identified through the form on the difficulties faced in Edmodo, because the scales do not allow to identify the problems that affected the UX in this level of specification.

### B. Results Regarding the UX Evaluation Methods

In order to verify the participants' perception regarding the usefulness, ease of use and intention to use, we applied the TAM3-based questionnaire. We used the median as a statistically significant measure for ordinal scales [24] with the same number of items. Table 4 shows the description of the items that compose each of the dimensions evaluated by TAM3.

TABLE 4. TAM3-BASED QUESTIONNAIRE ITEMS.

Description of the items on "Perceived Usefulness" (PU)	
PU1	Using the method improves my performance by reporting my experience with the application.
PU2	Using the method improves my productivity by reporting my experience with the application.
PU3	Using the method allows me to fully report the aspects of my experience.
PU4	I find the method useful for reporting my experience with the application.
Description of the items on "Perceived Ease of Use" (PEOU)	
PEOU1	The method was clear and easy to understand.
PEOU2	Using the method did not require much mental effort.
PEOU3	I think the method is easy to use.
PEOU4	I find it easy to report my experience with the application using the method.
Description of the items on "Intention to Use" (IU)	
IU1	Assuming I have access to the method, I plan to use it to evaluate my experience with an application.
IU2	Given that I have access to the method, I predict that I would use it to evaluate my experience with an application.
IU3	I plan to use the method to evaluate my experience with an app next month.

Table 5 shows the median values for each TAM3 item. Based on these data, we verified the items that had some variation, since these indicate which of the methods was better.

The items that had variations were PU2, PEOU1, PEOU4 and IU2. These items show that HED/UT had a better perception regarding AttrakDiff by the participants, indicating that short methods improve productivity (PU2), methods that use less formal terms are easier to use (PEOU1 and PEOU4), and these aspects influence intention to use (IU2).

TABLE 5. MEDIAN OF EACH ITEM PER METHOD.

	AttrakDiff	HED/UT
<b>Perceived Usefulness (PU)</b>		
PU1	5	5
PU2	5	6
PU3	5	5
PU4	5 and 6	6
<b>Perceived Ease of Use (PEOU)</b>		
PEOU1	5	6
PEOU2	6	6
PEOU3	6	6
PEOU4	5 and 6	7
<b>Intention to Use (IU)</b>		
IU1	5	5
IU2	5	6
IU3	4	4

The results of the questionnaire with open questions about the UX evaluation methods indicated some opportunities for improving them. In both methods, some participants reported that they were not able to express their experiences of use only through the scales. The methods do not allow them to write the problem that affected the UX, or in which part of the application they consider that there should be improvements, which indicates a limitation of the methods evaluated. One possibility of improvement would be, for example, the addition of a field so that the participant can report the difficulties that were not possible to be described only with the scales.

*"I cannot describe the experience I had"* – P08 (HED/UT).

*"Not being able to express [the experience] in a more justified way"* – P05 (AttrakDiff).

Regarding AttrakDiff, the participants reported the difficulty in understanding some terms, considering them formal (see quote from P12). This may impact the UX report, since the participant can point out in any way when evaluating. The results of Table 5 showed that HED/UT had a better perception of users than AttrakDiff, because it used less formal terms. This could be an indication that terms that are used daily by users must be used instead.

*"[There were] some formal words that I did not know what they meant"* – P12

Regarding HED/UT, some participants indicated that the available options were insufficient to evaluate the experience satisfactorily (see quote from P02). Others reported that just having to point out an "X" makes the method simple and easy (see quote from P03).

*"Only more options to point out"* – P02

*"It's simple and easy"* – P03

Thus, there are some gaps with regards to the methods used in the study. Based on the reports, scale methods should be complemented with questionnaires or a comment field, allowing the evaluator to describe the difficulties faced and what aspects were enjoyable when using the application. The lack of these fields makes it difficult to implement improvements in the evaluated application, since it is possible only to know that the application needs improvements, but not which problems users have faced. In addition, it is recommended to use less formal

terms, in order to make the method more comprehensible, as shown in Table 5.

## V. CONCLUSIONS

Performing a UX assessment is important for gaining end-user insight about a particular application. In this paper, we showed that the use of scale-type methods allows to perform the UX evaluation quickly, not making the evaluation process tiring for the user. In addition, few resources are required to evaluate the quality of an application, reducing the cost of evaluation, which makes the use of these methods attractive.

However, this type of method has the limitation of not collecting qualitative data of the evaluation, i.e. the subjective information that describes the difficulties encountered by the users and that could point out the problems of the evaluated application. This may indicate that only using scale-type methods may not produce such detailed results, making it difficult to precisely identify which aspects affected the UX during the use of the application. For a holistic assessment, complementation of the scales with the open questions was positive, making it possible to obtain the positive aspects and the aspects that need to be improved in the application.

The results of the UX evaluation showed that Edmodo has a positive UX and that it can be used by teachers as a tool to support the teaching/learning process. However, some improvements are needed, such as facilitating the access to the library and fixing the mix of languages on the interface. By doing this, the application can have a greater acceptance and become easier and more enjoyable to use, important aspects to have a competitive advantage over other applications.

We hope that the results from this study contribute to the development of UX evaluation techniques that make use of the positive aspects found in the scale-type methods and that provide the negative aspects, such as the lack of a field where the participants can detail their experiences, to obtain a more complete and detailed UX report. In addition, we hope that suggestions for improvements can contribute to the improvement of the Edmodo application.

## ACKNOWLEDGMENT

We would like to thank the financial support granted by UFAM, CNPq through processes numbers 423149/2016-4 and 311494/2017-0, and CAPES through process number 175956/2013.

## REFERENCES

[1] C. Rusu, V. Rusu, S. Roncagliolo and C. González, "Usability and User Experience: What should we care about?", *International Journal of Information Technologies and Systems Approach*, v.8, n.2, p.1-12, 2015.

[2] M. Hassenzahl and N. Tractinsky, "User experience-a research agenda", *Behaviour & information technology*, v.25(2), p.91-97, 2006.

[3] J. A. Bargas-avila and K. Hornbæk, "Old wine in new bottles or novel challenges: a critical analysis of empirical studies of user experience", In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, p. 2689-2698, 2011.

[4] A. P. Vermeeren, E. L. C. Law, V. Roto, M. Obrist, J. Hoonhout and K. Väänänen-Vainio-Mattila, "User experience evaluation methods: current

state and development needs", In *Proc. of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, p. 521-530, 2010.

[5] N. Harrati, I. Bouchrika, A. Tari and A. Ladjailla, "Exploring user satisfaction for e-learning systems via usage-based metrics and system usability scale analysis", *Computers in Human Behavior*, 61, 463-471, 2016.

[6] L. Marques, W. T. Nakamura, L. Rivero, N. Valentim and T. Conte, "TR-USES-2017-0012. Supporting Material for Evaluating User Experience in a Mobile Education Application". *Technical Report of Usability and Software Engineering Group (USES)*, 2017 Available in <http://uses.icomp.ufam.edu.br/relatorios-tecnicos>.

[7] M. Hassenzahl, M. Burmester and F. Koller, "AttrakDiff: Ein Fragebogen zur Messung wahrgenommener hedonischer und pragmatischer Qualität", In *Mensch & Computer*, p. 187-196, 2003.

[8] H. Van Der Heijden and L. S. Sørensen, "Measuring attitudes towards mobile information services: An empirical validation of the HED/UT scale", In *Proceedings of the European Conference on Information Systems (ECIS)*, p. 765-777, 2003.

[9] ISO DIS 9241-210:2010. "Ergonomics of human system interaction -Part 210: Human-centred design for interactive systems". *International Standardization Organization (ISO)*. Switzerland.

[10] A. Moreno, A. Seffah, R. Capilla and M. Sanchez-Segura, "HCI practices for building usable software", *IEEE Computer*, (4), p. 100-102, 2013.

[11] L. Rivero and T. Conte, "A systematic mapping study on research contributions on UX evaluation technologies", In *Proc. of the 16th Brazilian Symposium on Human Factors in Computing Systems (IHC)*, Joinville, Brazil, 2017.

[12] P. Desmet, "Measuring emotion: Development and application of an instrument to measure emotional responses to products", In *Funology*, Springer Netherlands, p. 111-123, 2005.

[13] V. Basili, G. Caldiera and H. Rombach, "Goal question metric paradigma", *Encyclopedia of Software Engineering (1)*, John Wiley & Sons, New York, p. 528-532, 1994.

[14] V. Venkatesh and H. Bala, "Technology acceptance model 3 and a research agenda on interventions". *Decision sciences*, 39(2), 273-315, 2008.

[15] A. Foelstad, "Group-based Expert Walkthrough", In *R<sup>3</sup>UEMs: Review, Report and Refine Usability Evaluation Methods*, edited by Scapin, D. and Law, E., p. 58-60, 2007.

[16] H. Väättäjä, T. Koponen and V. Roto, "Developing practical tools for user experience evaluation: a case from mobile news journalism", In *European Conference on Cognitive Ergonomics: Designing beyond the Product-Understanding Activity and User Experience in Ubiquitous Environments*, p. 23, 2009.

[17] M. M. Bradley and P. J. Lang, "Measuring emotion: the self-assessment manikin and the semantic differential". *Journal of behavior therapy and experimental psychiatry*, 25(1), p. 49-59, 1994.

[18] R. Lanzilotti, C. Ardito and M. F. Costabile, "eLSE methodology: A systematic approach to the eLearning systems evaluation", In *Educational Technology & Society*, v. 9, n. 24, p. 42-53, 2006.

[19] H. B. Santoso, M. Schrepp, R. Isal, A. Y. Utomo and B. Priyogi, "Measuring User Experience of the Student-Centered e-Learning Environment", In *Journal of Educators Online*, v. 13, n. 1, p. 58-79, 2016.

[20] J. Nielsen, "Usability Engineering", In *Morgan Kaufmann Publishers Inc*. San Francisco, CA, USA, 1993.

[21] K. E. Voss, E. R. Spangenberg and B. Grohmann, "Measuring the hedonic and utilitarian dimensions of consumer attitude", In *Journal of marketing research*, v. 40 (3), p. 310-320, 2003.

[22] C. Distefano, M. Zhu and D. Mindrila, "Understanding and using factor scores: Considerations for the applied researcher", In *Practical Assessment, Research & Evaluation*, v. 14 (20), p. 1-11, 2009.

[23] G. M. Sullivan and A. R. Artino Jr, "Analyzing and interpreting data from Likert-type scales", In *Journal of Graduate Medical Education*, v. 5, n. 4, p. 541-542, 2013.

[24] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell and A. Wesslén, "Experimentation in software engineering", *Springer Science & Business Media*, 1st edition, 2012.

# A Systematic Approach for Developing Cyber Physical Systems

Xudong He

Florida International University, Miami, USA

Zhijiang Dong

Middle Tennessee State University, Murfreesboro, USA

Yujian Fu

Alabama A & M University, Huntsville, USA

**Abstract**— Cyber physical systems (CPSs) are pervasive in our daily life from mobile phones to auto driving cars. CPSs are inherently complex due to their sophisticated behaviors and thus difficult to build. In this paper, we propose a systematic approach to develop CPSs with quality assurance throughout the development process. A CPS is abstracted and partitioned into a set of independent executing agents, where each agent is further refined into a set of behaviors. Each behavior is modeled with a high level Petri net, called behavior net. The overall behavior of an agent is modeled by an agent through composing individual behavior nets. Finally, the overall system behavior is modeled by a system net through integrating individual agent nets incrementally. Simulation and model checking can be performed on individual behavior nets, agent nets, and the final system net. The resulting system net is systematically mapped to behavior programs in Java, which are enhanced and extended with domain specific functionality. A set of property patterns based on behavior program is developed, which are used to generate runtime monitors to check behavior program executions. We demonstrate our approach using a multi-car parking system.

**Keywords** - cyber physical systems; behavior programming; high level Petri nets; simulation; model checking, runtime verification

## I. INTRODUCTION

Cyber physical systems (CPSs) are pervasive in our daily life and need to be extremely reliable since they are often safety critical. CPSs consisting of computation and physical processes are inherently complex and demonstrate many sophisticated behaviors including synchronous, asynchronous, distributed, real-time, discrete, and continuous [1]. In [2], several major design challenges of CPSs were discussed, including concurrency and timing, which are intrinsic and critical in CPSs but are not adequately addressed in current computing abstractions. While fundamental new technologies are needed to develop CPSs, improving and integrating existing technologies including software engineering processes, design patterns, formal verification, and simulation provides a potential solution [2].

In [3], we provided a concrete framework to realize the ideas in [2], where a model driven approach from high level Petri nets to Java programs was presented. Essential CPS design issues including concurrency and timing are modeled using high level Petri nets and analyzed through model checking and simulation. Assumed environment constraints from hardware devices are checked during implementation and runtime verification. The overall framework is shown in Fig. 1. An agent oriented modeling approach is used to capture CPSs at a high abstraction level where meaningful computational components and physical processes with independent behaviors are viewed as agents and modeled using individual high level Petri nets. An aspect oriented approach is used to incrementally integrate

system components represented using individual agent nets into a complete system net. Agent nets and the system nets are

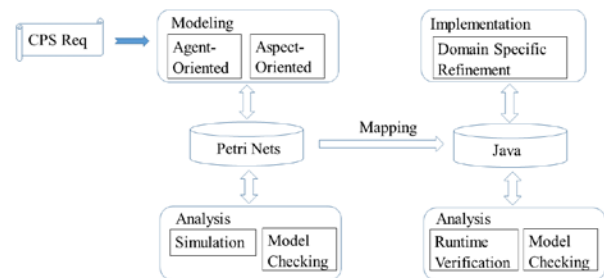


Fig. 1 – A framework for developing cyber physical systems

analyzed through simulation as well as model checking. The above modeling and analysis techniques are supported by tool chain PIPE+ [4] and SPIN [5]. A systematic translation approach has been developed, where a set of translation rules is used to map the individual agent nets into corresponding Java threads to form the general program structure. A complete Java program is obtained by combining the translated general program structure with domain specific program refinements. The additional refinements are necessary to realize CPSs, especially domain dependent physical devices. Bounded symbolic model checking and runtime-time verification are performed to ensure model level properties and additional properties are not violated in the implementation. The model level analysis and implementation level analysis are complementary. At model level, both safety and liveness properties can be checked to detect potential errors in the requirements with environmental assumptions such as the hardware devices working properly. At the implementation level, safety properties can be checked through bounded symbolic model checking and monitoring the actual behavior of hardware devices.

In this paper, we enhance the above framework with an additional behavior-oriented modeling approach that complements the agent-oriented modeling approach. While the agent-oriented approach provides a higher level system decomposition driven by concurrency, in which physical devices and computational processes are abstracted and modeled as agents; the behavior-oriented approach offers a finer system decomposition driven by unique non-deterministic behaviors within each physical device or computation process. Behaviors provide a more intuitive, natural, and concrete way to incrementally understand and develop CPSs. This systematic and multi-level incremental approach helps us to better understand and develop CPSs. A new set of runtime monitoring property patterns based on behavior programming are developed to ensure the dependability of the implementation.

Our new contributions include: (1) a systematic approach for modeling and analyzing CPSs, (2) a new behavior-oriented approach to incrementally model and analyze CPSs, (3) a pattern based translation method for generating behavior programs from behavior nets, and (4) a set of behavior based runtime monitoring property patterns. Our systematic approach is demonstrated through a multi robotic car parking system.

## II. CYBER PHYSICAL SYSTEM MODELING

To effectively model and analyze the complex behaviors of CPSs, many modeling techniques have been proposed and adapted in recent years including formal methods such as hybrid automata [6] and special graphical modeling languages such as actor-oriented MoC [7]. High level Petri nets [8] are well suited to model the complex behaviors of CPSs, especially combined with well-established software engineering approaches such as agent-oriented approach and aspect-oriented approach [3]. However most existing techniques only provide very general guidelines and lack fine grained rules. Behavior based modeling [9] provides an intuitive, natural, and concrete way to incrementally understand and develop CPSs. In the following sections, we describe a systematic approach in modeling and analyzing CPSs, which consists of three levels – a behavior-oriented approach for modeling the internal behaviors of an agent; an agent-oriented approach to model the components of a system, and an aspect-oriented approach to synthesize the whole system. We demonstrate our approach using a multi robotic car parking system.

### A. Modeling Individual Behaviors

In behavior-oriented modeling, the unique behaviors of a physical device (sensors and actuators) or a computation process are identified and abstracted from the requirement specifications and are modeled with individual high level Petri nets called behavior nets that interact with external environments. Specifically, we provide the following general and simple design pattern of a behavior net shown in Fig. 2:

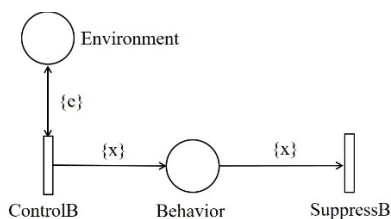


Fig. 2 – A Behavior Net Pattern

Where place *Behavior* models a behavior based on a uniquely identified behavior, which can be further refined by replacing the place with a more detailed net if needed. The type of place is a power set of a Cartesian product to capture multiple instances of behaviors of different objects, where each object has a unique identifier and other fields to capture important information. Place *Environment* models the external environment that can be detected by an object. Transition *ControlB* defines a condition to start the behavior and transition *SuppressB* models the end of the behavior. An additional incoming arc to transition *ControlB* will be created to indicate the selection of the behavior when the behavior net is integrated into an agent net and an outgoing arc from transition *SuppressB* will be created to integrate the behavior net.

We demonstrate our behavior-oriented approach in modeling a multi robotic car parking system. Each robotic car has two motors, two color sensors, and two IR (infrared obstacle) sensors. The color sensors are mounted on both front sides of a robotic car and are used to detect driving lane, two garage entrances, one exit, and four parking lots (all marked with unique colors). The IR sensors are mounted at the left front side (for left turning only) and the front of a robotic car to detect obstacle such as another robotic car or garage wall. Each robotic car has the following unique scenarios: (1) detecting an entrance using color sensors, (2) detecting the exit using color sensors, (3) searching for lane using color sensors, (4) detecting the lane using color sensors, (5) detecting obstacles for collision avoidance using IR sensors, (6) detecting a vacant parking lot using color sensors and IR sensors, (7) entering a parking lot using IR sensors, (8) leaving a parking lot using IR sensors, and (9) exiting the garage. Some of the above scenarios can be combined to form a more complex scenario such as searching and detecting lane, and some scenario such as detecting an entrance can be split into two specific scenarios – detecting entrance one and detecting entrance two. A screenshot of the behavior net search for lane (3) created in PIPE+ is shown in Fig.3. Since there is only one lane, place *Lane* holds only one token modeling the lane. Place *SearchLane* is a power set of tokens that model individual cars (4 cars in this system). Each car has a structured type of 3 string fields, the 1st field denotes car identifier, the 2nd field models a communication socket (not used in the model), and the 3rd field records a car status that is used to keep track behavior history and to select follower up behaviors.

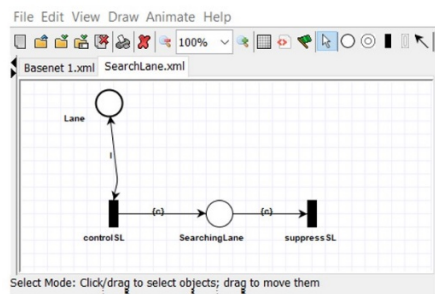


Fig.3 – Behavior Net of Searching Lane

### B. Modeling Individual Components

A high level Petri net can be used to capture the structure and the behavior of a physical or computation process. Petri nets naturally support synchronous, asynchronous, and distributed control and data flows. High level Petri nets are capable to model virtual time through time stamps associated with tokens and transition constraints representing delays and durations. Continuous behaviors of physical devices can be abstracted and discretized using real typed places and the associated transitions, and can be further refined during implementation.

Each type of physical devices (sensors and actuators) or computation processes is modeled with an agent net that has its own independent reactive and/or proactive behavior interacting with the external environment. Based on the behavior-oriented modeling, an agent net is obtained by integrating a set of



remarkably simple behavior nets through a place *Arbitrator*, which is used to control the selection of individual behaviors within an agent. The complete agent net of a single car after integrating all 12 behavior nets (the four parking lot behaviors are separately modeled) is shown in Fig. 4, which contains 22 places, 26 transitions, and over 60 arcs (many are bidirectional).

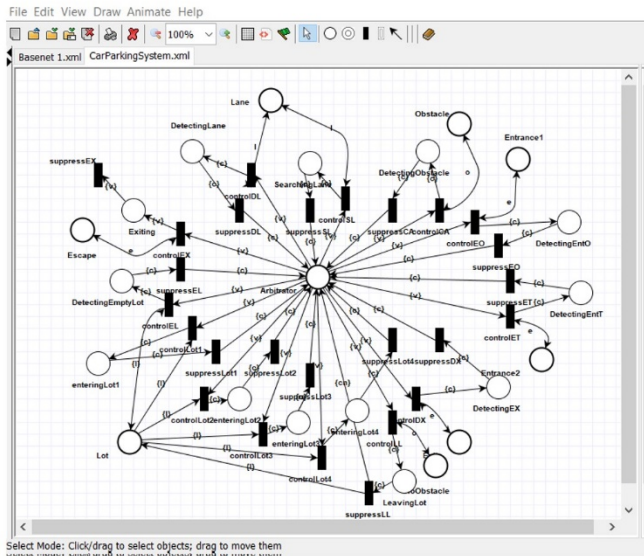


Fig. 4 – The Complete Agent Net of A single Car

### C. Modeling the Whole System

The overall system net is obtained by integrating individual agent nets that shows the interaction, communication, and cooperation among different agents. Synchronized activities are modeled through new joint transitions with modified constraints, and asynchronous activities are modeled through connecting a place in one agent net to a transition in another agent net. An aspect oriented approach [8] is used to build a complex model incrementally through weaving individual Petri nets representing agents capturing physical devices and computation processes. This aspect oriented approach further supports system adaptation and evolution, and facilitates compositional analysis. In this multi robotic car system, all the cars have the same behaviors and they do not communicate with each other. Thus the overall system net has the same structure as that of a single car. However multiple tokens with unique identifiers representing different cars are added to the place *Arbitrator* as part of initial marking.

### III. CYBER PHYSICAL SYSTEM ANALYSIS

A CPS system is often a hybrid system consisting of both continuous hardware devices and discrete computation processes. In most cases, the only available technique for continuous components is simulation. High level Petri nets are executable and thus support simulation of hybrid system models. Formal verification techniques based on symbolic reachability analysis is available for sub classes of hybrid systems such as those can be modeled using linear hybrid automata [1] where the state transition rates are constants with restricted checking and updating actions. Our tool PIPE+ supports simple reachability analysis and model checking using

SPIN in addition to simulation.

Model checking performs exhaustive search on finite state systems and thus is not directly applicable to continuous systems. However we may be able to model check the bounds (called barrier certificates) of some continuous state variables. PIPE+ has a translator that automatically converts a high level Petri net model to a Promela program in SPIN. During the translation, each place is translated into a channel with the place’s type. This kind of conversion may not always work due to the loss of precision since Promela only supports integer. There are currently two translation schemes:

- (1) Translating each transition as an inline function consisting a part realizing the precondition – checking the enabling condition, and another part capturing the post-condition – transition firing. Each transition is non-deterministically selected in a loop within a single Promela process;
- (2) Translating each transition as a Promela process. Each translation schema has its own advantages and disadvantages. The first one seems more efficient in checking safety properties, while the second one can be used to check liveness property using strong fairness assumption.

The translated Promela model after adding linear time temporal logic specifying properties is model checked using SPIN. Safety and liveness properties are expressed in the general form  $\square$ placename(x) and  $\diamond$ placename(x) respectively, where  $\square$  and  $\diamond$  are the temporal operators always and sometimes in SPIN and x can be a variable or a constant (a specific token). More complex formulas are defined using logical connectives.

With regard to behavioral programming, we can define many generic safety and liveness property patterns using linear time temporal logic, and then instantiate the patterns using concrete behaviors and check them using SPIN model checker. Some generic property patterns include (where B, B1, B2 denote place names representing different behaviors, x and y denote symbolic tokens):

$$(1) \quad \diamond B(x) \tag{G1}$$

This liveness property states that a behavior B will eventually active;

$$(2) \quad \square(B(x) \rightarrow \diamond !B(x)) \tag{G2}$$

This liveness property states that an active behavior B will eventually terminate;

$$(3) \quad \square !(B1(x) \wedge B2(y)) \tag{G3}$$

This safety property states that two behaviors B1 and B2 cannot be active at the same time due to the sequential nature of behavioral programming;

$$(4) \quad \square(B1(x) \rightarrow \diamond B2(y)) \tag{G4}$$

This liveness property states that a behavior B1 leads to behavior B2.

More sophisticated properties can be defined such as there is one particular behavior in between two other behaviors.

Here we provide our model checking results of the concrete properties for in the car parking system.

$$\diamond \langle \rangle \text{SearchingLane}(v1,v2,found) \tag{C1}$$

$$\square (\text{SearchingLane}(v1,v2,found) \rightarrow \diamond !\text{SearchingLane}(v1,v2,found)) \tag{C2}$$

$$\square !(\text{DetectingEntO}(v1,v2,ent1) \wedge \text{SearchingLane}(v1,v2,found))$$

(C3)

[](DetectingEntO(v1,v2,ent1) →  
    <> SearchingLane(v1,v2,found)) (C4)

Since the concrete values of symbolic variables v1 and v2 are not used in checking the above properties, we use bit type to abstract their types to reduce the number of states and instantiate their values according to the initial marking. Furthermore, these properties are about the same car, we can restrict our initial marking to one car in place Arbitrator. Also checking liveness property (C1) can be done more effectively in SPIN by finding a counter example of its negation:

[] !SearchingLane(v1,v2,found) (C1\*)

With the above abstraction and reduction to the resulting Promela model and using the -DBITSTATE storage option in SPIN, we have checked all of the above properties as shown in Table II.

Property	Satisfied	Depth	Stored States	Time
C1*	No	128	201	5(ms)
C2	Yes	763	554077	895(ms)
C3	Yes	611	550966	923(ms)
C4	Yes	140	540451	897(ms)

#### IV. MODEL REALIZATION

Design models help us to better understand system features including functionality, structure, and behavior as well as to detect and prevent early system development errors. To leverage the design models to increase productivity and improve code quality, model driven development based on UML emerged in the last decade [10], in which UML based models are translated into programs of object oriented programming languages. However since there are multiple UML notations such as class diagram, state machine diagram, and sequence diagram for representing different aspects of a system, it is not easy to obtain a coherent set of code. In [3], we presented a model driven approach to realize our high level Petri net models, which provided a systematic way of writing Java programs and establishes the traceability between the models and resulting programs. Our model driven approach consists of the general code structure and domain specific refinement. The general code structure is systematically generated from the agent models and the overall system model. However the domain specific refinement requires manual process in identifying and defining additional features of the system, especially with regard to the physical devices. Different from [3] where each agent net was mapped to a thread in Java, this paper maps each behavior net into a behavior program [11] that includes 3 template methods: boolean takeControl(), void action(), and void suppress().

The following translation rules are used to generate the general code structure from high level Petri net models:

(1) A behavior program is generated for each behavior net, in which 3 methods are created corresponding to transitions *controlB* and *suppressB*, and place *Behavior* shown in Fig.3. The body of each method is empty and requires manual refinements. The constraint of the transition is attached as comments for ensuring the correct implementation;

(2) A behavior object class is created, which is to be manually

refined according to the application domain;

(3) The main program is created based on the initial system net with a single place *Arbitrator*, and includes the definition of an arbitrator object and the instantiations of all the behaviors. In a behavior model, the control flows between behaviors are enforced through a data field in a behavior object. In behavior programming, the control flows are based on the priorities of the behaviors according to their appearances in the behavior array from low to high. As a result, manual reordering is needed to ensure correct control flows. Additional manual refinements are necessary to make the program complete;

(4) A Java project is created to include the above code files.

The above code generation rules are implemented in PIPE+. A Java project is automatically generated from net by selecting BehaviorProgram under Export button in File pulldown menu.

The following code segments are automatically generated by PIPE+ from the multi robotic car parking system model:

```

package parkingsystem.behavior;
public class DetectEntryOne implements Behavior {
    private boolean suppressed = false;
    public boolean takeControl() {
        //TODO:: to be implemented
        //Pre: v.field3=waiting
        //Post: c.field1 == v.field1 && c.field2 ==
        v.field2 && c.field3 == ent1
    }

    public void action() {
        //TODO:: to be implemented
        suppressed = false;
    }

    public void suppress() {
        //TODO:: to be implemented
        //Pre:
        //Post:
        suppressed = true;
    }
}
...
package parkingsystem.object;
public class Robot {
    //TODO:: to be implemented
}
package parkingsystem.main;
public class Main {
    //TODO:: to be implemented
    public static void main(String[] args) throws
    Exception {
        //TODO:: to be refined
        Robot robot = new Robot();
        ...
        Behavior detectEntryOne = new DetectEntryOne();
        ...
        Behavior[] behavior_array =
        { ...
        detectEntryOne,
        ...
        };
        Arbitrator arbitrator = new
        Arbitrator(behavior_array);
        ...
    }
}

```

The actual LEGO car parking system implementation refines the above code templates with domain specific functions imported from `lejos.robotics` and `lejos.hardware` packages.

## V. RUNTIME VERIFICATION

Runtime verification is a lightweight formal approach to detect violation of properties during the execution of a system. It complements the formal methods applied to system models such as model checking and theorem proving by detecting errors either introduced in the process of model implementation or undetected at model level due to the abstraction of models and limitations of formal methods.

Runtime verification was adopted in the multi-car parking system to ensure dependability at implementation level. In our work, properties are specified using linear temporal logic (LTL) formula built from the atomic propositions defined using events written in JavaMop [12]. Monitors are generated from LTL formulas and woven into system implementation as aspects using AspectJ [13]. This ensures the independence of system implementation from monitor – the runtime verification code.

To monitor systems developed with the behavior-oriented approach, several major events are defined for each behavior: *takecontrolT*, *takecontrolF*, *actionR*, *actionE*, and *suppress*. Event *takecontrolT* occurs whenever the method `takeControl()` in the behavior-generated code is executed and returns true. Event *takecontrolF* is similar to *takecontrolT* except the value false is returned. Event *actionR* occurs whenever the method `action()` of the behavior becomes active, while event *actionE* occurs whenever the method `action()` is executed. Event *suppress* occurs whenever the method `suppress()` of the behavior is executed. To distinguish these events defined for different behaviors, behavior name is added in the front of these events. To make the formula more concise, we use the behavior name only to represent the event *actionE*. The following JavaMop code shows an event definition for the behavior `DetectingEntranceOne`. Event definitions for other behaviors are similar.

```
event DetectingEntranceOne_takecontrolT
after(DetectingEntranceOne bhv) returning(boolean b):
execution(public boolean
DetectingEntranceOne.takecontrol()) && this(bhv) &&
condition(b)
{
    //code to be executed when the event occurs;
}
```

In JavaMOP, the properties to be monitored are specified as LTL formulae using defined events, and are evaluated against an execution trace abstracted as a sequence of events. As event definition implies, an event represents the occurrence of a concrete action, typically the entry or exit of an action, which can be calling a method, executing a method, or updating a primitive variable. Events are atoms when used in a LTL formula. In a sequence of events, an event atom is true only when it matches the corresponding event occurrence.

Due to the competition and sequential nature of behaviors in the multi-car parking system, we divided the properties to be monitored into two groups: properties of the arbitrator, and

properties on the temporal relations among different behaviors. The former properties ensure the correctness of the arbitrator. The latter properties ensure the correct behavior order from the system specification.

Property patterns of the arbitrator include:

(A1) A behavior *b* becomes active only if it is selected by the arbitrator:  $\square (b\_actionR \rightarrow \langle * \rangle b\_takecontrolT)$ , where  $\langle * \rangle$  is the past temporal operator previously;

(A2) A behavior *b* will become active:  $\diamond (b\_actionR)$ ;

(A3) Current behavior *b* will eventually terminate if the arbitrator calls its `suppress()` method:  $\square (b\_suppress \rightarrow \diamond b\_action)$ ;

(A4) Two behaviors *b1* and *b2* cannot be active at the same time:  $\square ((b1\_actionR \rightarrow !b2\_actionR \cup b1) \wedge (b2\_actionR \rightarrow !b1\_actionR \cup b2))$ , where  $\cup$  is the until operator;

(A5) If both behaviors *b1* and *b2* are ready to become active, the arbitrator always picks *b1* assuming *b1* has a higher priority over *b2*:  $\square ((b2\_takecontrolT \rightarrow \langle * \rangle b1\_takecontrolF)$ .

Properties (A2) and (A4) correspond to the generic properties (G1) and (G3) at the model level. Property (A1) involves some past concept that cannot be represented in SPIN model checker. Property (A5) with regard to behavior priorities is dealt with using an attribute of a token at the model level. Properties (A2) and (A3) are liveness properties, therefore cannot be verified at runtime since the monitor doesn't know when "the good thing" will happen. To effectively monitor these liveness properties, a timeout event is introduced to make these properties bounded (thus turning them into safety properties). For example (A2) becomes (A2'):  $\diamond (!timeout \cup b\_actionR)$ .

Property patterns relating different behaviors include:

(B1) An event *e1* occurs at most once before another event *e2*:  $\square (e1 \rightarrow \circ (!e1 \cup e2))$ , where  $\circ$  is next operator;

(B2) Whenever an event *e1* occurs, another event *e2* should occur later:  $\square (e1 \rightarrow !timeout \cup e2)$ , which corresponds to generic property (G4) at the model level;

(B3) Whenever an event *e1* occurs, another event *e2* must occur before it:  $\square (e1 \rightarrow \langle * \rangle e2)$ ;

(B4) An event *e1* should never occur before the first occurrence of another event *e2*:  $!e1 \cup e2$ ;

(B5) An event *e1* should never occur after another event *e2*:  $\square (e2 \rightarrow \square !e1)$ ;

(B6) An event *e1* should never occur between event *e2* and event3:  $\square (e2 \rightarrow !e1 \cup e3)$ .

An experiment was conducted to verify the effectiveness of monitoring behavior of the arbitrator and temporal orders of multiple behaviors. In the experiment, we have two LEGO cars running the same piece of code in a parking garage with two different entrances and one exit. The LEGO cars can enter the parking garage through the same or different entrances. When entering the garage through entrance one 1, a car can park at lot 1, 2, 3, or 4; otherwise, the car can only park at lot 3 or 4. To simplify the situation, there is only a one-way lane without

circle. Both cars monitor the same set of properties including 5 concrete A type and 6 concrete B type properties. During the experiments, some properties failed due to the unreliable nature of color sensors. We also noticed the priorities of the behaviors have a major impact on the overall system performance: frequent checking of the readiness of a high priority behavior has a huge negative impact on the performance of the robotic cars. Thus it is important to design the monitors carefully to reduce the performance penalty.

## VI. RELATED WORK

Our CPS development approach covers many research topics including system modeling, system analysis using simulation and model checking, model driven development, and runtime verification. Our main contribution is a systematic CPS development approach by integrating successful existing technologies. Thus we only discuss several most relevant CPS development methodologies.

In [14], a general model-based design methodology for CPSs was proposed. A cook book process was defined, which contains ten general steps in developing a CPS. The process was demonstrated through a bouncing ball example. This methodology is generic and independent of a particular formal model, and thus is not supported by a tool chain.

In [7], an actor-oriented design approach was described for modeling CPSs. Actors are used to model components that communicate through ports. This design approach adopts a multiple model view and is supported by the modeling and simulation environment Ptolemy for heterogeneous systems. Several experimental component modeling modules have been developed, including discrete events (DEs), continuous time (CT), finite state machines (FSMs), synchronous reactive (SR), process networks (PNs), and data flow models. Hybrid system models are obtained by hierarchically composing CT models with discrete models such as FSM or DE. Although this approach provides powerful system modeling and analysis capabilities, it does not cover code generation and code level analysis.

In [15], a foundational framework, called VeriDrone, for reasoning about CPSs at all levels from high-level models to C code was presented. VeriDrone becomes a built-in library of the theorem prover CoQ and enables CoQ users define and verify CPS related properties. This work focuses on formal analysis of CPS, but does not address how to model and design CPSs.

In [3], we developed an overall framework for developing CPSs. This framework is model driven and based on a single formalism – high level Petri nets. This paper extends our framework in [3] with the following new results: (1) a new behavior-oriented approach for modeling internal behaviors of within agents, (2) a net pattern for modeling individual behaviors, (3) a set of property patterns for specifying behaviors, (4) a new translation scheme for generating behavior programs from high level Petri nets, and (5) a set of runtime property patterns for monitoring behavior program execution.

## VII. CONCLUSION

This paper presented a systematic approach for developing

CPSs supported by a tool chain. High level Petri nets are used for modeling CPSs due to their capability in addressing the critical features including concurrency and timing of CPSs. This approach supports a multi-level incremental modeling consisting of behavior-oriented approach for capturing the internal behaviors within agents, agent-oriented approach for system decomposition, and aspect-oriented approach for system composition. The resulting models are analyzed using simulation and model checking to detect early design problems. A translation method for generating general behavior program structure from high level Petri net models is provided. The resulting general behavior program is manually refined with domain specific code to obtain a complete program. This partial manual process of domain specific refinement requires creativity in adding details and thus is unavoidable; however is minimized in our framework. We are currently working on genetic algorithms to further automate the code refinement process. Implementation level quality assurance is carried out using runtime verification. We have developed a set of property patterns based on behavior programming. We demonstrated our approach thorough a multi robotic car parking system. We are currently working on a drone system to gain more experience with regard to the applicability and scalability of our approach.

## ACKNOWLEDGMENT

This work was partially supported by AFRL under FA8750-15-2-0106. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

## REFERENCES

- [1] R. Alur: "Principles of Cyber-Physical Systems", MIT Press, 2015.
- [2] E. Lee: "Cyber Physical Systems: Design Challenges", Proc. of International Symposium on Object/Component/Service-oriented Real-Time Distributed Computing, Orlando, FL, 2008, 363-369.
- [3] X. He, Z. Dong, H. Yin, Y. Fu: "A Framework for Developing Cyber Physical Systems", Proc. of the 29th International Conference on Software Engineering and Knowledge Engineering, Pittsburgh, July 5-7, 2017.
- [4] D. Alam and X. He: "A Method to Analyze High Level Petri Nets using SPIN Model Checker", Proc. of the 29th Int'l Conf. on Software Engineering and Knowledge Engineering, Pittsburgh, 2017.
- [5] Gerard Holzmann: The SPIN Model Checker, Addison Wesley, 2004.
- [6] R. Alur, C. Courcoubetis, N. Halbwachs, T. Henzinger, P. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine: "The algorithmic analysis of hybrid systems", Theoretical Computer Science, vol. 138, 1995, 3 – 34.
- [7] P. Derler, E. Lee, and A. Vincentelli: "Modeling Cyber-Physical Systems", Proceedings of the IEE, vol. 100, no.1, 2012, 13 – 28.
- [8] X. He: "A Comprehensive Survey of Petri Net Modeling in Software Engineering", International Journal of Software Engineering and Knowledge Engineering, vol. 23, no. 5, 2013, 589-626.
- [9] D. Harel, G. Katz, R. Marelly, and A. Marron: "First Steps towards a Wise Development Environment for Behavioral Models", International Journal of Information System Modeling and Design, vol. 7, no. 3, July-September, 2016.
- [10] B. Selic: "The Pragmatics of Model-Driven Development", IEEE Software, 2003, 10 – 25.
- [11] <http://www.lejos.org/nxt/nxj/tutorial/Behaviors/BehaviorProgramming.htm>
- [12] D. Jin, P. Meredith, C. Lee, and G. Rosu: "JavaMop: Efficient Parametric Runtime Monitoring Framework", International Conference on Software Engineering, Zurich, Switzerland, June 2 – 9, 2012.
- [13] The AspectJ Project homepage: <https://eclipse.org/aspectj/>.
- [14] J. Jensen, D. Chang, and E. Lee: "A Model-Based Design Methodology for Cyber-Physical Systems", Proc. of the First IEEE Workshop on Design, Modeling, and Evaluation of Cyber-Physical Systems (CyPhy), Istanbul, Turkey, 2011.
- [15] G. Malecha, D. Ricketts, M. Alvarez, and S. Lerner: "Towards Foundational Verification of Cyber-physical Systems", 2016 Science of Security for Cyber-Physical Systems Workshop (SOSCYPS), 2016.

# Investigating Technical Debt Folklore: A Replicated Survey

Nicolli Rios

Department of Computer Science  
Federal University of Bahia  
Salvador, Brazil  
nicollirios@gmail.com

Manoel Mendonça

Department of Computer Science,  
Federal University of Bahia  
Fraunhofer Project Center @ UFBA  
Salvador, Brazil  
manoel.mendonca@ufba.br

José Amâncio Macedo Santos

State University of Feira de Santana  
Feira de Santana, Brazil  
zeamancio@uefs.br

Rodrigo Oliveira Spínola

PPGCOMP, Salvador University  
Fraunhofer Project Center at UFBA  
State University of Bahia  
Salvador, Brazil  
rodrigo.spinola@unifacs.br

**Abstract**—[Context] The software engineering community considers the technical debt (TD) concept intuitive, because it facilitates discussion among team members about problems that can impact the software development. Personal opinions and experiences related to the concept have been published in blogs and other channels without any evaluation, originating the TD Folklore. [Goal] This work aims to investigate TD Folklore statements classifying them by agreement and consensus. Besides, we also investigated if software development experience affects the perception of developers. [Method] We replicated a survey to evaluate TD Folklore statements. In the replication, we increased the number of respondents and added a new research question to analyze the difference of opinions between participants with and without software experience. [Results] At total, the survey was answered by 107 respondents. The list of TD Folklore was reorganized by the ranking of agreement and consensus indicated by participants. We also identified that professional experience does not change the participants' perception on the concept of TD for the most cases. [Conclusion] We believe that TD Folklore can help researchers and practitioners identify gaps for new research efforts.

**Keywords**—Technical debt; technical debt folklore; survey

## I. INTRODUCTION

Ward Cunningham cited technical debt (TD) for the first time in 1992 as: "shipping first time code is like going into debt. A little debt speeds development so long as it is paid back promptly with a rewrite. The danger occurs when the debt is not repaid. Every minute spent on not-quite-right code counts as interest on that debt." [5]. Since then, the concept, that originally had its scope limited to source code issues, has been expanded and considered in different stages of a software development project [1][2].

Currently, it is common to see subjective opinions, personal points of view and catch phrases about TD in blogs and websites. All this attention-grabbing information has raised

concerns, since it was not evaluated before it was published and reflects only the opinions and experience of the authors. This scenario characterized by different and contradictory opinions, but without any evaluation, could led to the emergence of TD Folklore [12]. The term folklore corresponds to traditional stories, beliefs and customs of a group of people. TD Folklore needs to be investigated, because it may contain valuable information about experiences that could contribute significantly to the study of the area. Thus, it can help researchers formulate theories and hypotheses, and identify gaps to direct new research efforts.

In this context, Spínola *et al.* [12] conducted searches on the Internet looking for TD folklore statements. The search was performed on online websites, blogs, and published papers. As result, the authors selected a list of 14 potential TD folklore statements. After, the authors performed a survey with the purpose of answering the following research questions:

**(RQ1) Agreement:** With which folklore statements did participants agree or disagree?

**(RQ2) Consensus:** How strong is the consensus on each of the folklore statements?

The rationale behind these questions was that if any folklore is either widely agreed to or disagreed with by a large group of people, then those propositions are more likely to be good candidates for future research. On the contrary, mixed responses can indicate that a TD Folklore item is not commonly believed, depends on many factors, or that the statement itself is not yet formulated as precisely as needed.

The results initially presented gave us interesting insights on the subject. From then, we extend the work of Spínola *et al.* [12] by replicating their study<sup>1</sup>. In this paper, we present this replication, which has two main objectives: i) to mitigate

---

<sup>1</sup> Replication based on previous insights is widely recommended in the experimental paradigm [3][13][14].

limitations of the previous study and; ii) to expand the knowledge on the topic. We address these points as follows. First, the main limitation of the previous study presented by Spínola *et al.* [12] is related to the number of participants. In this replication, we expand the population of the study from 37 to 107 (70 new participants). Second, the insights of the previous survey indicated that we need to investigate how the developers' experience impacts on TD Folklore statements. Then, in this replication, we also revisited the original research questions including the following new question:

**(RQ3) Behavior:** Do participants with and without software development experience have the same perception on TD folklore statements?

By answering RQ3, we intend to collect evidence that could provide some support for a known claim in the TD area: one of the advantages of the TD concept is that it has a common understanding in the software development community. We want to analyze this claim by investigating if the lack of industry experience affects the perception of software engineers about TD folklore items.

As a result of this replication, we highlight that the agreement and consensus analysis (RQ1 and RQ2) reinforce previous findings reported by Spínola *et al.* [12]. We also found that the participants' experience (RQ3) does not affect their agreement with the TD folklore statements, for the most cases. This builds evidence on the idea that TD concept has a common understanding among developers and represents a simple metaphor to discuss software development problems.

The contribution of this paper is twofold. First, we reevaluated and reorganized the TD Folklore list by rank of agreement and consensus. We also complemented this analysis by investigating if experience on software development activities influences the perception of participants about each TD folklore statement. Second, this replicated study will help us in understanding what participants have said about TD and what folklore seems to make sense and constitute good candidates for more detailed investigation.

In addition to this introduction, this paper has other five sections. Section 2 presents a background on the area. Then, in Section 3, the TD Folklore's survey will be presented. Section 4 discusses the results of this replication. Next, Section 5 presents some threats to validity of this study. Finally, Section 6 presents the final remarks of this work.

## II. BACKGROUND

Different surveys have been conducted in the TD area. Klinger *et al.* [9] interviewed four experienced software architects to investigate how decisions on incurring debt are taken within a company and what is the extent of the consequences of those decisions. They concluded that often the decision to incur debt is not direct action of the architects, but a consequence of activities carried out by people that do not perform technical activities in the project.

In another study, Lim *et al.* [10] characterized how software professionals perceive and understand the context in which TD occurs. After interviewing 37 professionals, they concluded

that to deal with the balance related to TD, professionals must make this debt explicit, communicate their costs and benefits to all stakeholders and manage it making its presence healthy for the project.

Snipes *et al.* [11] conducted an interview with the change management committee regarding the defect debt management. As a result, they indicated that the highest cost of this type of debt was related to their identification and validation activities (cost of testing). In addition, the authors also identified that there are six major components which affect decisions about incurring/paying a debt item: severity, existence of an alternative solution, urgency of the correction, effort to implement the correction, risk of the proposed correction, and the extent of the required test.

In another study, Codabux and Williams [4] conducted a survey to identify best practices with respect to TD management. They analyzed 28 teams working with Scrum. As results, they reported that: (i) developers considered their own TD taxonomy based on the type of work they performed and their personal understanding of the term; (ii) developers pay more attention to design and test debt; and (iii) having dedicated teams to eliminate debt items during sprints is a good initiative to reduce TD.

Holvitie *et al.* [8] conducted a survey with professionals from Finland and found that most participants were familiar with the term TD. The authors also pointed out that more than half of the interviewees realized that practices directly related to software implementation have a positive effect on TD and its management. Finally, it was also identified that the project stage most affected by TD is the implementation, and the main cause for the occurrence of TD is an inadequate definition of its architecture.

More recently, Ernst *et al.* [7] reported the results of a survey with software engineers and architects. The authors found that architectural decisions are the most important source of technical debt. Furthermore, while respondents believe the metaphor is itself important for communication, existing tools are not currently helpful in managing the details.

Finally, Spínola *et al.* [12] investigated the level of agreement of software professionals with phrases of effect ("TD Folklore") on TD. The results of this study indicated that TD is an important factor in software project management and not simply another term for "bad code". The replication process of this study, as well as the results obtained will be discussed in the sequence.

## III. TD FOLKLORE SURVEY REPLICATION

### A. TD Folklore Survey

The goal of the research conducted by Spínola *et al.* [12] was to evaluate a set of folklore statements about TD. For that, a survey was conducted with professionals in the area of software engineering. Its goal was to answer the research questions **RQ1** and **RQ2**.

The survey contains statements about TD, collected on websites, blogs and published articles. This list has only

TABLE I. TD FOLKLORE LIST

ID	TD Folklore Statement			Groups
		Agreement	Consensus	
1	Accruing technical debt is unavoidable on any non-trivial software project.	3	1	No tendency
2	Technical debt usually comes from short-term optimizations of time without regard to the long-term effects of the change.	4	2	Agreement and high to medium consensus
3	It is very difficult for software developers to see the true effect of the technical debt they are incurring.	3	2	No tendency
4	“Working off debt” can be motivational and good for team morale.	4	2	Agreement and high to medium consensus
5	The root cause of most technical is pressure from the costumer.	3	1	No tendency
6	Unintentional debt is much more problematic than intentional debt.	4	2	Agreement and high to medium consensus
7	The individuals choosing to incur technical debt are usually different from those responsible for servicing the debt.	3	1	No tendency
8	If technical debt is not managed effectively, maintenance costs will increase at a rate that will eventually outrun the value it delivers to customers.	4	1	Agreement and high to medium consensus
9	No matter what, the cost of fixing technical debt increases the longer it remains in the system.	4	2	Agreement and high to medium consensus
10	Paying off technical debt doesn't result in anything the customers or users will see.	2	2	Disagreement and medium consensus
11	The biggest problem with technical debt is not its impact on value or earnings, but its impact on predictability.	3	2	No tendency
12	Technical debt should not be avoided, but managed.	3	2	No tendency
13	Not all technical debt is bad.	3	2	No tendency
14	All technical debt is intentional.	1	1	Strongly disagreed

statements based on personal opinions and experiences without any evaluation.

The survey was structured into two sets of questions. The first one aims to establish the level of knowledge of the interviewees about software development and TD. In the second one, the survey contains 14 sentences (see Table I) and, for each of them, the authors asked the participants to indicate their level of agreement. The questionnaire used the 5-point Likert scale to indicate the level of agreement: "1: strongly disagree" to "5: strongly agree". In addition, participants had the option "I do not know". The survey was designed to be answered in about ten minutes.

To perform the data analysis, for RQ1 (agreement), the authors computed the median as indicator for central tendency. Thus, a median of 4 or 5 shows tendency towards agreement on a statement. On the opposite side, values of 1 and 2 indicate a tendency towards disagreement. For RQ2 (consensus), the authors calculated the spread in the distribution of responses for each statement by computing the inter quartile range (IQR). An interval size value of 1 indicates a low spread and high consensus. On the contrary, higher values show more spread and indicate less common opinion among participants.

### B. Survey Replication

This section details how we evolved the initial design by adding RQ3, and how we planned and performed the replication of the study.

#### 1) Procedure

To replicate the survey, it was not necessary to make any change in the original questionnaire. The main differences between the study and its replication rely on its:

- analysis methodology for RQ3 (discussed in Section III.B.3);
- population: differently from the original study that focused on practitioners, our replication has focused on both participants with experience and those with none prior experience on software development activities.

#### 2) Data collection and subject characterization

We replicated the questionnaire in undergraduate and graduate software engineering classes with participants of differing expertise and background. Not all participants had experience with software development, however, theoretical concepts were presented in the software engineering discipline. Prior to the application of the questionnaire, the basic concepts of TD were presented to ensure that everyone knows the term. The concepts were carefully presented by the last author in order to do not affect the perception of the participants regarding the list of folklore statements.

In total, 70 participants (see Table II) answered the replicated questionnaire and the average time to complete it was 15 minutes. The survey participants were also asked for their target degree and years of experience, as well as the roles they had taken in software projects. Almost half of them (36) do not have experience with software development. Among the other 34 participants that have some experience on software development activities, most of them were developers (29), followed by project managers (6) and requirements analyst (6).

TABLE II. SUBJECTS' CHARACTERIZATION

Role	[12]	Replication	Total
Developer	29	29	58
Project Manager	9	6	15
Requirement Analyst	2	6	8
Tester	4	1	5
Architect	3	0	3
Operations	1	0	1
Maintainer	1	0	1
Database Administrator	0	1	1
Academic Degree	[12]	Replication	Total
Undergraduate Student	2	52	54
Bachelor in Comp. Science	2	0	2
Graduate Student	14	0	14
Master Student	1	16	17
PhD Student	1	2	3
Undefined	17	0	17
# Experienced / no Experienced Subjects	[12]	Replication	Total
# Subjects with Soft. Exp.	37	34	71
# Subjects with no Soft. Exp	0	36	36
Years of Software Experience	[12]	Replication	Total
Mean	4.8	4.8	4.8

The mean time of experience for those who have software experience was 4.8 years (coincidentally, the mean time of experience was the same considering both data sets). We can also see on Table II that most of participants (54) are undergraduate students, followed by master students (14) and PhD students (2).

Finally, by analyzing the whole population scenario (107 participants), we can notice that the most of participants have some experience as developer (58), followed by project managers (15) and requirements analyst (8). We can also notice that we have approximately 2/3 of participants with (71) and 1/3 of them without (36) prior experience on software development.

### 3) Analysis methodology

The research questions RQ1 and RQ2 were analyzed considering the whole dataset, including the data collected in [12]. Besides, the same methodology (based on median for RQ1 and inter quartile range for RQ2) considered by [12] was applied. In order to address RQ3, initially, we divided the whole dataset into two subsets representing participants with (71 subjects) and without (36 subjects) experience on software development. Our approach was twofold.

First, we computed the median and IQR values for each statement of each subset. Then, we compared differences between median and IQR values for three subsets (all participants indistinctly, and more and less experienced participants). In other words, we observed if the agreement and consensus were similar for each subset. A significant difference on agreement and consensus between these subsets evidences that the experience impacts on the level of agreement for that statement.

Second, we statistically compared the Likert scale values filled in by the participants in different subsets (more and less experienced participants). We adopted the Shapiro-Wilk normality test. For all cases, the distribution was not normal. Due to this, we adopted the Mann-Whitney, a non-parametric alternative to t-test, with a 0.05 p-value, to statistically test our hypothesis. The null hypothesis (H0) is: for a specific TD Folklore, there is no difference of the Likert scale values between more and less experienced participants. Rejecting the null hypothesis (p-value<0.05) evidences that the experience impacts on the TD Folklore level of agreement.

We then considered the two evidences in our analysis: i) the differences on the agreement and consensus; and ii) the statistical Mann-Whitney p-value. When these outcomes presented some inconsistency, we graphically analyzed the distribution of the Likert scale values.

## IV. RESULTS AND DISCUSSION

This study surveyed participants with and without software development experience to investigate: (i) what statements from a TD Folklore list the participants agree with; (ii) what is the consensus around the statements collected about TD, and; (iii) if the perception on TD is similar or different between participants with and without experience on software development.

### A. Agreement with statements and Consensus

Results of both research questions RQ1 and RQ2 are presented and grouped in Table I by central agreement tendency and consensus. Initially, we can see that no single folklore statement was commonly strongly agreed with. This indicates that none of the folklore statements were considered to be universally true. On the other side, there was one folklore statement that was commonly strongly disagreed with (#14). This result (i) suggests that software engineers are aware that there might be unknown TD items in their projects, and (ii) supports the ongoing line of research into tools that analyze source code for unknown debt. In this replication, we can also notice that we did not have higher values for IRQ, all values were close to 1. Thus, in general, there was a low spread and high consensus among the participants for this statement.

Some statements (#2, #4, #6, #8, and #9) presented a median of 4, which indicates a tendency towards agreement and high to medium consensus. These results indicated that there is a common belief that TD is an important part of software management. On the other hand, the statement #10 received general disagreement and medium consensus. This result indicated that from the point of view of participants, the presence of TD items could bring some impact for system users. Finally, seven other statements (#1, #3, #5, #7, #11, #12, and #13) showed no tendency on either side of the scale.

Fig. 1 complements the results of the analysis by median presented on Table I. In this figure, the percentages represent the distribution of answers according to the Likert scale. We can see that there is an inclination of the graph to the right side, indicating agreement (agree or strongly agree) on the folklore statements. By observing this graph, we also can highlight some statements (#2, #4, #5, #6, and #8) that reached a



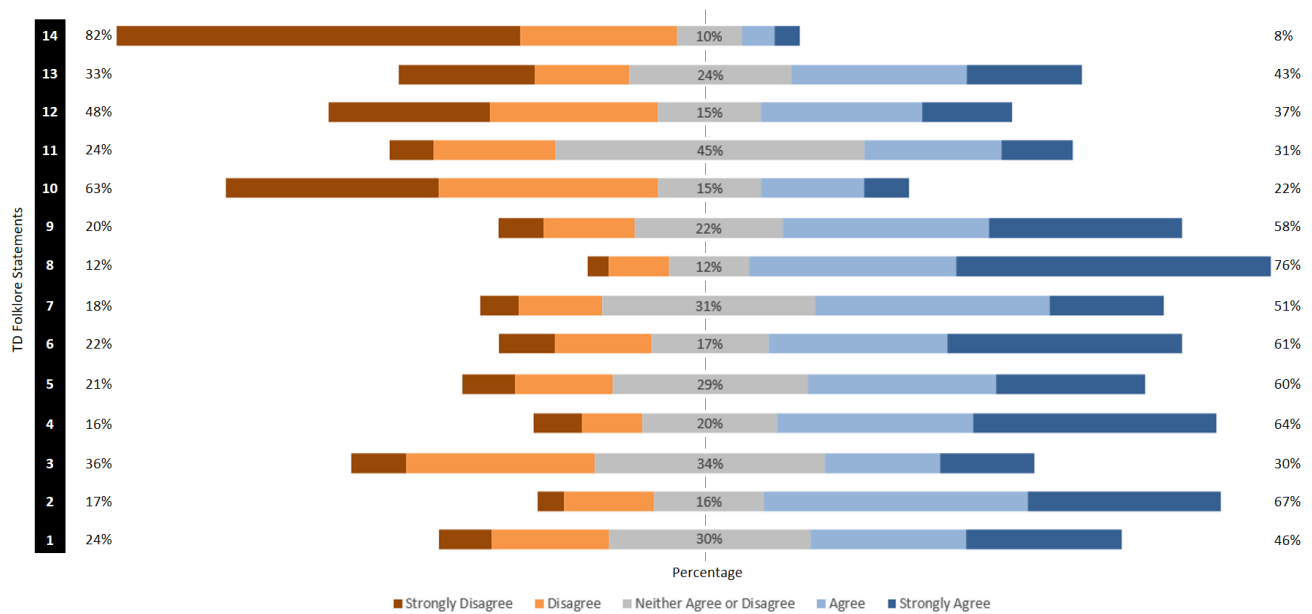


Figure 1. Agreement tendency analysis. Distribution of answers according to the likert scale

widespread agreement (>60% of answers are strongly agree or agree). On the other side, the participants clearly disagreed (>60% of answers are strongly disagree or disagree) with statements #10 and #14. Finally, despite the median analysis did not indicate an agreement tendency regarding the statements #5 and #7, most of the participants agreed with them.

*B. Do participants with and without software development experience have the same perception on TD concepts?*

Results of RQ3 are presented in Table III by central agreement tendency and consensus grouped by the participants' experience, and the Mann-Whitney p-value (last column).

The light gray lines represent the cases where agreement and consensus for both experienced and no experienced participants are similar to the values considering all participants indistinctly, as presented in Table I. Moreover, for these cases, it was not possible to reject the null hypothesis (Mann-Whitney

p-value>0.05). The results evidence that the agreement with the statements was not impacted by the participants' experience for the statements #1, #2, #3, #4, #6, #8, #9, #10, #13 and #14.

The dark gray lines represent the cases where agreement, consensus, and the Mann-Whitney p-value (<0.05) evidence that the agreement with the statements was impacted by the participants' experience. It occurred only for the statements #5 and #12.

For the statements #7 and #11, we found inconsistencies between agreement/consensus and the hypothesis test. In order to better understand these cases, we show the distribution of the participants' agreement with the statements in Fig. 2. Statement #7 has similar distribution considering both group of participants. In opposition, is evident that values for statement #11 are higher for experienced than for no experienced participants. These results reinforce the Mann-Whitney p-value presented in Table III: the participants' experience impacts the agreement with the statement #11, and it does not impact the agreement with the statement #7.

Overall, we observed that the experience impacted only the

TABLE III. AGREEMENT TENDENCY, CONSENSUS AND SHAPIRO-WILK P-VALUE GROUPED BY THE PARTICIPANTS' EXPERIENCE

ID	Experienced Participants		Participants without experience		Mann-Whitney p-value
	Agreement	Consensus	Agreement	Consensus	
1	3	3	3	1	0.3635
2	4	1	4	2	0.2789
3	3	2	3	2	0.5785
4	4	2	4	2	0.9725
5	3	2	4	2	0.0204
6	4	2	4	3	0.9719
7	3	1	4	1	0.0722
8	4	1	4	1	0.7241
9	4	2	4	2	0.2551
10	2	2	2	2	0.2503
11	3	1	3	1	0.0107
12	3	2	2	2	0.0219
13	3.5	2	3	3	0.0789
14	1	1	1	1	0.9574

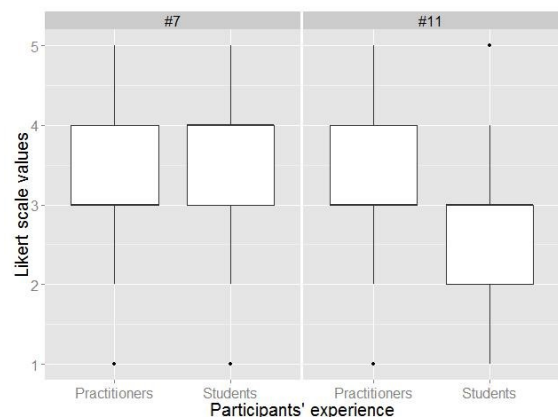


Figure 2. Agreement distribution for the Folklores #7 and #11

results for three of the statements (#5, #7, #12). For us, this indicates that the experience does not have significant impact on the TD Folklore analysis. However, this also evidences that we should not to consider all TD Folklores in the same way. For example, observing TD Folklores #5, #7, #12, we conjecture that the experience might be impacted the results because concerns about customer pressure (#5), decisions about who will pay the debt off (#7) and if the debt should be avoided (#12) seem to be a reflex of the scenarios faced by development teams in their daily activities. We are planning investigate these aspects in the future.

## V. THREATS TO VALIDITY

In this section, we discuss some threats to validity:

*External validity.* The participants of this replication were graduate and undergraduate students. One aspect mitigates the threat: in total, most participants had some professional software development experience. As can be seen in Table II, there are 71 participants with some software development experience against 36 participants without software development experience. Despite this, the results might not generalize to a context in which developers have long years of experience (15 to 20 years, for instance). However, it is important to note that our findings are based on a comparison between two groups of participants that are clearly distinct regarding to the level of professional experience.

*Internal validity.* Another threat to the validity was the possibility of the presentation made on TD before the distribution of the questionnaire influence the responses of the participants. To deal with this threat, the TD concepts were carefully presented by one of the authors of this work.

*Construct validity.* Likert scales assume that participants can accurately map their answers to a question into one dimension (e.g., strongly agree or disagree). Since TD is a complex concept, this may not be realistic in some cases. The TD Folklore statements investigated in this work may not be 100% mutually exclusive and exhaustive.

## VI. CONCLUDING REMARKS

In this paper, we presented the results of a replicated survey on TD folklore. We revisited the original work from Spínola *et al.* [12] by expanding the population (from 37 to 107 respondents) and reviewing the results concerning the agreement/disagreement tendency and consensus about each folklore item. Besides, we also investigated if experience with software development affects the perception of the participants about the considered statements.

Regarding agreement/disagreement tendency and consensus, our results reinforce the findings previously reported in [12]. Thus, the results provide some evidence and motivation for exploring the following issues in TD research: (i) impact of TD management on maintenance costs (#2, #8, #9), (ii) relationship between servicing the debt and team motivation (#4), (iii) relation between unintentional and intentional debt impact on software projects (#6), (iv) how the impact of debt items increases (or decreases) during software

evolution (#9), (v) prediction or estimation models for TD impact (#8, #9), (vi) impact of paying TD items off on customers (#10), and (vii) development of strategies to identify unintentional TD items (#14). Finally, the results suggest that software development experience does not interfere in the perception on the TD concept for the most cases.

In our future research agenda, we intend to combine the evidence identified in this work with new theories and empirical studies developed by our research group. Specifically, we intend to investigate causes and impacts of TD on software projects.

## ACKNOWLEDGMENT

This work was partially supported by the CNPq Universal grant 458261/2014-9, by the State of Bahia's SECTI-Fraunhofer-UFBA cooperation agreement 2012-1, and by RESCUER project Grant: 490084/2013-3.

## REFERENCES

- [1] N.S.R. Alves, R.S. Araújo, and R.O. Spínola. A Collaborative Computational Infrastructure for Supporting Technical Debt Knowledge Sharing and Evolution. In: Americas Conference on Information Systems, 2015, Puerto Rico.
- [2] N.S.R. Alves, T.S. Mendes, M.G. Mendonça, R.O. Spínola, F. Shull, and C. Seaman. Identification and management of technical debt: A systematic mapping study. *Information and Software Technology*, v. 70, p. 100-121. 2016. DOI: <https://doi.org/10.1016/j.infsof.2015.10.008>
- [3] F. Shull, V. Basili, J. Carver, J.C. Maldonado, G.H. Travassos, M. Mendonça, and S. Fabbri. 2002. Replicating Software Engineering Experiments: Addressing the Tacit Knowledge Problem. In Proc. of the 2002 Int. Symp. on Empirical Software Engineering, USA.
- [4] Z. Codabux and B. Williams. Managing technical debt: An industrial case study. In 2013 4th International Workshop on Managing Technical Debt (MTD), (pp. 8-15). IEEE.
- [5] W. Cunningham. The WyCash portfolio management system. In ACM SIGPLAN OOPS Messenger (Vol. 4, No. 2, pp. 29-30). ACM. 1992.
- [6] A. Erickson. 2009. Don't "Enron" Your Software Project. Retrieved from <http://www.informit.com/articles/article.aspx?p=1401640>.
- [7] N.A. Ernst, S. Bellomo, I. Ozkaya, R.L. Nord, and I. Gorton. Measure it? Manage it? Ignore it? Software practitioners and technical debt. In Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2015). ACM, New York, NY, USA, 50-60.
- [8] J. Holvitie, V. Leppanen, and S. Hyrnsalmi. (2014), Technical Debt and the Effect of Agile Software Development Practices on It - An Industry Practitioner Survey. In MTD 2014, pp. 35-42.
- [9] T. Klinger, P. Tarr, R. Wagstrom, and C. Williams. "An enterprise perspective on technical debt," in Proceedings of the 2nd Workshop on Managing Technical Debt. ACM, 2011, pp. 35-38.
- [10] E. Lim, N. Taksande, and C. Seaman. "A balancing act: what software practitioners have to say about technical debt," *Software*, IEEE, vol. 29, no. 6, pp. 22-27, 2012.
- [11] W. Snipes, B. Robinson, Y. Guo, and C. Seaman. Defining the decision factors for managing defects: a technical debt perspective. In 2012 Third Int. Work. on Managing Technical Debt, (pp. 54-60). IEEE.
- [12] R.O. Spínola, N. Zazworka, A. Vetrò, C. Seaman, and F. Shull (2013). Investigating technical debt folklore: Shedding some light on technical debt opinion. In Proc. of the 4th Int. Work. on Managing Technical Debt
- [13] N. Juristo and S. Vegas. Using differences among replications of software engineering experiments to gain knowledge. In Proceedings of the 2009 3<sup>rd</sup> Int. Symp. on Empirical Soft. Eng. and Measurement. IEEE Computer Society, Washington, DC, USA, 356-366.
- [14] F. Shull, J.C. Carver, S. Vegas, and N. Juristo. 2008. The role of replications in Empirical Software Engineering. *Empirical Software Engineering*. 13, 2 (April 2008), 211-218.

# Knowledge Management Governance in Software Development Process with GI-Tropos

Vu H. A. Nguyen  
LouRIM-CEMIS  
Université catholique de Louvain  
Belgium  
Email: vu.nguyenhuynh@uclouvain.be

Manuel Kolp  
LouRIM-CEMIS  
Université catholique de Louvain  
Belgium  
Email: manuel.kolp@uclouvain.be

Yves Wautelet  
Faculty of Economics and Business  
KULeuven  
Belgium  
Email: yves.wautelet@kuleuven.be

**Abstract**— Governing Knowledge Management in a Requirements-Driven Software Processes such as (GI-) Tropos or even waterfall, V, UP-based ones allows IT managers to propose rules for efficient handling of information and resources to cope with stakeholders' requirements. On the one hand, Knowledge Management Governance in software engineering has to ensure that software organization business processes determine organizational knowledge access conditions, quality maintenance, decision making processes and means of resolving knowledge management obstacles of the organization. On the other hand, requirements-driven software methods are development processes using high-level social-oriented models to drive the software life cycle both in terms of project management and deductive iterative engineering techniques. Typically, such methods are well-suited for the straightforward inclusion and adaptation of knowledge management governance principles into the software development life cycle. To consolidate both perspectives, this paper proposes a generic framework allowing to drive Knowledge Management in the GI-Tropos software processes.

**Index Terms**—Knowledge Management Governance, Software Process, GI-Tropos

## I. INTRODUCTION

Software Engineering [1] is a wide knowledge area. It requires various other types of data, information, skills and know-how during the software development and operation processes [2]. It is notably devoted to implement human activities and cope with socio-intentional problems through business modeling and requirements engineering techniques at the strategic level [3]. The software development life-cycle process itself is a structure of groups of activities in which communicating and collaborating tasks are required from any user and stakeholder. At the project and organizational level, each individual's knowledge can (or must) be shared as to what knowledge management activities they do.

Knowledge management (KM) is defined as 'the effective learning processes associated with exploration, exploitation and sharing of human knowledge (tacit and explicit) that use appropriate technology and cultural environments to enhance an organizations intellectual capital and performance.' [4]. The foundations of KM include infrastructure, mechanisms, and technologies [5]. KM mechanisms are organizational or structural instruments that are used to enable KM systems. KM technologies are information technologies that can be

used to promote KM. Both KM mechanisms and KM technologies are supported by the KM infrastructure. From an organizational perspective, KM infrastructure includes five major components: organization culture, organization structure, organization's information technology infrastructure, common knowledge, and physical environment [5].

Common dimensions of knowledge management governance under conceptualization are leadership, organizational structure, and relational mechanisms among stakeholders [6]. The goal of knowledge management governance is to ensure that KM processes deliver value to the identified stakeholders and minimize risks related to organizational knowledge loss [7]. Moreover, governing knowledge management sets the structural principles and rules within which all the components of a knowledge management system should be deployed in the broader organizational context. Unfortunately, little specific research has been completed on KM governance in software development process, including the problem of aligning both approaches. Indeed, most studies on knowledge management governance have rather focused on more wide-ranging fields than just software engineering life cycle.

Therefore, this paper proposes a generic framework to align knowledge management rules and constraints to software processes. The framework uses strategic modeling techniques to represent the organizational setting but also governance and management structures. We will also more specifically discuss the adoption of our framework within particular processes in order to align KM governance with requirements-driven software specification.

This paper is organized as follows. Section 2 overviews our proposed software development template called Governance I-Tropos (GI-Tropos) for governing requirements-driven software process. Section 3 illustrates knowledge management governance in requirements-driven software development process. Section 4 introduces a use case for validation. Finally, Section 5 concludes the paper and points out further work.

## II. GI-TROPOS

Figure 1 represents the GI-Tropos [8] process in a classical iterative perspective based on a series of phases illustrated in the horizontal dimension and a series of disciplines presented in the vertical dimension.

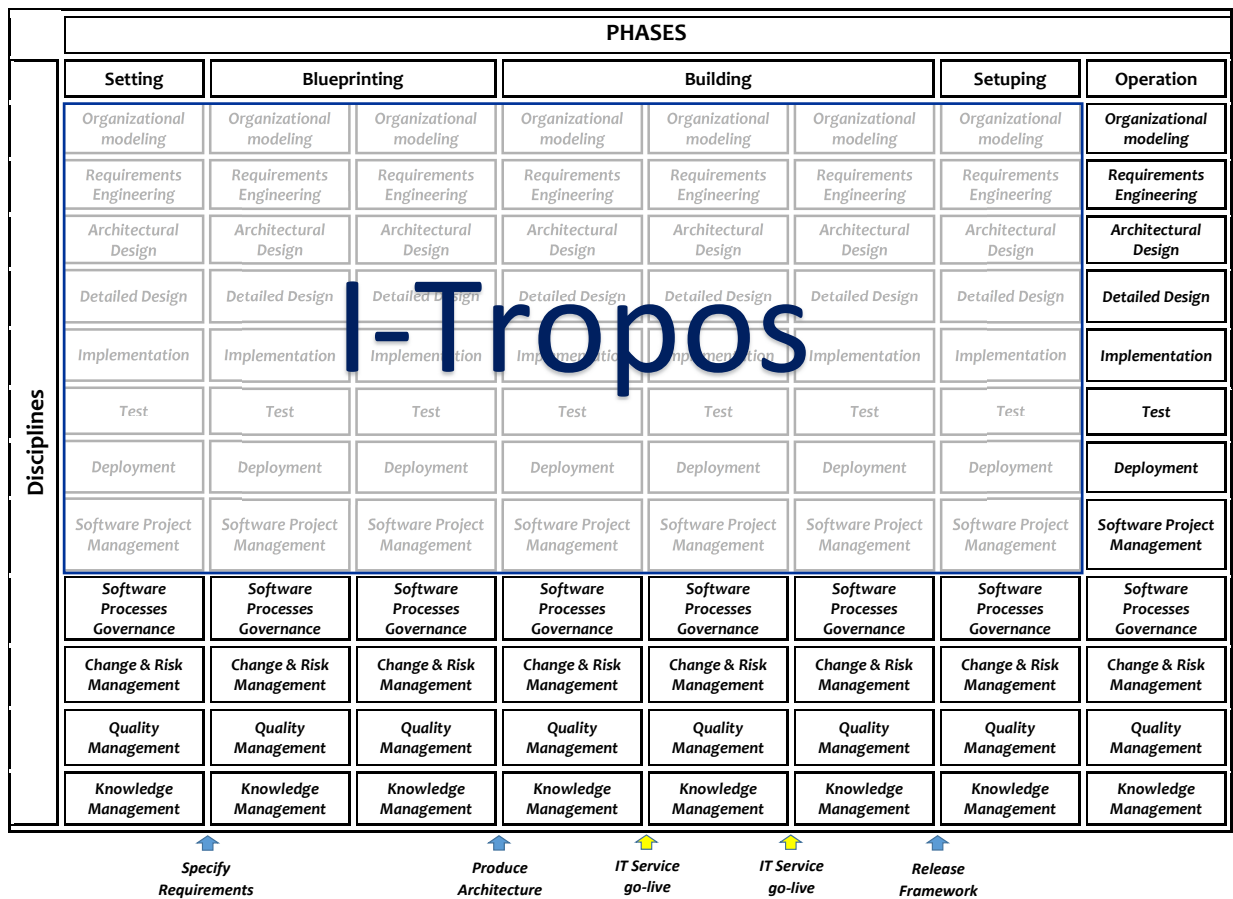


Fig. 1. GI-Tropos iterative process framework

GI-Tropos is an extension of Iterative Tropos (I-Tropos) [9], a development process using coarse-grained (i.e., high-level) and social-oriented requirement models to drive the software development both in terms of project management [10] and deductive forward engineering (transformational) techniques, for aligning requirements-driven software processes with IT governance rules and principles. This extension aims to enable governing and managing requirements-driven software processes to cope with stakeholders' requirements and expectations in the context of business aspects.

I-Tropos extends Tropos [11], a requirements-driven development methodology using the i\* modeling framework [12] that supports iterative [13] and agent development [14]. The five phases of traditional Tropos (Early Requirements, Late Requirements, Architectural Design, Detailed Design and Implementation) are considered as groups of iterations that are workflows with a minor milestone in order to comply with the most generic terminology. In I-Tropos, the Organizational Modeling and Requirements Engineering disciplines respectively correspond to Tropos' Early and Late Requirements phases. The Architectural and Detailed Design disciplines correspond to the same stages of the traditional Tropos process. I-Tropos not only includes core disciplines (i.e., Organizational Modeling, Requirements Engineering, Architectural Design,

Detailed Design, Implementation, Test and Deployment) but also supports disciplines to handle Risk Management, Time Management, Quality Management and Software Process Management [15]. The research method we have followed uses a bottom-up approach, I-Tropos was considered as a given and validated framework and has been enhanced with a governance level.

From a systems development perspective, GI-Tropos improves and redefines the four following phases (Setting, Blueprinting, Building, Setuping) of I-Tropos plus a new one, Operation, to operate the system in the perspective of IT enterprise governance and management. GI-Tropos also includes all I-Tropos disciplines plus four new ones: Software Processes Governance, Change & Risk Management, Quality Management, Knowledge Management. These new disciplines ensure that software processes are evaluated, directed and monitored to meet stakeholders' requirements and achieve value added by aligning requirements-driven software processes with IT governance rules and constraints. They also enable identifying, analyzing and assessing changes and risks as well as developing strategies to manage them. Moreover, these disciplines ensure that quality expected and contracted with stakeholders is achieved throughout the system. Finally, they enable acquiring, storing and utilizing knowledge for such

things as problem solving, dynamic and deep learning, strategic planning, decision making and business processes. GI-Tropos' disciplines are grouped and transversal to each phase. They can be deployed in several iterations by phase depending on each software project characteristics. Consequently, the disciplines of GI-Tropos can be repeated iteratively and the effort/workload spent on each discipline varies from one iteration to another.

### III. KM GOVERNANCE IN GI-TROPOS

This section describes knowledge management governance rules and constraints in a requirements-driven software processes. Figure 2 summarizes the alignment while Figures 3 and 4 illustrate the Strategic Dependency model and Strategic Rationale model for the Software Processes KM governance.

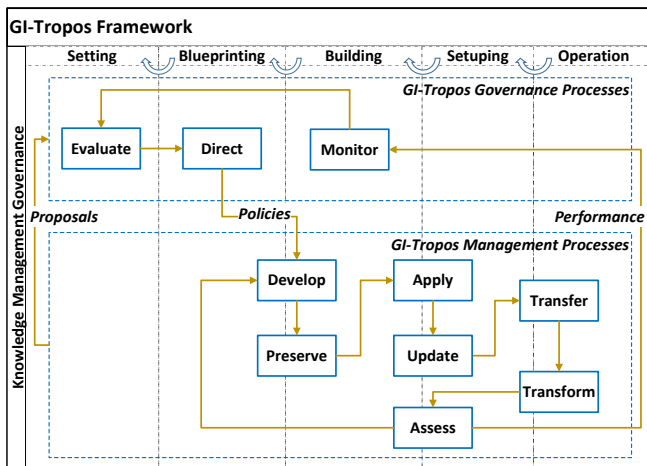


Fig. 2. GI-Tropos Knowledge Management Governance Alignment Model

Knowledge Management Governance processes aligned with requirements-driven software processes such as GI-Tropos, as depicted in Figure 2, can be summarized as follows:

- The **Evaluate** process ensures that stakeholders needs, conditions and options are evaluated to determine balanced, agreed-on organizational objectives to be achieved. It allows examining and judging current and future use of knowledge, considering internal and external pressures, evaluating continuously, considering current and future business needs and objectives: competitive advantage and specific strategies.
- The **Direct** process enables setting direction through prioritization and decision making. It assigns responsibility, directs preparation and implementation of KM plans and policies, sets directions for KM, establishes sound behavior in KM use through policies, properly plans transition of project to operational status, encourages culture of good KM governance, directs submission of proposals identifying needs.
- The **Monitor** process enables monitoring performance and compliance against agreed-on direction and objectives. It allows monitoring and measuring KM performance, assures that performance is in accordance with

plans and business objectives, ensures that KM conforms with external obligations (regulatory, legislation, common law, and contractual) and internal work practices.

- The **Develop** process ensures that knowledge is acquired, captured, created, discovered correctly. It also plans activities in alignment with the direction set by the governance body to achieve the organizational objectives. It covers the use of information and technology and how best it can be used in an organization to help achieve the organization's goals and objectives.
- The **Preserve** process ensures that knowledge is stored, secured, conserved, retained. It also highlights the organizational and infrastructural form KM is to take in order to achieve the optimal results and to generate the most benefits from the use of KM.
- The **Apply** process enables knowledge to be used, enacted, executed, exploited properly.
- The **Update** process enables knowledge to evolve, improve, be maintained and refreshed.
- The **Transfer** process ensures that knowledge is communicated, deployed, disseminated, shared. It also deploys activities in alignment with the direction set by the governance body to achieve the organizational objectives. It identifies KM requirements, acquires the technology, and implements it within the organization's current business processes.
- The **Transform** process ensures that knowledge is compiled, formalized, standardized, explicated. It also delivers activities in alignment with the direction set by the governance body to achieve the organizational objectives. It focuses on the delivery aspects of the KM system.
- The **Assess** process ensures that knowledge is appraised, evaluated, validated, verified. It also assesses activities in alignment with the direction set by the governance body to achieve the organizational objectives. It deals with the organizational strategy in assessing its needs and whether or not the current KM system still meets the objectives for which it was designed and the controls necessary to comply with regulatory requirements. It also covers the issue of an independent assessment of the effectiveness of KM system in its ability to meet business objectives and the organizational control processes by internal and external auditors

The Strategic Dependency model, as depicted in Figure 3, has three main actors depending on each other (Implementer, Knowledge Management Development Board, Knowledge Management Governance Board), resources (Organization Culture, Organization Structure, IT Infrastructure, Common Knowledge, and Physical Environment), goals (Develops Knowledge Management Structure, Continuous Implement Knowledge Management), qualities (Organization Strategy, Knowledge Management Quality), and tasks (Knowledge Management Modeling, Knowledge Management Operation).

The Knowledge Management Governance Board decides on the processes and the environmental factors (risks, quality

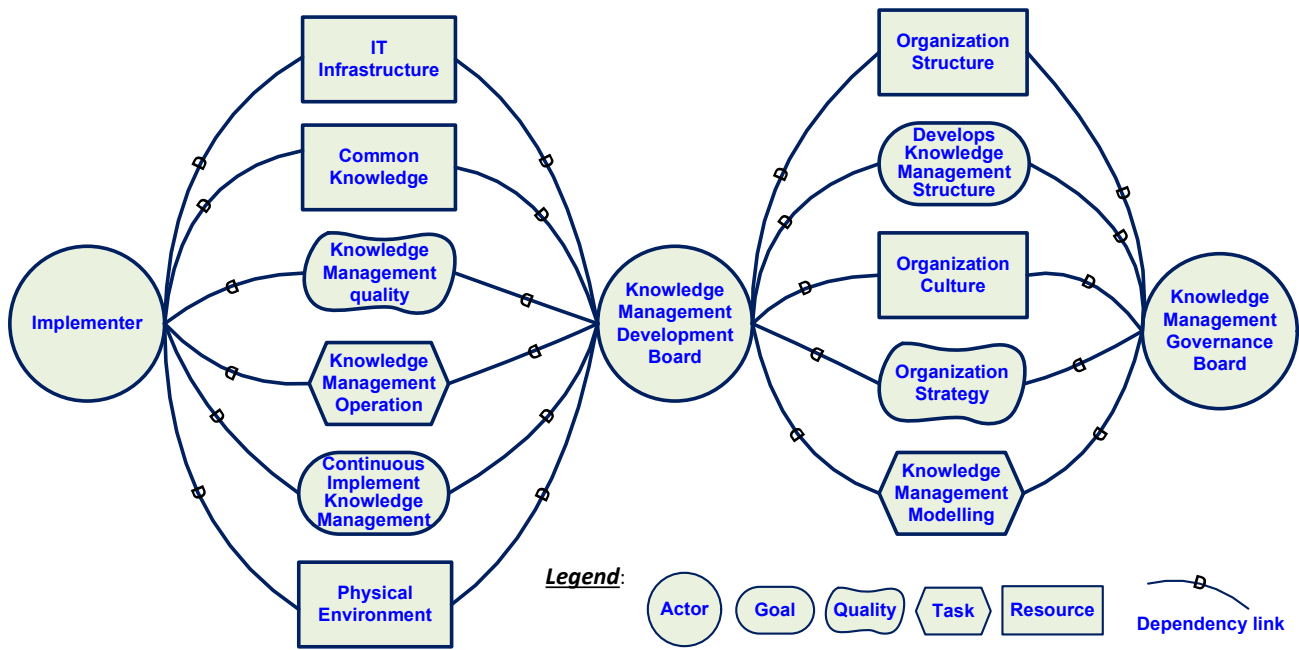


Fig. 3. GI-Tropos Knowledge Management Governance Strategic Dependency Model

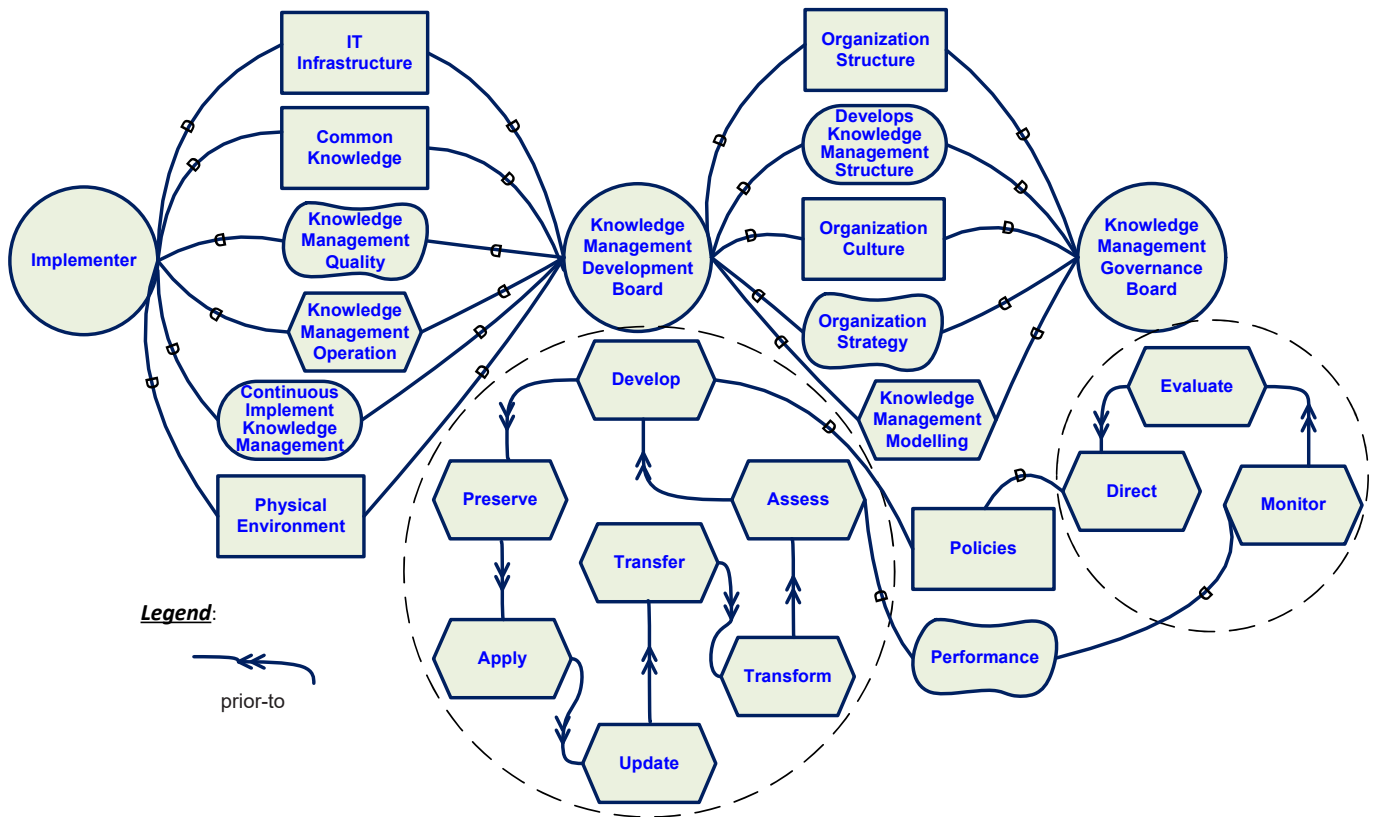


Fig. 4. GI-Tropos Knowledge Management Governance Strategic Rationale Model

factors). The scope of the KM governance decisions relevant for GI-Tropos is thus only software processes. The Knowledge Management Development Board allows aligning requirements-driven software processes with KM governance.

The Knowledge Management Development Board is thus a management board, not a governance one.

In the Strategic Rationale model, as depicted in Figure 4, the **Knowledge Management Governance Board** performs

three tasks (Evaluate, Direct, and Monitor) corresponding to the three governance core processes (Evaluate, Direct, and Monitor) respectively. The **Knowledge Management Development Board** performs seven tasks (Develop, Preserve, Apply, Update, Transfer, Transform and Assess) corresponding to the seven management core processes (Develop, Preserve, Apply, Update, Transfer, Transform and Assess) respectively. The **Preserve** task depends on the **Direct** task based on the **Policies** resource and the **Monitor** task depends on the **Assess** task based on the **Performance** quality.

#### IV. USE CASE

This section describes TransLogisTIC, a project used as a use case to illustrate the paper. This project developed a collaborative platform for outbound logistics. The purpose was to propose combined, performing and complete transportations in the Walloon Region with transport by rail particularly promoted in accordance with the European policy. The project involved several complementary actors including 10 private companies and 5 universities and research labs on a 3 year basis and a 14 million euro budget.

The TransLogisTIC project was re-built with I-Tropos models for validation purpose in the context of this work. We focused on validating knowledge management governance in the software development process of an online collaborative logistic platform allowing the major outbound logistics stakeholders (chargers, carriers, infrastructure managers and final clients,...) to share information for a better optimization of the logistic chain. Firstly, we defined the knowledge areas of the software development process (I-Tropos). Then, we explained reasons why this knowledge was required. Table I represents the knowledge areas in I-Tropos based on the Guide to the Software Engineering Body of Knowledge (SWEBOK, IEEE Professional Practices Committee [16]) and explains these reasons. Finally, we aligned knowledge management governance rules and constraints with GI-Tropos's governance processes (Evaluate, Direct, Monitor) based on the workflow proposed in Figure 5.

The workflow performed during the alignment of requirements-driven software processes with knowledge management governance rules and constraints is presented in Figure 5. These tasks are assumed by the Knowledge Management Governance Board in interaction with the Stakeholders and Knowledge Management Development Board. They are sorted in three groups (Evaluating, Directing, and Monitoring) corresponding to the three core governance processes (Evaluate, Direct, and Monitor) respectively. Each task is summarized as follow:

- **Consider:** Examine and judge current and future use of KM include strategy proposals; survey and categorize, analyze KM activities, elicit, codify and organize; consider the foundations of KM (KM infrastructure, KM mechanisms, and KM technologies); evaluate continuously; consider current and future business needs and objectives: competitive advantage and specific strategies; appraise and evaluate value of knowledge and KM.

TABLE I  
KNOWLEDGE NEEDED IN I-TROPOS

I-Tropos phase	Knowledge areas	Reason knowledge is needed
<b>Setting</b>	<ul style="list-style-type: none"> <li>• Software Requirements</li> <li>• Software Engineering Management</li> <li>• Software Engineering Process</li> <li>• Software Engineering Models and Methods</li> <li>• Software Quality</li> <li>• Software Engineering Professional Practice</li> <li>• Software Engineering Economics</li> <li>• Computing Foundations</li> <li>• Mathematical Foundations</li> <li>• Engineering Foundations</li> </ul>	<ul style="list-style-type: none"> <li>• To justify behind the generalized model, such as forces and trade-offs, success factors associated with practices;</li> <li>• To establish objectives, constraints, alternatives;</li> <li>• To evaluate product and alternatives;</li> <li>• To resolve risks.</li> <li>• To justify behind the task plan, including risk assessments, contingency plans, management goals and criteria.</li> <li>• To justify behind the system, including development goals and criteria, alternatives evaluated, and their evaluation.</li> </ul>
<b>Blueprinting</b>	<ul style="list-style-type: none"> <li>• Software Design</li> </ul>	<ul style="list-style-type: none"> <li>• To design systems and processes definitions.</li> </ul>
<b>Building</b>	<ul style="list-style-type: none"> <li>• Software Construction</li> <li>• Software Testing</li> </ul>	<ul style="list-style-type: none"> <li>• To implement systems and processes definitions.</li> </ul>
<b>Setuping</b>	<ul style="list-style-type: none"> <li>• Software Configuration Management</li> </ul>	<ul style="list-style-type: none"> <li>• To validate systems and processes definitions.</li> </ul>
<b>Operation</b>	<ul style="list-style-type: none"> <li>• Software Maintenance</li> </ul>	<ul style="list-style-type: none"> <li>• To make sure the commitments.</li> </ul>

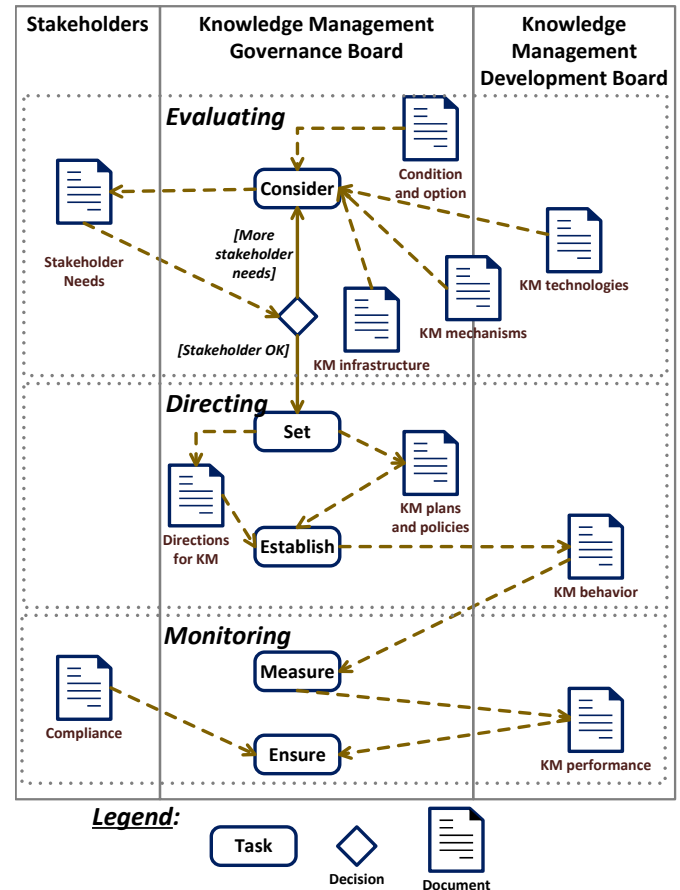


Fig. 5. GI-Tropos Knowledge Management Governance Workflow

- **Set:** Assign responsibility and direct preparation and implementation of KM plans and policies; set directions for KM.
- **Establish:** Establish sound behavior in knowledge use through policies; properly plan transition of project to operational status; encourage culture of good KM governance; direct submission of proposals identifying needs.
- **Measure:** Monitor and measure KM performance.
- **Ensure:** Assure that performance is in accordance with plans and business objectives; ensure that KM conforms with external obligations (regulatory, legislation, common law, and contractual) and internal work practices; handle, use, control, leverage, distribute and automate knowledge.

## V. CONCLUSION

For defining software systems methods and software project methodologies, governance can be viewed as evaluating, directing and monitoring software processes all along the life cycle. Governing knowledge management in requirements-driven software processes such as GI-Tropos enables coping with stakeholders' requirements and expectations. This paper aimed at specifying integration and alignment of knowledge management governance rules and constraints to requirements-driven software processes based on the software processes knowledge areas needed by governance.

The paper presents a new identification of critical moments in the software development life cycle for knowledge management governance since the main alignment objective was to deliver an efficient KM approach that meets stakeholders' needs and expectations. On the one hand, the strengths of GI-Tropos are to systematically offer structure and direction through the whole software processes governance and enable tailoring the process to the project needs. On the other hand, GI-Tropos also points out how to establish knowledge management governance rules to the software processes. Knowledge management can be governed in software development processes by a proper alignment performed on the software processes knowledge areas that need to be handled and KM governance processes themselves. Our proposed alignment specifies how to carry out these KM processes in the context of a collaborative software development life cycle.

Further work points to other additional practices that need to be integrated in this alignment to propose a complete framework taking into consideration, for instance, IT management, project management and agile practices [17]–[19] for managing the day-to-day activities and reacting to changing requirements and feedback. In addition, a CASE tool should be developed to help designing and implementing all the processes defined in this paper.

## REFERENCES

- [1] I. Sommerville, *Software Engineering*, 9th ed. USA: Addison-Wesley Publishing Company, 2010.
- [2] K. C. Desouza, "Barriers to effective use of knowledge management systems in software engineering," *Commun. ACM*, vol. 46, no. 1, pp. 99–101, Jan. 2003.
- [3] Y. Wautelet and M. Kolp, "Business and model-driven development of BDI multi-agent systems," *Neurocomputing*, vol. 182, pp. 304–321, 2016. [Online]. Available: <http://dx.doi.org/10.1016/j.neucom.2015.12.022>
- [4] A. Jashapara, *Knowledge Management: An Integrated Approach*, 2nd ed. Prentice-Hall, 2011.
- [5] I. Becerra-Fernandez and R. Sabherwal, *Knowledge Management: Systems and Processes*. Routledge, 2015.
- [6] A. Schroeder, D. Pauleen, and S. Huff, "Governance and leadership of knowledge management," in *Leadership in the Digital Enterprise: Issues and Challenges*. Idea Group Publishing, 2010, pp. 46–61.
- [7] S. Zyngier, "Knowledge management governance," in *The Encyclopaedia of Knowledge Management*. Idea Group Publishing, 2005, pp. 373–380.
- [8] V. H. A. Nguyen, M. Kolp, Y. Wautelet, and S. Heng, "Aligning Requirements-driven Software Processes with IT Governance," in *ICSOFT 2017 - Proceedings of the 12th International Conference on Software and Data Technologies, Madrid, Spain, 24-26 July, 2017*, 2017, pp. 338–345.
- [9] Y. Wautelet, M. Kolp, and S. Poelmans, "Requirements-driven iterative project planning," in *Software and Data Technologies - 6th International Conference, ICSOFT 2011, Seville, Spain, July 18-21, 2011. Revised Selected Papers*, 2011, pp. 121–135.
- [10] PMI, *A Guide To The Project Management Body Of Knowledge*. Project Management Institute, 2013.
- [11] J. Castro, M. Kolp, and J. Mylopoulos, "Towards requirements-driven information systems engineering: the Tropos project," *Inf. Syst.*, vol. 27, no. 6, pp. 365–389, 2002.
- [12] E. Yu, P. Giorgini, N. Maiden, and J. Mylopoulos, *Social Modeling for Requirements Engineering*. The MIT Press, 2011.
- [13] P. Kruchten, *The Rational Unified Process: An Introduction*, 3rd ed. Addison-Wesley, 2003.
- [14] J. Mylopoulos, M. Kolp, and P. Giorgini, "Agent-oriented software development," in *Hellenic Conference on Artificial Intelligence*. Springer Berlin Heidelberg, 2002, pp. 3–17.
- [15] Y. Wautelet, "A goal-driven project management framework for multi-agent software development: The case of I-Tropos," Ph.D. dissertation, Universite catholique de Louvain, 2008.
- [16] I. C. Society, P. Bourque, and R. E. Fairley, *Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0*, 3rd ed. Los Alamitos, CA, USA: IEEE Computer Society Press, 2014.
- [17] S. W. Ambler and M. Lines, *Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise*, 1st ed. IBM Press, 2012.
- [18] P. Kruchten, "Contextualizing agile software development," *Journal of Software: Evolution and Process*, vol. 25, no. 4, pp. 351–361, 2013. [Online]. Available: <http://dx.doi.org/10.1002/smr.572>
- [19] A. J. H. d. O. Luna, P. Kruchten, and H. P. d. Moura, "Agile governance theory: conceptual development," *CoRR*, vol. abs/1505.06701, 2015. [Online]. Available: <http://arxiv.org/abs/1505.06701>



# A Search-based Software Engineering Approach to Support Multiple Team Formation for Scrum Projects

Alexandre Costa

Federal University of Campina Grande  
Intelligent Software Engineering Group  
antonioalexandre@copin.ufcg.edu.br

Felipe Ramos

Federal University of Campina Grande  
Intelligent Software Engineering Group  
feliperamos@copin.ufcg.edu.br

Mirko Perkusich

Federal Institute of Paraiba  
mirko.perkusich@ifpb.edu.br

Arthur Freire

Federal University of Campina Grande  
Intelligent Software Engineering Group  
arthurfreire@copin.ufcg.edu.br

Hyggo Almeida

Federal University of Campina Grande  
Intelligent Software Engineering Group  
hyggo@dsc.ufcg.edu.br

Angelo Perkusich

Federal University of Campina Grande  
Intelligent Software Engineering Group  
perkusic@dee.ufcg.edu.br

**Abstract**—Search-Based software engineering (SBSE) deals with metaheuristic search-based optimization techniques to provide solutions for complex problems. A popular problem in literature is the team formation problem (TFP), which consists of finding the best allocation of human resources to a software development project. This problem is recognized as NP-hard and it is more complex in companies that carry out multiple projects. This paper presents an effective and automated approach to allocate multiple developers into multiple teams to maximize the technical compatibility between them. The approach consists of an SBSE method that uses Genetic Algorithm to simultaneously build multiple teams, using data from tag-based profiles. We conducted an empirical evaluation using data from eight real-world software projects of a Brazilian company. The results indicate that tag-based profiles is a promising information source to represent technical knowledge, since the suggested teams were considered to have the proper skills to the attend the technical demand of the projects. The approach was able to reach high levels of satisfaction, delivering teams in an effective and automated way. Although, further investigation needs to be conducted to reach stronger conclusions.

## I. INTRODUCTION

Search-Based Software Engineering (SBSE) is a field that deals with metaheuristic search-based optimization techniques to provide automated and semi-automated solutions for complex problems in Software Engineering (SE) [6], [5], [7]. According to Harman et. al. [7], typical SE problems that involve testing, design, requirements engineering, management, and refactoring can be well adapted for SBSE and have been successfully formulated as search-based optimization problems. For instance, the team formation problem (TFP), which consists of finding the best allocation of resources (developers) to a software development project. This problem has gain special attention over the last years and became a well-researched topic in the literature [1], [4], [11], [2].

The TFP is recognized as NP-hard [13] and it is more complex in companies that carry out multiple projects. It is

necessary to optimize the allocation of multiple developers with different sets of skills to multiple projects with different sets of requirements. The goal is to optimize each projects' chances of success considering the limited resources available and attempt to have all projects succeeding. The previously described scenario represents a multiple team formation problem (MTFP) [4].

Finding optimal teams brings special concern with agile methods, such as Scrum [10], a team-centered framework designed to deliver products with the highest possible value. In Scrum, the developer team is a self-organized and multidisciplinary group which means that, among others, the team must have the all the required technical knowledge to deliver the product. A nine-month field study of professional developers in a Scrum team [8] found that highly specialized skills and a corresponding division of work are the most important barrier for achieving effective teamwork. Therefore, it is necessary to make the most of the available resources, i.e., choosing right teams to the right projects.

In this paper, we present a SBSE approach to form multiple teams for Scrum projects. The approach is divided into two parts. First, we learn the technical knowledge of the developers through information extracted from the projects in which they have worked. For this purpose, we instrument the Scrum process by creating tag-based profiles for developers and projects. Therefore, we complement the Scrum framework with activities to assign tags to artifacts, i.e., the Product Backlog (PB), User Stories (US) and Technical Tasks. Second, we build a data structure to run a Genetic Algorithm to allocate developers to projects, forming teams with maximum technical compatibility, considering the projects demands and the developer's skills. We chose Genetic Algorithm, because it one of the most suitable methods for combinatorial optimization problems [3].

To evaluate our solution, we used real-world data from eight Scrum software projects and we collected data from

16 developers and 4 projects managers. The results indicates that the tag-based profiles are a useful information source to support the multiple team formation with a high level of satisfaction.

This paper is organized as follows. Section II presents related work on SBSE applied to MTFP. Section III presents our solution to mitigate the multiple team formation problem. Section IV presents the design and discusses the results of the empirical evaluation. Section V presents the validity threats; and Section VI presents the final remarks, limitations and future work.

## II. RELATED WORK

As mentioned before, a particular case in the TFP is the MTFP, which consists of simultaneously allocating multiple developers to multiple projects to maximize attributes from both developers and projects.

Palacios et. Al. [1] presented a recommender system designed to assist project managers in configuring multiple teams for work packages defined by Scrum Projects. The system is based on fuzzy logic, rough set theory and semantic technologies. In this work each project manager is responsible for building both project and developers profiles using competence baseline documents.

Strnad and Guid [11] proposed a decision support system based on Fuzzy Logic and Genetic Algorithm. The system uses fuzzification to automatically obtain fuzzy skill assessments from numerical data of an employee database. Genetic Algorithms are used to optimize the teams formations according to the projects requirements, which are obtained by a standard document specification inside company.

Silva and Costa [2] presented a framework based on dynamic programming to allocate human resources in multiple information system projects. The main goal is to determine the fit between the complete set of skills available from a candidate member of a project team and the skills required for that project to minimize the time required to complete a project demand.

Gutierrez [4] proposes an optimization model using a quadratic objective function, linear constraints and integer variables. The optimization model is solved by three algorithms: a Constraint Programming approach provided by a commercial solver, a Local Search heuristic and a Variable Neighborhood Search metaheuristic. Sociometric techniques are used to estimate performance characteristics such as productivity, training ability, leadership, efficiency, among others.

Ren et. al. [9] proposed a search-based software project method to build teams based on Cooperative Co-evolution. The teams are formed aiming to reduce the required time to complete the projects work packages.

The main difference between the cited works and ours, is the way the profiles of developers and projects are built. Most of the approaches depend on the knowledge and feeling of the project managers to build the profiles. For this reason, the results are subjective. In our approach, the profiles are derived from the technologies assigned to the Scrum artifacts

during the Scrum Instrumentation, minimizing the subjectivity. Another difference is that the profile information grows dynamically, because the instrumentation is refined throughout the project.

## III. PROPOSED APPROACH

The proposed solution is divided into two parts: Scrum Instrumentation and Solution Operationalization, which are detailed as follows.

### A. Scrum Instrumentation

Scrum is a framework for developing, delivering, and sustaining complex products [12]. It was designed to be complemented with technical and managerial processes as needed. We complement Scrum by applying tags to the artifacts to register the technical skills necessary to develop the features associated with them. These skills corresponds to programming languages, frameworks, platforms, APIs, architectures, databases, or any technology or technical knowledge necessary to develop a feature.

To lead the tag labeling process, a new role is proposed: (Scrum) Tagger. The Tagger is responsible for all the tag labeling process. To assure a standardized tags assignment, it is recommended that the company establish a small team of Scrum Taggers who should define specific rules to avoid that the same technical skill is rerepresented using different tags in different projects. The Scrum Instrumentation occurs throughout the project and includes the following labeling processes:

1) *Product Backlog Labeling*: the Tagger starts to work during the initial PB definition (Figure 1[1]). The PB is an ordered list of the product requirements. The Product Owner (PO) is the responsible for building the PB. Therefore, the Tagger conducts informal conversations with the PO to elicit all or at least most of the technologies necessary to develop the PB. Then, these technologies are converted into tags, forming the Backlog Tag set.

2) *User Story Labeling*: the labeling process occurs during Sprint Planning meetings (Figure 1[2]), which is an event where some requirements are selected from the PB to be developed during the given Sprint, defining the Sprint Backlog. This requirements are usually represented by User Stories. The Tagger, together with the Development Team, labels all the USs of the Sprint Backlog. For this purpose, the Tagger consults the team regarding the necessary technical skills to develop the USs. Then, the Tagger converts the skills into tags, forming the User Story Tag Set. It is important to note that this set is a subset of the Backlog Tag, therefore, if a new tag appears during this step the Backlog Tag set is updated.

3) *Task Labeling*: to define the Sprint Backlog, the USs are splited into Technical Tasks. When a task is done, the responsible selects a subset of tags from the User Story Tag set. The selected subset, called Task Tag set, represents the technical skills demanded to develop the feature represented by the task. This labeling process (Figure 1[3]) occurs throughout the Sprints. If necessary, the developer can suggest new tags,

which can be reviewed by the Tagger after the end of Sprint (Figure 1[4]). If new tags are created, the User Story Tag and the Backlog Tag sets are updated.

### B. Solution Operationalization

The second part of the proposed approach consist of creating the data structure necessary to run a Genetic Algorithm to allocate multiple developers into multiple projects. First, tag-based profiles are created for both developers and projects from the data acquired from the Scrum Instrumentation. Then, technical compatibilities between developers and projects are calculated using feature vectors. Lastly, we use weighted similarity coefficients to create a technical compatibility metric that is used in the Genetic Algorithm's fitness function to find the optimal team formation for each project. These steps are detailed as follows.

1) *Tag-based profile building*: The developer profile is defined from two sources of information: Curriculum and Development History. First, technical knowledge documented in the developers' curriculum is converted into weighted tags. The weight corresponds to the period of experience with the given technology. For example, if the developer worked with the Java programming language for 18 months, his profile will contain a tag *java(18)*. Second, during the Sprint, the developers works in tasks in which tags were applied during the Scrum Instrumentation. Over time, the developer profile is being formed by the tags that were applied to the tasks completed by him. These tags also have weight, in this case it is given by the number of times a tag appeared in a task. For example, if during the Sprint the developer completed five tasks with a java tag applied, his profile is filled with the weighted tag *java(5)*.

The project profile is built from the Product Backlog Tag set. As the developer profile, it grows throught the execution of the project, as the project requirements change and the product backlog items are more detailed.

2) *Feature vector*: A feature vector is a  $k$  dimensional vector composed by numerical values and represents a set of the object's characteristics. The feature vectors are created from tag-based profiles. They are necessary to calculate the similarity between profiles.

The project feature vector is called Backlog Vector ( $V_B$ ). Each tag presented in the project profile corresponds to an index in the vector. Note that different projects may have different profiles, consequently, they have different feature vectors.

For each developer, two feature vectors are created from his profile. The Curriculum Vector ( $V_C$ ) is built from the Curriculum Source; the Development Vector ( $V_D$ ), from the Development History Source. Both vectors use the same structure as the project feature vector to allow the similarity calculation.

3) *Developer Feature Vector Normalization*: Since the weight of the tags that compose the developer feature vectors come from two different data sources with different magnitudes, it is necessary to harmonize the scales by normalizing

it to values between 0 (zero) and 1 (one). The normalization uses the Equation 1, where  $x = (x_1, x_2, x_3, \dots, x_k)$  is the key value in the same position at each developer feature vector and  $z_i$  is the  $i_{th}$  normalized value. Note that there are two normalization processes, one for each developer feature vector.

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \quad (1)$$

4) *Similarity calculation*: We used Manhattan Similarity to calculate the similarity between two  $k$ -dimensional feature vectors. The similarities are used to determine the technical compatibility between the developer and project profiles. Between each profile, two different similarity coefficients are calculated. One represents the similarity between the Backlog Vector and Curriculum Vector and the other represents the similarity between the Backlog Vector and Development Vector. Before the similarity calculation, the Backlog Vector is filled with 1s (ones) to represent a maximum technical demand of a project, i.e., the developer feature vectors, which are composed by values closer to 1s (ones), will have higher values of similarity.

$$1 - \frac{\sum_{i=1}^k |u[i] - v[i]|}{k} \quad (2)$$

5) *Genetic Algorithm to Form Multiple Teams*: The multiple team formation process consists of allocating multiple developers into multiple project team to maximize the technical compatibility between the project demands and the developer skills. For this purpose, we propose a metric called Technical Compatibility Level (TCL). The TCL is generated based on a weighted similarity coefficient given by the Equation 3. The  $\alpha$  represents the weight of the similarity calculated between the Backlog and Curriculum vectors [ $sim(v_B, v_C)$ ]. The  $\beta$  represents the weight of the similarity calculated between the Backlog and Development vectors [ $sim(v_B, v_D)$ ]. For each project and developer, a TCL is generated.

$$TCL = \frac{\alpha \cdot sim(v_B, v_C) + \beta \cdot sim(v_B, v_D)}{2} \quad (3)$$

An example of the chromosome structure used in the Genetic Algorithm is presented in Figure 2. In this case, we have 3 projects and 12 developers. The genes represent the developers and, depending on his position in the structure, a different TCL is used during the fitness calculation. For instance, if a developer D1 appears in the indexes between 0 and 4, the TCL between D1 and the Project A is used. If D1 appears in the indexes between 5 and 7, the TCL between D1 and the Project B is used. The goal of the fitness function is to find a candidate solution, where the sum of the TCLs corresponded to each gene is maximized. Consequently, an optimal solution corresponds to a configuration where the matching represents maximum technical compatibility.

## IV. EMPIRICAL EVALUATION

To evaluate our approach, we conducted an empirical evaluation in a Brazilian software development company, in

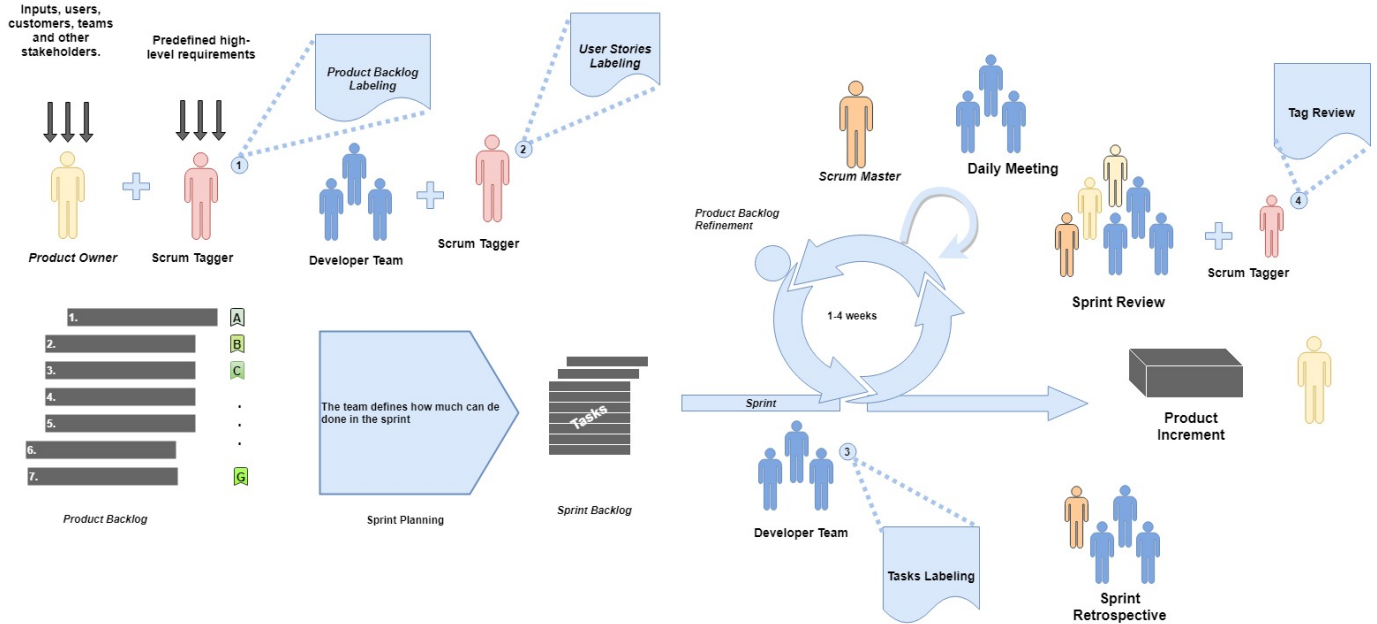


Fig. 1. Scrum Instrumentation

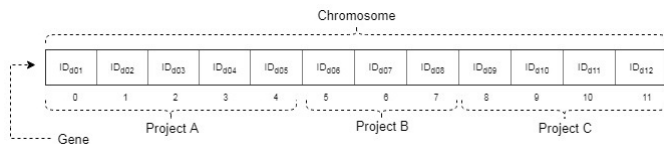


Fig. 2. Example of a chromosome used in the Genetic Algorithm

which we collected real-world data from eight Scrum software projects. Four of them (P1, P2, P3 and P4) were running during the evaluation and the remaining projects (P5, P6, P7 and P8) were already finished. Each of the running projects is composed by a project manager (PM) and four junior developers. The PMs know the profiles of all 16 developers, since there is a regular rotation among the projects, to avoid knowledge islands. The PMs have 2-5 years of experience in Scrum projects and the developers 1-3 years in software development.

In Table I, we show statistics regarding the projects. The first column contains the identification of the running projects; the second contains the number of Sprints executed in the project. Each Sprint lasted 2 weeks; the third presents the number of User Stories completed; the fourth is the number of tasks done (each task was performed individually); the fifth shows the identification of the team members; the sixth, contains the Backlog Tag set collected during the instrumentation process performed in the evaluation. For reasons of confidentiality, the projects and developers identifications were presented in an anonymous way.

We performed the instrumentation process with the four running projects (P1, P2, P3 and P4). The instrumentation was designed to be executed in a interactive and incremental

way during the Scrum Events and throughout the project. Unfortunately, the projects were already started and we had to apply the process at once to retrieve the previous data of the projects. For each project, we applied the three labeling processes described in Section III-A. To avoid overloading of the participants, which could compromise the process, we divided the labeling in turns. During the morning, the team performed the Backlog and User Story labeling processes and during the afternoon the Tasks' one. The instrumentation was applied in different days for each project. The Scrum Tagger role was performed by the first author of this paper. In Table I, we can see the Backlog Tag set of projects P1, P2, P3 and P4.

To create the Curriculum Source, we provided an online questionnaire and all developers were guided to respond to it. The questionnaire was composed by the tags present in the Backlog Tag set (Table I) and complemented with tags that represent popular technologies which not appeared during the instrumentation. Also, the developers were free to add other technologies he have worked outside the company by answering an open-ended question. Naturally, duplicated tags were excluded from the questionnaire. For each tag, the developers were asked to register their experience (in months) with the given technology.

To evaluate the approach, we simulated a scenario in which four projects (P1, P2, P3 and P4) were used to learn the profile of the developers with the goal of allocating them to other four projects (P5, P6, P7 and P8). We present the Backlog Tag set for projects P5, P6, P7 and P8 in Table II. To collect the data for projects P5, P6, P7 and P8, we only executed the Product Backlog labeling process.

We executed our approach varying  $\alpha$  and  $\beta$  weights and the results are presented in Table III. For each new project, we

TABLE I  
DATASET STATISTICS

Project	#Sprints	#User Stories	#Tasks	Team Formation	Backlog Tag set
P1	6	41	136	D3, D7, D9, D14	angular-charts, angular-material, angularjs, bower, Chart.js, checkstyle, css, ESLint, express, Firebase, Gitlab, gulp, html, javascript, jenkins, JSHint, json, Mocha, MongoDB, node.js, pmd
P2	4	15	97	D1, D2, D10, D11	angular-charts, angular-material, angularjs, bower, Chart.js, checkstyle, css, cytoscape, eslint, findBugs, gitlab, gson, hamcrest, html, java, javascript, jpa, JSHint, json, junit, maven, mockito, mySQL, PMD, spring-boot, spring-jpa, swagger
P3	3	12	63	D4, D5, D6, D8	angular-charts, angularjs, bootstrap, bower, chart.js, css, eclipse, gitlab, gulp, html, http-request, java, javascript, javascript, javax, json, maven, mysql, npm, spring-boot, spring-jpa, sts, swagger, webstorm
P4	7	21	76	D12, D13, D15, D16	android, angular-material, angularjs, http-request, Beaglebone, Bonescript, Bootstrap, C++, Chart.js, Css, Express, Gitlab, Heroku, Html, iot, jade, java, javascript, material-design, MongoDB, node.js, npm, parse-server, python, Raspberry, Socket.io, XML

TABLE II  
PROJECT'S BACKLOG TAG SET

Project	Backlog Tag set
P5	android, android-studio, Beaglebone, firebase, git, glide, http-request, iot, java, javascript, node.js, parse-server, postman, sqlite, xml, zxing
P6	android, java, xml, node.js, volley, firebase, butterknife, Body-parser, Json, Express, Mongoose, Nodemailer, Validator, webstorm, javascript
P7	Body-parser, node.js, bootstrap, css, express, Gitlab, html, http-request, javascript, JSHint, json, jwt, MongoDB, Mongoose, Nodemailer, npm, webstorm, git
P8	angularjs, Body-Parser, bootstrap, bower, css, cytoscape, ESLint, express, git, Gitlab, gulp, html, http-request, jasmine, javascript, jenkins, JQuery, JSHint, json, karma, material-design, Mocha, MongoDB, Mongoose, bootstrap, node.js, NPM, tslint, typescript, Postman

formed five teams and presented to the corresponding project managers. The PM1 was manager of P8, PM2 was manager of P7, PM3 was manager P5, PM4 was manager P6. Then, we asked them to answer a couple of questions. First, the PM was required to rank the five teams suggested, according to the level of suitability to the project, i.e., the better ranked teams should be those who possess the set of technical skill that best meets the demands of the project. Second, they should rate using five point Likert scale their satisfaction with the best ranked team(s) (i.e., multiple teams could be considered tied with the best rank).

In Figure 3, we show the results related to the first question. It indicates that when the weight of the Curriculum source of the developer is high the team is ranked in the best position by most of the managers. When we increase the weight of the Development History source the suggested teams start to be ranked in the last positions. Although the Curriculum may not be the most reliable information source, it has the bigger amount of data, since it represents the knowledge accomplished by the developers during all their professional journey. On the other hand, the volume of the Development History source is much smaller, because the data collected came only from the projects the developers were participating during the evaluation and it corresponded just to a few months

of development. This indicates that Curriculum data reflected most of the technical knowledge the developers claimed to have, at least by the PMs expectations. Unfortunately, we can not make solid conclusions about the Development History source, because of its small amount of data. We believe that, at a given point after collecting data, this source is more trustworthy than the Curriculum, but need to collect more data to verify this hypothesis. In Figure 4, we present the results related to the second question, which show that the PMs were satisfied with the allocations.

#### V. THREATS TO VALIDITY

We identified a few threats to validation in our work. As same as others agile methodologies, Scrum stands for individuals and interactions over processes and tools. We proposed a Scrum instrumentation as an incremental and interactive process. It is designed to be less intrusive as possible to Scrum framework, since it occurs during already existents events and demands just a few more steps. Although the process was carefully designed, we could not apply it as it is suppose to be, because the projects had already started. So, we applied the process at once in each project. We consider this to be an internal threat to validity.

Since we only collected data from one company for a short period of time, we cannot claim external validity. We aim to address this threat in future work.

#### VI. CONCLUSIONS

In this paper, we propose a SBSE approach to support multiple team formation for Scrum projects. Among our contributions, we can highlight the Scrum Instrumentation process, which allows the creation of tag-based profiles for developers and projects. These profiles can be used to assist technical knowledge management. Since, the instrumentation process is designed to be incremental and interactive, the profiles are susceptible to reflect changes during the project execution and grow gradually. We can also emphasize the creation of an automated method based on Genetic Algorithm to support the project managers during the simultaneously allocation of multiple developers into multiple software projects, forming teams with maximum technical compatibility.

TABLE III  
OBTAINED RESULTS FROM THE VARIATION OF  $\alpha$  AND  $\beta$  WEIGHTS

	P5	P6	P7	P8
$\alpha = 100\%$ and $\beta = 0\%$	D4, D12, D13, D15	D7, D8, D9, D14	D1, D2, D3, D6	D5, D10, D11, D16
$\alpha = 75\%$ and $\beta = 25\%$	D7, D12, D13, D15	D4, D8, D9, D14	D1, D2, D3, D6	D5, D10, D11, D16
$\alpha = 50\%$ and $\beta = 50\%$	D7, D12, D13, D15	D4, D8, D9, D16	D1, D2, D3, D6	D5, D10, D11, D14
$\alpha = 25\%$ and $\beta = 75\%$	D7, D12, D13, D15	D2, D4, D8, D16	D3, D5, D6, D14	D1, D9, D10, D11
$\alpha = 0\%$ and $\beta = 100\%$	D7, D12, D13, D16	D2, D4, D8, D11	D3, D5, D14, D15	D1, D6, D9, D10

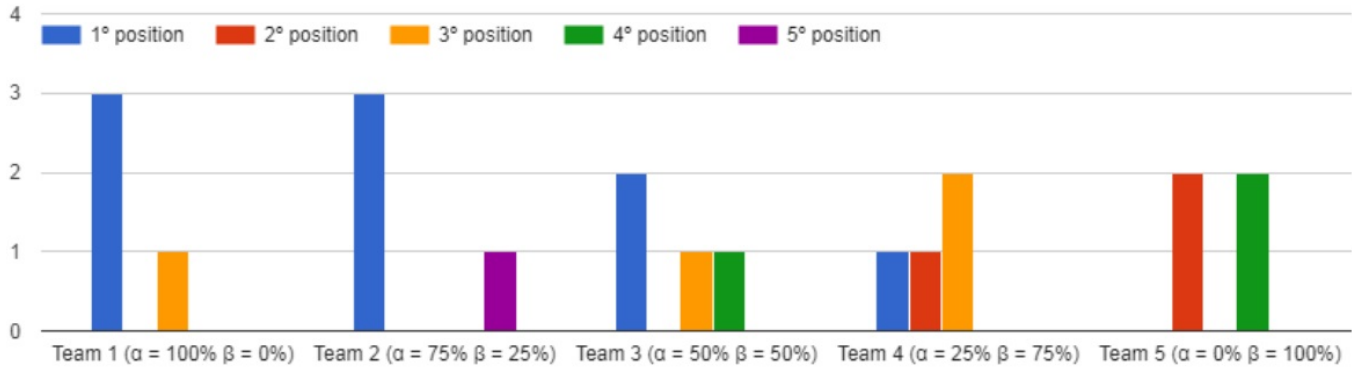


Fig. 3. Ranked teams by the projects managers

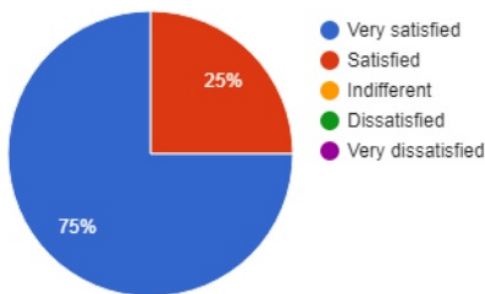


Fig. 4. Project managers satisfaction to the better ranked teams

As a limitation of our approach, we can point the slow growth of the Development History data source. Since this source depends on the developer participation in software projects of the company, the amount and diversity of these data might need time to reach the same size as the Curriculum source. We hypothesize that the Development History is a more reliable source, but we could not verify it in this study. We plan to verify it in future work, by increasing the number of projects and developers for the next empirical evaluation and test different scenarios. Also, we intend to determine optimum values for  $\alpha$  and  $\beta$  dynamically, according to the information volume of each data source. Furthermore, we plan to integrate our tagging mechanism to quality and productivity indicators, to have a more reliable regarding the knowledge (i.e., expected performance) of the developers.

## REFERENCES

- [1] R. Colomo-Palacios, I. González-Carrasco, J. L. López-Cuadrado, and Á. García-Crespo. Resyster: A hybrid recommender system for scrum team roles based on fuzzy and rough sets. *International Journal of Applied Mathematics and Computer Science*, 22(4):801–816, 2012.
- [2] L. C. e Silva and A. P. C. S. Costa. Decision model for allocating human resources in information system projects. *International Journal of Project Management*, 31(1):100–108, 2013.
- [3] E. Falkenauer. *Genetic algorithms and grouping problems*. Wiley New York, 1998.
- [4] J. H. Gutiérrez, C. A. Astudillo, P. Ballesteros-Pérez, D. Mora-Melià, and A. Candia-Véjar. The multiple team formation problem using sociometry. *Computers & Operations Research*, 75:150–162, 2016.
- [5] M. Harman. The current state and future of search based software engineering. In *2007 Future of Software Engineering*, pages 342–357. IEEE Computer Society, 2007.
- [6] M. Harman and B. F. Jones. Search-based software engineering. *Information and Software Technology*, 43(14):833 – 839, 2001.
- [7] M. Harman, S. A. Mansouri, and Y. Zhang. Search-based software engineering: Trends, techniques and applications. *ACM Comput. Surv.*, 45(1):11:1–11:61, Dec. 2012.
- [8] N. B. Moe, T. Dingsøyr, and T. Dybå. A teamwork model for understanding an agile team: A case study of a scrum project. *Information and Software Technology*, 52(5):480–491, 2010.
- [9] J. Ren, M. Harman, and M. Di Penta. Cooperative co-evolutionary optimization of software project staff assignments and job scheduling. *Search Based Software Engineering*, pages 127–141, 2011.
- [10] K. Schwaber and M. Beedle. *Agile software development with Scrum*, volume 1. Prentice Hall Upper Saddle River, 2002.
- [11] D. Strnad and N. Guid. A fuzzy-genetic decision support system for project team formation. *Applied soft computing*, 10(4):1178–1187, 2010.
- [12] J. Sutherland and K. Schwaber. The scrum guide. *The definitive guide to scrum: The rules of the game*. Scrum.org, 268, 2013.
- [13] G. J. Woeginger. Exact algorithms for np-hard problems: A survey. In *Combinatorial Optimization/Eureka, You Shrink!*, pages 185–207. Springer, 2003.

# Bug or Not Bug? Labeling Issue Reports via User Reviews for Mobile Apps

Haoming Li<sup>1</sup>, Tao Zhang<sup>1,2\*</sup>, Ziyuan Wang<sup>3</sup>

<sup>1</sup>College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China

<sup>2</sup>Key Laboratory of Network Assessment Technology, Institute of Information Engineering, CAS, Beijing 100093, China

<sup>3</sup>School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, China  
{heulaoyoutiao, cstzhang}@hrbeu.edu.cn, wangziyuan@njupt.edu.cn

\*Corresponding author: Tao Zhang

**Abstract**—A great number of mobile applications (apps) have been released to the market. Therefore, software maintenance for these apps become an important and challenging task. For each app, developers usually submit issue reports to report the bugs, the features, the questions, and other changes appearing in it. In the process of software maintenance, developers refer to the corresponding labels to decide which one should be fixed first. If the label of an issue report is “bug” which means the report describes a serious error, developers should fix the bug first. Otherwise, if the label is not “bug” (e.g., feature or question), developers can resolve it later. However, according to our investigation, 36.7% of issue reports in top-10 popular mobile apps are not labeled. In other words, there are not any labels in them. It is difficult for developers to decide which issue should be resolved preferentially. To resolve this problem, we propose a method to verify whether an issue report describes a bug or not by using user reviews. Developers usually extract useful information from user reviews to maintain mobile apps. In this work, we utilize  $tf \cdot idf$ , Word2Vec, and Microsoft Concept Graph (MCG) to compute the textual similarity between issue reports and user reviews related to real bug in order to find the issue reports which describe the real bugs. As a result, our approach with Word2Vec performs the best among three similarity metrics.

**Keywords**—issue report; user reviews; similarity metrics; mobile apps; software maintenance

## I. INTRODUCTION

Recently, the number of mobile devices such as smartphones and tablet computers have been produced for a wider group of people. These mobile devices result in the increase of the number of mobile applications (apps). Therefore, maintaining mobile apps is becoming important and challenging [1]. Fixing bugs is a core task in software maintenance activities [2]. Once a bug is found, a developer can upload an issue report to describe this bug. The detailed information can help developers fix it. However, we find that 36.7% of issue reports in top-10 popular mobile apps were not labeled as “bug” or others. In this situation, developers should verify whether these issue reports describe the real faults or not. For a large number of issue reports, it is a time-consuming work. Thus, it is necessary to develop an approach to automatically label the issue reports instead of manual verification.

In online app stores (e.g., Google Play Store, Apple Store), users can evaluate each app by using scores and post their reviews. These reviews are free-form text that may include important information such as bugs that need to be fixed by developers. The buggy user reviews can guide developers resolve bugs appearing in apps. Therefore, the bugs appearing in user reviews may be very close to the real faults reported in the issue report so that they can help to verify whether an issue report describes a bug or not. To the best of our knowledge, there was no any work to study the relationship between user reviews and issue reports in mobile apps.

In this paper, we propose an automated labeling approach to verify whether an issue report describes a bug or not. First, according to the suggestion mentioned in the literature [3], we choose user reviews that have less than 3 stars or lower because the reviews with few number of stars have high probability of describing bugs. Second, we use SURF [4], a popular review analysis tool, to automatically classify these user reviews into five categories: information giving, information seeking, feature request, *problem discovery*, and others. Next, we utilize natural language processing techniques to pre-process the user reviews in the category-*problem discovery* to build an index set. Third, we compute the similarity scores between each candidate issue report and the index set by using three popular similarity metrics that include  $tf \cdot idf$  [5], Word2Vec [6], and Microsoft Concept Graph (MCG) [7]. If the similarity is more than a threshold, the issue report is labeled as “bug”. Otherwise, the issue report does not describe a software fault.

We perform experiments on user reviews and issue report selected from 10 open source mobile apps on GitHub. As a result, the average F1 scores of our approach using  $tf \cdot idf$ , Word2Vec, and MCG achieve 56.1%, 61.4%, and 58.9%, respectively when the best threshold is set for each metric. The result denotes that our approach with Word2Vec performs the best among three similarity metrics.

We summarize the contributions of our work as follows:

- We *first* propose an approach to automatically label the issue reports with “bug” by computing the similarity scores between the issue reports and buggy user reviews.

- We perform our approach on top-10 popular mobile apps. The result shown that our approach with Word2Vec performs the best among three similarity metrics.

**Roadmap.** Section 2 introduce the background and motivation of our work. In Section 3, we detail the proposed automated labeling approach. Section 4 presents the experimental results. In Section 5, we show the limitations of this work and corresponding solutions. Section 6 introduces the related work. Section 7 concludes the paper and introduce the future work.

## II. BACKGROUND AND MOTIVATION

For each app, users can input free-text to comment it. These comments are called user reviews. Since some user reviews describe the real bugs, they can help us label the issue reports as “bug” or not. Fig. 1 shows the examples of four reviews in AntennaPod, which are collected from Google Play Store.

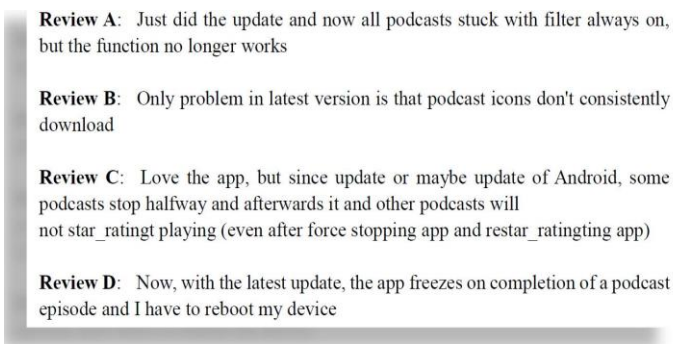


Fig. 1 User reviews in AntennaPod

From these user reviews, we note that users find the running problems of podcasts due to the update of the versions. They post their comments in Google Play Store and give the relatively low scores (*i.e.*, less than 4 stars). In fact, this is a bug appearing in AntennaPod. An issue report also describes it, we show it in Fig. 2.



Fig. 2 An issue report with the label- “bug”

We note that the developer reported a problem when he downloaded podcast files. The bug is also caused by the version

update. It is similar to the problems described in four reviews shown in Fig. 1, especially for Review B.

The label of the issue report-#544 is “bug”, therefore, the user reviews which describe the real faults have the close-knit relations with the issue reports which are labeled by “bug”.

According to our investigation for issue reports in top-10 popular mobile apps, we find that 36.7% of issue reports are not labeled (See Table I). In Fig. 3, we show an example of unlabeled issue report.



Fig. 3 An example of unlabeled issue report

In Fig. 3, we can see that the reporter did not label this issue report. In such a situation, a bug fixer should verify whether the issue report describes a bug or not so that he or she can decide the priority of resolving the issue. Obviously, this is a time-consuming task when he or she needs to resolve the increasing number of unlabeled issue reports.

When we carefully read the issue report-#237, note that the report describes a crash problem appearing in AntennaPod. In the other words, this is a bug. Thus, it should be fixed first. By analyzing the relationship between the user reviews shown in Fig. 1 and the issue report-#544 shown in Fig. 2, we think that buggy reviews can help to verify whether an issue report describes a bug or not. In the following sections, we introduce the details of the method and the corresponding experiment.

## III. RELATED WORK

### A. Software maintenance for mobile apps

Software maintenance for mobile apps become an important task due to an increasing number of mobile apps. However, only a few research teams study this problem. Syer et al. [12] analyzed 15 most popular open source Android apps, and they found that the “best practices” of existing desktop software development cannot be utilized for mobile apps due to the different features. Bhattacharya et al. [13] executed an empirical analysis of issue reports and bug fixing in open source Android apps. They analyzed the bug-fixing process and the quality of issue reports. Zhou et al. [14] conducted a cross-platform analysis of bugs and bug-fixing process in open source projects of different platforms such as desktop, Android, and IOS. They analyzed the different features such as fixing time and severity of bug-fixing process in these different platforms.

These studies on empirical analysis of issue reports and bug fixing process of mobile apps give us the inspiration for beginning this work. In our work, we not only analyze issue reports, but also analyze user reviews. In addition, we utilize



user reviews to label the issue reports which describe the real software bugs.

### B. User review analysis

In online app stores such as Google Play Store, Apple App Store, and Windows Phone App Store, users can rate the apps by selecting the stars from 1 (the lowest rating level) to 5 (the highest rating level) and inputting the reviews. These reviews describe users' impressions, experience, and preference degree. Therefore, they can be used by developers as a feedback to facilitate the process of software maintenance. Some studies focus on user review analysis to extract important information. Palomba et al. [15] traced informative crowd reviews onto source code change, and use this relation to analyze the impact of reviews on the development process. Ciurumelea et al. [16] analyzed the reviews and classify them. They also recommended for a particular review what are the source code files that need to be modified to handle the issue. Genc-Nayebi and Abran [17] presented the proposed solutions for mining online opinions in app store user reviews. Chen et al. [18] proposed an approach to identify attackers of collusive promotion groups in an app store by exploiting the unusual ranking change patterns from user reviews.

In our work, we not only analyze and classify user reviews, but also utilize the relation between user reviews and issue reports to automatically label the unlabeled reports as "bug" or "not bug".

## IV. METHOD

### A. Framework

In this paper, we propose an automated labeling approach to verify whether an issue report describes a real bug or not. Fig. 4 shows the framework of this method.

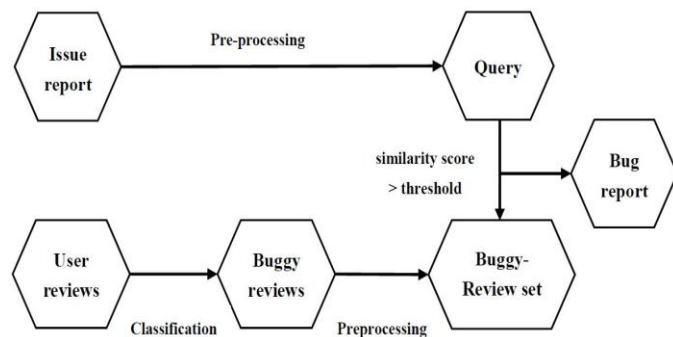


Fig. 4 The framework of automated labeling approach

From this figure, we first use Natural Language Processing (NLP) techniques to pre-process the issue reports in our data sets. These issue reports are treated as queries to be labeled. Second, we classify user reviews into five categories: information giving, information seeking, feature request, *problem discovery*, and others by using an automated review analysis tool. We extract the user reviews in *problem discovery* as buggy reviews according to the suggestion proposed in the literature [3]. Third, we also preprocess these buggy reviews to

produce the buggy-review set. Finally, we compute the textual similarity between each query and the buggy-review set by using three similarity metrics that include  $tf \cdot idf$ , Word2Vec, and MCG. If the similarity score is more than the threshold, we label this issue report to "bug".

In the following subsections, we show the details of each step.

### B. Document Preprocessing

As a first step, we preprocess the issue reports in our data set by using NLP technologies [8]. We use the python libraries NLTK<sup>1</sup> and TEXTBLOB<sup>2</sup> to implement the following steps:

- **Tokenization:** an issue report is divided into a bag of words (*i.e.*, tokens), which can be used to compute the textual similarity.
- **Stop word removal:** Some stop words such as "the", "a", "are" are common words that are frequent in written English. These words cannot provide more semantic information. Thus, they should be removed according to the list of WordNet English stop words.
- **Stemming:** the words should be transformed to their basic forms (*i.e.*, stems) in order to keep the high accuracy when we compute the textual similarity. For example, "providing" is changed to "provide", and "faults" is changed to "fault".
- **Nouns and verbs filtering:** we adopt a part of speech (POS) tagging classification to identify the nouns and verbs from issue reports. Since these words are the most representative, they are considered to compute the textual similarity scores.

### C. Review classification

In order to remove the uninformative reviews and find buggy reviews, we adopt SURF [4], a state-of-the-art review analysis tool, to classify them into five categories: Information Giving, Information Seeking, Feature Request, Problem Discovery, and Others. Because we focus on buggy reviews, we only collect user reviews in the category **Problem Discovery** to label the issue reports as real bugs.

To verify whether the classification is acceptable, we randomly select 20% of user reviews in the category **Problem Discovery**. The first author and one research assistant (RA)-Mr. Jiachi Chen from the Hong Kong Polytechnic University are responsible for checking whether each user review describes a real fault. The selected user reviews are equally divided into two groups. Each person is invited to check one group. In order to reduce the possible bias, two persons exchange their data to execute the verification again. The corresponding author can make a final decision when the verification results are inconsistent. As a result, we get the accuracy of 91.4%. Therefore, the classification results are acceptable.

### D. Structuring buggy-review set

When extracting the user reviews in the category **Problem Discovery**, we preprocess these reviews using the same

<sup>1</sup> <http://www.nltk.org/>

<sup>2</sup> <http://textblob.readthedocs.io/en/dev/>

approach described in Section III.B. Then they are grouped into buggy-review set to be used to compute the similarity scores with issue reports.

### E. Bug report verification

In our work, we propose an automated labeling approach to verify whether an issue report is a bug or not. To implement this purpose, we adopt buggy user reviews to verify bug reports by computing the textual similarity between issue reports and the buggy-review set via three similarity metrics that include  $\bullet$ idf, Word2Vect, and MCG. These metrics are introduced to transfer the documents to different kinds of vectors so that they can be input into cosine similarity function [9] to compute the similarity scores. These metrics are presented as follows:

**tf•idf**: it is a popular metric to represent documents as vectors of words. The value for each word is its tf-idf weight which is defined by:

$$\text{tf-idf weight} = \text{tf}_{t,d} \times \log \frac{N}{n_t}, \quad (1)$$

where  $\text{tf}_{t,d}$  is an appearing frequency of term  $t$  in the document  $d$ .  $\log \frac{N}{n_t}$  is the inverse document frequency which is a measure of how much information the word provides.  $N$  is the total number of documents while  $n_t$  is the number of documents which contain term  $t$ .

When we get all words' tf-idf weights, a document can be transferred to a vector of the tf-idf weights. Thus, we can use the cosine similarity function to compute the textual similarity between an issue report  $IR_i$  and buggy-review set  $BRS$ . It is defined by:

$$\text{sim}(IR_i, BRS) = \frac{\sum_{k=1}^n \omega_{ki} \omega_k}{\sqrt{\sum_{k=1}^n \omega_{ki}^2} \times \sqrt{\sum_{k=1}^n \omega_k^2}}, \quad (2)$$

where  $\omega_{ki}$  and  $\omega_k$  denote the weight of  $k^{\text{th}}$  word in  $IR_i$  and  $BRS$ , respectively. They are computed by *tf-idf weight* (see formula (1)).

**Word2Vec**: it maps a word into semantic word embedding. A large corpus of text can be transferred to a vector space, and each unique word in the corpus being assigned a corresponding vector in the space. We utilize Word2Vec with the skip-gram model [10]. In  $k$  dimensions ( $k=100$  in our work), each word can be represented as the vector defined as follows:

$$\overrightarrow{\text{vec}(\text{word})} = \langle v_1, v_2, \dots, v_k \rangle \quad (3)$$

Thus, a document can then be mapped into the space by:

$$C_s = \theta^T \cdot H^W, \quad (4)$$

where  $\theta^T$  is the vector of the *tf-idf weights* of the words in the document computed by formula (1) and  $H^W$  is the word vector matrix. In this matrix, the  $i$ -th line represents the word vector of the word  $i$ . The matrix is constructed by concatenating the word vectors of all words in the document. Via matrix multiplication, a document is transferred to a vector of semantic categories, denoted by  $C_s$ .

When we get the word vectors of the issue report and buggy-review set, we can use cosine similarity defined by formula (2) to compute their semantic similarity.

**MCG**: it maps text format entities into semantic concept categories with some probabilities. This similarity metric can also overcome the limitation in traditional token-based models such as  $\bullet$ idf that only compares lexical words in the document. It captures the semantics of words by mapping words to their concept categories. By using MCG, a word can be represented as its semantic concept categories with probabilities. For example, the word "Baidu", which can be categorized into a large number of concepts such as "company", "software", and "search". Therefore, a word can be transferred to a concept vector so that a document can then be mapped into the space by:

$$C_d = \theta^T \cdot H^M, \quad (5)$$

where  $\theta^T$  is the vector of the *tf-idf weights* of the words in the document computed by formula (1) and  $H^M$  is the concept matrix, which is constructed by concatenating the concept vectors of all words in the document. Via matrix multiplication, a document is transferred to a vector of concept categories, denoted as  $C_d$ . Actually, the document is mapped to the concept space by assigning a probability to each concept category to which the document belongs. This probability is estimated by summing up the corresponding probabilities of all the words contained in the document.

When we get the concept vectors of the issue report and buggy-review set, we can use cosine similarity defined by formula (2) to compute their semantic similarity.

When the similarity score is more than the threshold, we treat the issue report as the bug report. In other words, the issue report is labeled as "bug". Otherwise, the issue report's label is not bug.

## V. EXPERIMENT

### A. Setup of experiment

We collect the issue reports and the user reviews which have less than 3 stars or lower from 10 open source mobile apps in GitHub. Note that we treat the closed issue reports as the experimental object because they have the whole records of life-cycle so that it is easy to verify the experimental results. We first download top-100 popular open source mobile apps according to Ranking Repositories<sup>3</sup> in GitHub as our candidate projects, and then we filter out the projects which have less than 1400 reviews because a small number of user reviews can affect the result of automated labelling. The scale of our data set is shown in Table I.

In this data set, we note that there are 36.7% (2921/7966  $\approx 36.7$ ) of issue reports are not be labeled. Therefore, the goal of our experiment is to automatically label them using our approach described in Section III.

In order to easily verify our approach, we also utilize the proposed approach to predict the label "bug" for the issue report which had labeled as "bug". If the final list includes this issue report, the prediction is correct; otherwise, the prediction is wrong. For unlabeled issue reports, we first label them manually to verify the final result using our automated approach. The first author and Mr. Jiachi Chen who is a RA at the Hong

<sup>3</sup> <https://github-ranking.com/repositories>

Kong Polytechnic University are responsible for marking the unlabeled issue reports. They all have more than 3 years debugging experiences and are familiar with the projects in GitHub. One person is responsible for labeling half of unlabeled data while another person is responsible for labeling another half of data. Then they exchange their data each other. If the result is not consistent, a senior developer who has more than 10 years debugging experience and also has the experience to develop the projects managed in GitHub is invited to make the final decision for ensuring the reliability of the final result.

Table I The scale of our data set

Project	#reports	#unlabeled reports	#reviews	Period
AntennaPod	1,107	382	2,082	03.08.2012-21.12.2016
Automatic	222	44	1,394	18.01.2013-30.11.2016
cgeo	1,434	331	4,466	18.01.2011
chrislacy	193	22	1,471	
k-9 mail	783	337	4,456	15.03.2015-23.01.2017
OneBusAway	314	30	2,103	16.02.2013-28.06.2014
Twidere	653	245	2,031	06.07.2014-16.12.2016
UweTrottmann	399	192	4,459	26.07.2011-23.11.2016
WhisperSystems	1,524	1,170	4,443	26.12.2011-17.01.2017
WordPress	1,337	168	4,433	08.03.2013-14.01.2017
All	7,966	2,921	31,368	

We utilize F1 score [11] to evaluate our result. F1 is a frequently-used evaluation function, which is defined by:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}, \quad (6)$$

where Precision and Recall are computed by:

$$Precision = \frac{TP}{TP + FP}, \quad (7)$$

$$Recall = \frac{TP}{TP + FN} \quad (8)$$

Here, TP (*i.e.*, True Positive instances) indicates the number of instances (*i.e.*, issue reports in our work) labeled correctly, FP (*i.e.*, False Positive instances) represents the number of instances labeled incorrectly, and FN (*i.e.*, False Negative instances) shows the number of correct instances that are not labeled by the approach.

### B. Parameter adjusting

After we compute the similarity score between each issue report and the buggy-review set by using three metrics-tf•idf, Word2Vec, and MCG, the issue reports are labeled as “bug” when the scores are more than the defined threshold. Therefore, the threshold is the important parameter, which decides the performance of our approach. Table II shows the F1 scores of 10 projects when we select the different thresholds (0.1 to 0.9) using Word2Vec. We highlight the best values when selecting the corresponding threshold. Due to the limited space, we do not show the adjusting process of the thresholds for 10 projects by using other similarity metrics, *i.e.*, tf•idf and MCG. But we show them at our GitHub repository: <https://github.com/heulaoyoutiao/bugtag> in order to let each scholar easily reproduce our work.

### C. Experimental result

We adopt the best threshold to implement our approach. Table III shows the Precision, Recall, and F1 scores of all projects using three metrics.

From this table, we note that our approach using Word2Vec shows the best performance due to the highest Precision (47.26%), Recall (93.77%), and F1 scores (61.47%). Our approach using MCG shows the second-best performance while our approach using tf•idf shows the lowest performance.

We analyze the possible reasons for this evaluation result. tf•idf adopts term frequency and inverse document frequency, but it does not consider the term’s semantic concept. On the contrary, Word2Vec and MCG also preserve terms’ semantic and syntactic relationships. Therefore, our approach using Word2Vec or MCG performs better than that using tf•idf. Word2Vec performs the best may be caused by the characteristics of our data sets. We will deeply analyze the reasons in the future.

## VI. LIMITATION

We only collected the issue reports and bug reports from 10 mobile apps managed by GitHub to perform our experiments. These apps are selected according to Ranking Repositories in GitHub. Thus, our approach may not be generalizable to other projects. Even though we think that these popular projects are representative, we would like to further explore more projects in our future work.

Moreover, we only consider to automatically label the “bug” for issue reports by computing the textual similarity between buggy user reviews and issue reports. Machine learning techniques can also be utilized to do this task. In this future, we can consider to implement deep learning-based automated labelling approach for recommending more labels such as feature and question to unlabeled issue reports.

## VII. CONCLUSION AND FUTURE WORK

In this paper, we propose an approach to automatically label an issue report as “bug” or not. This approach considers the relationship between buggy user reviews and issue report. It computes the similarity scores between them using three metrics such as tf•idf, Word2Vec, and MCG. The experimental result shows that our approach with Word2Vec performs the best.

In the future, we will propose an approach and corresponding system to automatically recommend more labels (*e.g.*, feature, question) for issue reports. Moreover, we consider to utilize deep learning to tag these labels.

## ACKNOWLEDGEMENT

This work was supported in part by the National Natural Science Foundation of China under grant 61602258, the China Postdoctoral Science Foundation under grant 2017M621247, the Heilongjiang Postdoctoral Science Foundation under grant LBH-Z17047, and the Fundamental Research Funds for the Central Universities under grant HEUCFJ170604.



# A Topic Modeling Approach for Code Clone Detection

Sandeep Reddivari  
School of Computing  
University of North Florida  
Jacksonville, FL, USA 32224

Mohammed Salman Khan  
School of Computing  
University of North Florida  
Jacksonville, FL, USA 32224

## ABSTRACT

In this paper we investigate the potential benefits of Latent Dirichlet Allocation (LDA) as a technique for code clone detection. Our objective is to propose a language-independent, effective, and scalable approach for identifying similar code fragments in relatively large software systems. The main assumption is that the latent topic structure of software artifacts gives an indication of the presence of code clones. In particular, we hypothesize that artifacts with similar topic distributions contain duplicated code fragments. To test this novel hypothesis, we conduct an experimental investigation using multiple datasets from different application domains. Preliminary results show that, if calibrated properly, topic modeling can deliver satisfactory performance in capturing different types of code clones. It also achieves levels of accuracy adequate for practical applications, showing comparable performance to already existing tools that adopt different clone detection strategies.

## Keywords

Refactoring, Topic Modeling, Code Clones

## 1. INTRODUCTION

Code clones are similar code fragments that appear in a software system [10]. It is estimated that a typical mid-size industrial system contains up to 20% of duplicated code [4, 15, 24]. Clones are often produced by the *copy- $\&$ -paste* practice of programmers [16]. Rather than rewriting working code fragments from scratch, programmers prefer to copy, and perhaps slightly modify, working code that has already been tested before [9]. The main assumption is that, simply making a copy of a working code is faster and is less likely to introduce new bugs, especially when a deadline is approaching [29]. However, from a refactoring perspective, code clones are considered a major code smell [3, 10]. They significantly increase the maintenance cost and the error proneness of the code [14, 20]. For instance, inconsistent changes to code duplicates can lead to unexpected behavior [24]. Therefore, clones must be kept in sync during maintenance [21]. In particular, when a bug is fixed in an instance of a cloned code, all other duplicates must be altered as well. In addition, clones decrease the modularity of the system and its level of encapsulation, as well as unnecessarily increase the size of the code which can complicate future maintenance tasks and reduce understandability [9].

DOI reference number: 10.18293/SEKE2018-179

Therefore, code clones have to be refactored whenever detected [18, 30]. Based on the notion of similarity established between code fragments, various types of code clones can be identified. Clones can range from exact matches where the same code fragment is copied, to functional clones where two code fragments perform the same operation but they are syntactically and semantically different [30, 4, 19]. A plethora of clone detection tools have been proposed in the literature [8, 30]. Such tools often support a large variety of programming languages, and adopt different clone detection strategies, at different levels of complexity, designed to target various types of clones. Despite these advances, the usage of clone detection tools is still not pervasive in industry [14]. In general, most of these tools are still far from achieving optimal accuracy. This requires developers working with such tools to manually classify and verify the detected candidate clones [8, 30], a process that is often described as time-consuming and error-prone [25]. In addition, there is still a lack of adequate support for large-scale systems, where clones are likely to spread over several code modules [13, 21].

In an attempt to address these issues, in this paper we propose a novel approach, based on topic modeling, to facilitate a more accurate, language-independent, and scalable clone detection process. In particular, we experiment with Latent Dirichlet Allocation (LDA), the most commonly used technique for topic modeling in Natural Languages Processing (NLP) [7]. LDA is a probabilistic statistical approach for estimating a topic distribution over a text corpus [7]. Our main conjecture is that, the topic distribution over a code base gives an indication of the presence of code clones. Our contributions in this paper are the following. First, we propose an effective, language independent, and scalable approach for detecting code clones based on topic modeling. Second, we provide an experimental benchmark for calibrating LDA parameters and evaluating its performance in detecting various types of code clones.

The rest of the paper is organized as follows. Section 2 briefly introduces the background and related work. Section 3 introduces our research methodology. Section 4 presents our experimental analysis and results. Section 5 presents threats to validity and finally Section 6 concludes the paper and discusses the future work.

## 2. BACKGROUND AND RELATED WORK

### 2.1 Latent Dirichlet Allocation

LDA was first introduced by David Blei et al. [7] as a sta-

tistical model for automatically discovering topics in large corpus of text documents. The main assumption is that documents in a collection are generated using a mixture of latent topics, where a topic is a dominant theme that describes a coherent concept of the corpus’s subject matter. In recent years, LDA has been utilized heavily to aid several essential software engineering activities. For instance, Andrzejewski et al. [1] proposed an approach based on LDA to support statistical debugging tasks in software systems. Results showed that LDA-based approach outperformed existing methods for bug cause identification. In addition, Thomas et al. [31] used LDA to study source code evolution. Analysis showed that evolution caused by change activity was often reflected in the topic mixture of the system.

LDA takes the documents collection  $D$ , the number of topics  $K$ , and  $\alpha$  and  $\beta$  as inputs. Each document in the corpus is represented as a bag of words  $d = \langle w_1, w_2, \dots, w_n \rangle$ . Since these words are observed data, Bayesian probability can be used to invert the generative model and automatically learn  $\phi$  values for each topic  $t_i$ , and  $\theta$  values for each document  $d_i$ . In particular, using algorithms such as Gibbs sampling [28], an LDA model can be extracted. This model contains for each  $t$  the matrix  $\phi = \{\phi_1, \phi_2, \dots, \phi_n\}$ , representing the distribution of  $t$  over the set of words  $\langle w_1, w_2, \dots, w_n \rangle$ , and for each document  $d$  the matrix  $\theta = \{\theta_1, \theta_2, \dots, \theta_n\}$ , representing the distribution of  $d$  over the set of topics  $\langle t_1, t_2, \dots, t_n \rangle$ . Several methods have been proposed in the literature to approximate near-optimal combinations of LDA parameters ( $\alpha, \beta, K$ ) in source code. Asuncion et al. [2] and Oliveto et al. [26] proposed the usage of LDA to automatically capture traceability links in software systems. Maskeri et al. [23] proposed a human assisted approach based on LDA to extract business domain topics from source code.

## 2.2 Code Clones

Four types of clones can be identified based on the notion of similarity considered between code fragments [4, 19, 30]. These types include:

**Type I:** Exact clones in which the same fragment of code is copied without modification in its semantic or syntactic structure (except for spacing and comments).

**Type II:** Clones that are syntactically identical fragments except for slight variations, such as different identifiers names, literals, types or spacing.

**Type III:** Clones that have been slightly changed by added, removed or re-ordered statements, in addition to Type I and Type II modifications.

**Type IV:** Functional clones which refer to code fragments that perform similar operations but their syntactic and semantic structures are different.

Detecting different types of clones requires different levels of sophistication. While Type I and Type II can be relatively easy to detect using lexical-based techniques, other types (especially Type IV) require a higher level of complexity to match operationally identical code fragments.

## 3. RESEARCH METHODOLOGY

### 3.1 Datasets

To conduct our experimental analysis, we used four software systems from different application domains. Table 1 describes the characteristics of these systems including: the size of the system in terms of lines of source code (SLOC),

**Table 1: Experimental Datasets**

Dataset	VER.	CLS.	LANG.	SLOC	CLOC
<i>iTrust</i>	15.0	299	Java	20.7K	9.6K
<i>Apache Ivy</i>	2.3.0	451	Java	49.9K	16.7K
<i>QuantLib</i>	1.3.0	874	C++	178.8K	22.3K

lines of comments (CLOC), implementation language (LANG.) version (VER.) and number of classes (CLS).

### 3.2 Implementation and Tool Support

In this paper we use JGibbLDA, a Java implementation of LDA. This particular implementation uses Gibbs Sampling for parameter estimation and inference [12]. To integrate JGibbLDA in our analysis, a C# prototype is implemented upon the current Java implementation. We refer to this prototype as *CloneTM*, a code **Clone** detection tool based on Topic Modeling. Our interface provides options to tune the underlying LDA model ( $\alpha, \beta, K$ ), as well as visualization support for LDA results. For instance, stacked charts and bar charts are available for visually comparing topic distributions of multiple artifacts.

### 3.3 Dominant Topic Analysis

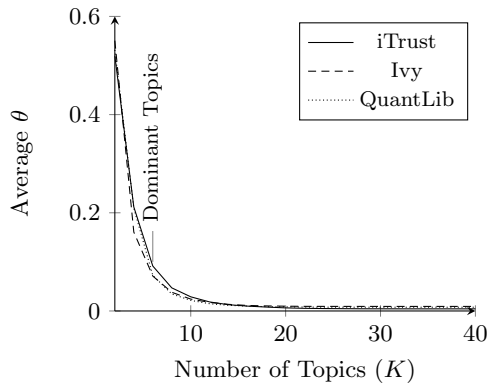
We start our analysis by investigating the effect of different values of  $K$  over the topic distribution generated for each artifact in each of our experimental systems. It is important to point out that the complexity of the study grows exponentially with the inclusion of other LDA parameters such as  $\alpha$  and  $\beta$ . Therefore, at this stage of our analysis, we fix the the values of these parameters. This strategy is often used in related research to control for such variables’ effect [11, 22, 31]. In particular, values of  $\alpha = 50/K$  and  $\beta = 0.1$  are used. These heuristics have been shown to achieve satisfactory results in the literature [12, 32].

One interesting observation of our topic analysis is that for each artifact  $d_i \in D$ , regardless of  $K$ , there is always a few number of topics that stand out from the rest of the topics in the document-topic matrix. Such topics have relatively larger  $\theta_i$  values. To demonstrate this effect, we run LDA using  $K=40$  over our *iTrust* experimental dataset. For each artifact  $d$  in the document-topic matrix of *iTrust*, we sort the 40 topics in a descending order according to their  $\theta_i$  value, so that topics at rank ( $r = 1$ ) have the highest  $\theta$  value. We then average these values for all artifacts (N) in *iTrust* over each rank  $r$ , producing  $A_r$  (Eq. 1).

$$A_r = \left(\frac{1}{N}\right) \sum_{j=0}^N d_j(\theta_r) \quad \forall r = 1, 2, \dots, 40 \quad (1)$$

The results are shown in Fig.1 which shows that in the topic distribution of each artifact, only a small portion ( $\approx 5$ ) of the topics has a  $\theta$  value larger than a certain threshold value ( $\lambda$ ). These topics with relatively large  $\theta$  values are known as dominant topics [22, 27]. In an attempt to specify  $\lambda$ , we conduct further empirical analysis over our open source experimental systems using different values of  $K$ . Results show that topic probability distribution for each artifact seems to always follow a regular distribution. In general,

<http://jgibbllda.sourceforge.net/>



**Figure 1: Average  $\theta$  values in the document-topic matrix arranged in a descending order ( $K=40$ ).**

three categories of topics, based on the empirically observed  $\lambda$ , can be identified as follows:

- $\lambda_1(\theta < 0.01)$ : Most of the topics in the document-topic distribution of each artifact fall under this category.
- $\lambda_2(0.1 > \theta \Rightarrow .01)$ : Dominant topics, an average of 4 to 8 topics for each document.
- $\lambda_3(\theta \Rightarrow > 0.1)$ : Absolute dominant topics, usually one or two topics are classified under this category.

We use these observations about dominant topic distributions to derive our main hypothesis in this paper. In particular, we assume that the presence of code clones can be reflected in the dominant topic distribution of software artifacts. Our main assumption is that documents sharing similar code fragments might also share a similar dominant topic distribution. Next we test these assumptions.

## 4. DETECTING CODE CLONES

To test the hypothesis, we devise an experimental benchmark to analyze the performance of LDA in detecting code clones. In particular, we manually inject code clones of Types I, II, and III in each of our experimental datasets. We exclude Type IV refactoring in this study. Manually injecting and verifying code smells for refactoring studies is a common practice in related research, especially in *proof-of-concept* studies [18, 6, 17, 8]. Also, since we are working with class granularity level, we limit our analysis in this paper to cross-class or cross-file clones.

Table 2 shows characteristics of our injected clones, including the number of clones injected of each type in each system (NO. C) and the number classes affected (CLS). Since *QuantLib* is a relatively larger system, we were able to inject more clones in it. Injecting Type I and II is a straightforward process. In particular, to produce Type I clones, a method call is simply replaced by the method itself, changes in spacing and comments are made. When injecting Type II clones, parameters names are changed. Injecting Type III clones was challenging as code statements have to be reordered, added, and removed. To achieve this, we apply random sequences of certain operation-preserving transformations into copied code fragments. These transformations include:

- Conditional Statements: Break and merge certain **if** and **while** statements into **if else** statements and vice versa. For example, the code segment:

```
If(validteUsrNm(uName) && validPwd(uPwd))
    return true;
```

can be broken down into:

```
If(validteUsrNm(uName))
    if(validPwd(uPwd))
        return true;
```

- Loops: Certain **for** statements were converted into **while** loops and vice versa. For example, the following loop statement:

```
for(line=br.readLine(); line!=null; line=br.readLine())
```

is transformed to the following **while** statement:

```
line = br.readLine();
while (line != null)
    line = br.readLine()
```

- Re-order: certain statements were reordered in such a way that does not change the structure of the code. For example, in the following code segment, variable `fBloodPressure` declaration can be moved above the method call `setPatientRecords(patientID)` without affecting the functionality of the code.

```
setPatientRecords(patientID);
float fBloodPressure = 0.0;
fBloodPressure = pm.getPressure(patientID)
```

The Second step is to define the notion of matching between dominant topics. In general, we follow a set-matching procedure, if any two classes have the same topic appearing in their set of topics with  $\theta_i > .01$ , we consider this case to be a candidate instance of code clones. We use the word “*candidate*” since we suspect that in some cases, matching also might happen without the presence of cloning. In that case we get a false positive. The procedure for our LDA-based clone detection technique can be described as follows.

```
Detect Clones: INPUT  $D, \alpha, \beta$ 
1.  $K = 40$ ;
2. Doc_Topic_Merix = Generate_Topic_Model( $D, \alpha, \beta, K$ )
3. FOR EACH  $d_i \in D$  IN Doc_Topic_Merix
4.   FOR EACH  $t_j \in d_i$ 
5.     IF  $\theta_j < .01$  THEN Remove  $t_j$ 
6. FOR EACH  $d_i \in D$ 
7.   FOR EACH  $d_j \in D$ 
8.     IF  $i \neq j$ 
9.       IF(Match (Doc_Topic_Merix( $d_i, d_j$ )) > 0)
10.      RETURN TRUE
11.  $K += 40$ 
12. GOTO 2
```

To assess the effectiveness of LDA in capturing instances of different types of clones. Standard recall and precision metrics of information retrieval are used. Such metrics are often used to assess the performance of clone detection tools [8, 6, 30]. Recall measures coverage and is defined as the percentage of clones that are correctly identified by the tool, and precision measures accuracy and is defined as the percentage of identified clones that are correct.

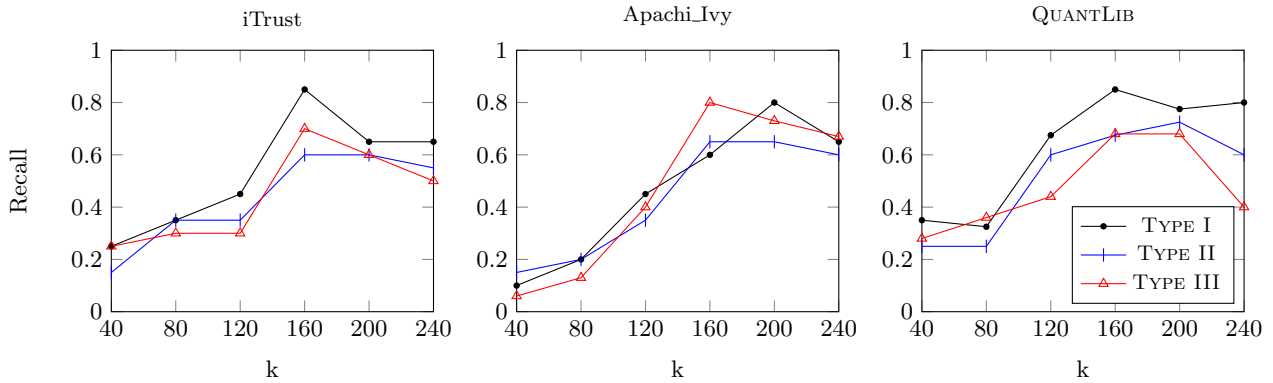


Figure 2: Recall values of different types of code clones at different values of K.

#### 4.1 Model Calibration

To run our analysis, we initially set  $K$  to be 40 topics. The document-topic distribution of each artifact in the system is then generated. A pair-wise comparison is conducted to capture matching in the latent topic structures of different classes. Results are then evaluated against the answer set using recall, or the number of injected clones the tool managed to identify. The value of  $K$  is then increased by 40 and the process is repeated. This particular step size has been found to yields noticeable changes in the recall values. We follow a hill climbing approach to monitor the changes in the recall. Our objective is to identify, or approximate, near-optimal  $K$  settings to detect clones. We tie optimality in this paper to recall. Therefore, in our analysis we emphasize recall over precision. The main rationale is that errors of commission (false positives) are easier to deal with than errors of omission (false negative). In other words, it is easier for a developer to discard instances that were misclassified as clones, rather than deal with clones that were not detected by the tool.

Since we have injected different types of clones, we were able to produce a separate recall curve for each type (cf. Fig 2). As for precision, a single precision chart, which shows the percentage of misclassified cases, is produced for each system (cf. Fig 3). The list of candidate clones generated for each system was scanned for already existing cross-file clones before injecting our clones, such clones were excluded from our precision calculations. We implemented our evaluation benchmark into *CloneTM*. Candidate clones are displayed to the user and the lines of code which include words from the matching topics are highlighted in the class view window of each class. Results show that in all three systems the recall seem to converge to a local maximum at the range of  $K = [160, 200]$  topics for all systems. The precision values also show satisfactory performance at this level. This can be explained based on the observation that at this range of  $K$ , topics tend to be more distinguishable from each other which makes this particular number of topics seem to be the most nearly optimal for code clone detection. However, at lower values of  $K$ , topics tend to have less density, generally spreading all over the class, and at higher  $K$  values (i.e.,  $> 200$ ) topics tend to be very specific, not able to cover code fragments with meaningful size. In general, the results show that our LDA-based approach was able to capture most types of clones, showing particu-

Table 2: Injecting Code Clones into Our Systems

Type	iTrust		Ivy		QuantLib	
	NO.	C	NO.	C	NO.	C
TYPE I	20	42	20	85	40	86
TYPE II	20	44	20	17	40	92
TYPE III	15	30	15	45	25	50

larly good performance in detecting Type I and Type III clones. Results also showed that the precision, while can be considered satisfactory, is still far from being optimal. To put the performance of *CloneTM* in perspective, we compare its recall and precision with other clone detection tools such as *CCFinder* [15] and *CloneDR* [5]. Table 3 shows the results of the tool comparison. For each type of clone in each system we compare the recall values. Results show that *CloneTM* is able to achieve comparable levels of recall to other tools in all systems. In particular, results show that our LDA-based approach managed to outperform *CCFinder* in Type III refactorings. Which can be explained based on the fact that the sequential analysis of code statements in *CCFinder* makes it fragile to statement reordering and code insertion. In general, many other token-based detection approaches do not detect clones with reordered statements [30]. However, the fact that LDA treats a class as a bag of words makes it immune to such changes. In contrast, results show that *CCFinder* was more successful in detecting Type II clones, this can be explained based on the fact that token-based methods are immune to name changing. On the other hand, LDA can be very sensitive to the information value embedded in the identifiers names and comments, so inconsistency in such information is expected to lower the accuracy. Results also show that, in comparison to *CloneTM* and *CCFinder*, *CloneDR* captured the smallest number of clones in all different types of clones. That might be explained based on the fact that this tool tends to do better in cross-method rather than cross-file clones detection [8].

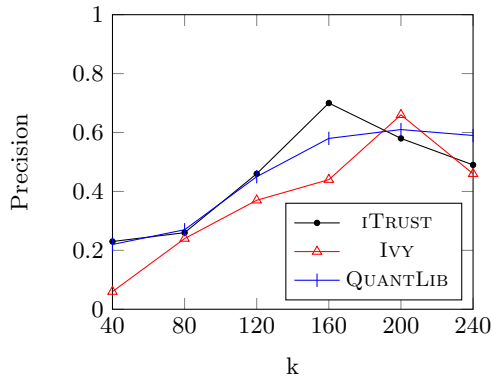
#### 5. THREATS TO VALIDITY

This study has several limitations that might affect the validity of the results. In terms of external validity, the results of this study might not generalize beyond the underlying experimental settings. For instance, only four systems



**Table 3: Comparing Recall Values of *CloneTM*, *CCFinder*, and *CloneDR***

System	Recall								
	TYPE I			TYPE II			TYPE III		
	<i>CloneTM</i>	<i>CCFinder</i>	<i>CloneDR</i>	<i>CloneTM</i>	<i>CCFinder</i>	<i>CloneDR</i>	<i>CloneTM</i>	<i>CCFinder</i>	<i>CloneDR</i>
iTRUST	0.85	0.9	0.2	0.6	0.8	0.2	0.7	0.4	0.13
IVY	0.8	0.8	0.2	0.65	0.85	0.15	0.73	0.67	0.27
QUANTLIB	0.775	0.7	0.35	0.575	0.7	0.25	0.68	0.53	0.32



**Figure 3: Precision values different values of K.**

were used in our analysis. Nevertheless, we believe that using four datasets from different domains, including a proprietary software product, helps to mitigate these threats. In fact, we believe that using these heavily-used, open source tools and systems increases the reliability of our results as it makes it possible to independently replicate our results. Other threats to the external validity might stem from specific design decisions, such as using heuristic values for  $\alpha$  and  $\beta$ . However, as mentioned earlier, due to the exponential complexity of the problem, it was not feasible to evaluate the effect of all LDA parameters in this study. In addition, the heuristics we used in our analysis have been proven to achieve satisfactory performance in related research.

Internal validity refers to factors that might affect the causal relations established in the experiment. A major threat to our study’s internal validity is the fact that we used manually injected clones to calibrate our model, in addition to manually verifying the candidate clones of different tools. This can lead to an experimental bias due to the subjectivity of this process. However, this particular experiment design decision was necessary to gain more insight into our procedure’s performance, in particular, its effectiveness in detecting different types of clones. In addition, as reported earlier, in the current state of research, human approval of the outcome of the code clone detection tool or method is inevitable.

## 6. CONCLUSION AND FUTURE WORK

In this paper we proposed a novel approach based on topic modeling for code clone detection. In particular, we investigated the potential benefits of using LDA to identify cross-class similar code fragments in Object Oriented software systems. We built our main research hypothesis upon

observations related to the latent topic structure of the software artifacts, and the effect code clones might have on that structure. In particular, we assume that matching on the dominant topic distribution between individual artifacts indicates cloning. To test our research hypothesis, calibrate, and evaluate our approach, we conducted an experimental analysis using four software systems from different application domains. We also compared the performance of our approach with other popular clone detection tools that adopt different clone detection strategies including, *CCFinder* and *CloneDR*. Results show that LDA can achieve satisfactory levels of recall, showing particularly good performance in detecting Type III clones that other related tools usually tend to miss. It also achieves levels of accuracy that can be adequate for practical applications. In the future, we plan to evaluate our approach by testing *CloneTM* using open source software systems to assess the usefulness and the scope of applicability of our approach. Also, we plan to fully implement our finding in the tool and provide visualization support to allow users to visually compare topic distributions of different classes as well as accept or reject candidate clones. We also investigate the potential effect of other code smells, such as God Class or Feature Envy, on the latent topic structure of software artifacts.

## 7. REFERENCES

- [1] D. Andrzejewski, A. Mulhern, B. Liblit, and X. Zhu. Statistical debugging using latent topic models. In *European conference on Machine Learning*, pages 6–17, 2007.
- [2] H. Asuncion, A. Asuncion, and R. Taylor. Software traceability with topic modeling. In *International Conference on Software Engineering*, pages 95–104, 2010.
- [3] L. Aversano, L. Cerulo, and M. Di Penta. How clones are maintained: An empirical study. In *European Conference on Software Maintenance and Reengineering*, pages 81–90, 2010.
- [4] B. Baker. On finding duplication and near-duplication in large software systems. In *Working Conference on Reverse Engineering*, pages 86–95, 1995.
- [5] I. Baxter, A. Yahin, L. Moura, M. SantAnna, and L. Bier. Clone detection using abstract syntax trees. In *International Conference on Software Maintenance*, pages 368–377, 1998.
- [6] S. Bellon, R. Koschke, G. Antoniol, J. Krinke, and E. Merlo. Comparison and evaluation of clone detection tools. *IEEE Transactions of Software Engineering*, 33(9):577–591, 2007.
- [7] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*,

- 3:993–1022, 2003.
- [8] E. Burd and J. Bailey. Evaluating clone detection tools for use during preventative maintenance. In *International Workshop on Source Code Analysis and Manipulation*, pages 36–43, 2002.
- [9] S. Ducasse, M. Rieger, and S. Demeyer. Language independent approach for detecting duplicated code. In *International Conference on Software Maintenance*, pages 109–118, 1999.
- [10] M. Fowler. *Refactoring: Improving the Design of Existing Code*. Addison–Wesley, 1999.
- [11] S. Grant and J. Cordy. Estimating the optimal number of latent concepts in source code analysis. In *International Working Conference on Source Code Analysis and Manipulation*, pages 65–74, 2010.
- [12] T. Griffiths and M. Steyvers. Finding scientific topics. In *The National Academy of Sciences*, pages 5228–5235, 2004.
- [13] Z. Jiang and A. Hassan. A framework for studying clones in large software systems. In *International Working Conference on Source Code Analysis and Manipulation*, pages 203–212, 2007.
- [14] E. Juergens, F. Deissenboeck, B. Hummel, and S. Wagner. Do code clones matter? In *International Conference on Software Engineering*, pages 485–495, 2009.
- [15] T. Kamiya, S. Kusumoto, and K. Inoue. CCFinder: A multilinguistic token-based code clone detection system for large scale source code. *IEEE Transactions Software Engineering*, 28(7):654–670, 2002.
- [16] M. Kim, L. Bergman, T. Lau, and D. Notkin. An ethnographic study of copy and paste programming practices in OOP. In *International Symposium on Empirical Software Engineering*, pages 83–92, 2004.
- [17] K. Kontogiannis. Evaluation experiments on the detection of programming patterns using software metrics. In *Working Conference on Reverse Engineering*, pages 44–54, 1997.
- [18] R. Koschke, R. Falke, and P. Frenzel. Clone detection using abstract syntax suffix trees. In *Working Conference on Reverse Engineering*, pages 253–262, 2006.
- [19] J. Krinke. Identifying similar code with program dependence graphs. In *Working Conference on Reverse Engineering*, pages 301–309, 2001.
- [20] B. Lague, D. Proulx, J. Mayrand, E. Merlo, and J. Hudepohl. Assessing the benefits of incorporating function clone detection in a development process. In *International Conference on Software Maintenance*, pages 314–321, 1997.
- [21] Z. Li, L. Shan, S. Myagmar, and Y. Zhou. CP-Miner: Finding copy-paste and related bugs in large-scale software code. *IEEE Transactions on Software Engineering*, 32(3):176–192, 2006.
- [22] Y. Liu, D. Poshyvanyk, R. Ferenc, T. Gyimóthy, and N. Chrisochoides. Modelling class cohesion as mixtures of latent topics. In *International Conference on Software Maintenance*, pages 233–242, 2009.
- [23] G. Maskeri, S. Sarkar, and K. Heafield. Mining business topics in source code using Latent Dirichlet Allocation. In *India software engineering conference*, pages 113–120, 2008.
- [24] J. Mayrand, C. Leblanc, and E. Merlo. Experiment on the automatic detection of function clones in a software system using metrics. In *International Conference on Software Maintenance*, pages 244–253, 1996.
- [25] E. Murphy-Hill and A. P. Black. Breaking the barriers to successful refactoring: Observations and tools for extract method. In *International Conference on Software Engineering*, pages 421–430, 2008.
- [26] R. Oliveto, M. Gethers, D. Poshyvanyk, and A. De Lucia. On the equivalence of information retrieval methods for automated traceability link recovery. In *International Conference on Program Comprehension*, pages 68–71, 2010.
- [27] A. Panichella, B. Dit, R. Oliveto, M. Di Penta, D. Poshyvanyk, and A. De Lucia. How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms. In *International Conference on Software Engineering*, pages 522–531, 2013.
- [28] I. Porteous, D. Newman, A. Ihler, A. Asuncion, P. Smyth, and M. Welling. Fast collapsed gibbs sampling for Latent Dirichlet Allocation. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 569–577, 2008.
- [29] F. Rahman, C. Bird, and P. Devanbu. Clones: What is that smell? In *MSR*, pages 72–81, 2010.
- [30] C. Roy and J. Cordy. A survey on software clone detection research. *Technical Report 541, School of Computing TR 2007-541, Queens University*, 2007.
- [31] S. Thomas, B. Adams, A. Hassan, and D. Blostein. Validating the use of topic models for software evolution. In *IEEE Working Conference on Source Code Analysis and Manipulation*, pages 55–64, 2010.
- [32] X. Wei and B. Croft. LDA-based document models for ad-hoc retrieval. In *ACM SIGIR conference on Research and development in information retrieval*, pages 178–185, 2006.

# XMILE - An Expert System for Maintenance Learning from Textual Reports

Eduardo Julião Máximo  
Programa de Pós-Graduação em Informática  
Aplicada  
University of Fortaleza - UNIFOR  
Fortaleza, Brazil  
ejmaximo@gmail.com

Vlândia Pinheiro  
Programa de Pós-Graduação em Informática  
Aplicada  
University of Fortaleza - UNIFOR  
Fortaleza, Brazil  
vladiacelia@unifor.br

**Abstract** - Software incidents are normally described in natural language (like English or Portuguese languages), because the users become free to express themselves about the incident. In this paper, we propose XMILE – an eXpert MaIntenance LEarning system based on NLP (Natural Language Processing) and machine learning techniques, that is capable of inferring the main attributes (type of intervention, maintenance action, cause and faulty zone) from textual reports of incidents. The XMILE was used on a real set of reports of maintenance incidents performed on IT systems of a Brazilian automobile enterprise, with excellent results in terms of precision and recall metrics.

## I. INTRODUCTION

Nowadays, the massive use of information systems and technologies (IT) requires a more specialized structure in IT support services and effective incident management. Incidents are some kind of unplanned interruption or reduction of the quality of an IT service, generating direct or indirect impact to the business. Initially, it is necessary that incidents be recorded or reported to a service center, generating software maintenance experiences databases. Several researches have proposed the reuse of experiences in Incident Management [1] and models such as the Information Technology Infrastructure Library (ITIL), contributing, for example, to the compliance with the rules of priority and time of attendance of the registered incidents, defined in Service Level Agreements (SLA).

The dissemination of Computerized Maintenance Management Systems (CMMS) has contributed to enriching Experiences Databases, which have been mainly used for traceability purpose. However, these bases could be processed in order to make explicit the “implicit knowledge”, in order to improve the decisions related to the maintenance activity. However, its extraction can hardly be done manually. In [2], the authors propose an original Experience Feedback process dedicated to maintenance, allowing capitalizing on past activities by formalizing the domain knowledge and experiences using Conceptual Graphs (CGs) [3]. The basis of the Experience Feedback process is an ontology that mainly allows modeling the maintenance interventions according to the three main parts of an experience: (i) *context*, that describes the general situation in which the event has

occurred (i.e. service order, functional localization of equipment involved, failure, technician); (ii) *analysis*, that presents the cause(s) of the problem; (iii) *solution*, that describes the type of intervention and the actions that have been performed for solving this problem.

However, software incidents are normally described in natural language (like English or Portuguese languages), because the users become free to express themselves about the incident. On the other hand, textual reports are a non-structured knowledge source, making the process of extracting information about the incident more difficult. Because of this, traditional CMMS use the form of description of maintenance experiences by attribute-value, whereby structured fields need to be informed manually by users or IT support technicians, making the process very time consuming.

In this paper, we propose XMILE – an eXpert MaIntenance LEarning system based on NLP (Natural Language Processing) and machine learning techniques, that is capable of automatically extracting information about the maintenance from incidents textual reports, inferring their main attributes. According to [2], we define four main attributes: type of intervention, maintenance action, cause and faulty zone. These attributes are required in order to respond the following question, respectively: in which conditions?, what?, why?, and where?. For example, for the incident report “*Please check for TGA and SG must be separate information and program assumes the same information*”, XMILE extracts: *Type of intervention* = “IT System”; *Fault Zone* = “System functionality” and *Cause* = “System Failure”.

A distinguishing feature of XMILE is the possibility of making explicit the information about maintenance cases (problems and solutions) described in natural language, for example, in the Portuguese language, in real time and without human intervention. The XMILE was used in a real set of reports on maintenance incidents performed on IT systems of a Brazilian automobile enterprise, with excellent results in terms of precision and recall metrics.

## II. BACKGROUND KNOWLEDGE

Our work is inspired in [2], which proposes a Experience Feedback Process consisting of two main modules that are presented in the next subsections.

## II.1 Experiences Database Structure

The basis of a knowledge-oriented representation of the experiences is an ontological knowledge, that can be considered as a rudimentary ontology [4] and is essentially defined as a couple (TC, TR), representing respectively the hierarchy of concept types and the hierarchy of relation types. This ontology is a high level and generic knowledge on the domain, specifying the vocabulary of the maintenance domain and the semantics of this conceptual vocabulary [5]. Since maintenance is a matter of communication between operators, maintenance actors and experts of various fields, a specific attention has been drawn to ontologies, ensuring that information/knowledge exchanged by different actors is meaningful, and that all the stakeholders interpret it in the same way [6]. An ontology is defined in [7] as “a formal, explicit specification of a shared conceptualization”. The “formal” aspect allows guaranteeing that the ontology is machine-readable [8]. The components of an ontology should thus allow formalizing the experience-knowledge in a specific domain.

TC has three main components of an experience: context, analysis and solution. In TC, the “context” part describes the general situation in which the event has occurred (i.e. Work Order (WO), functional localization of equipment involved, failure, technician); the “analysis” part presents the cause(s) of the problem; finally, the “solution” describes the type of intervention and the actions that have been performed for solving this problem (i.e. selected maintenance activities). TR expresses the basic relations of generic ontologies that will be used here, like “temporal” relation (i.e. before, after, parallel), “spatial” relation (i.e. in, out), “logic” relation (e.g. implies), “usual” relation (i.e. object, agent, involve, etc.) [9], as well as other specific relations of the domain of study, such as “experience relation” (i.e. generates, requires) or “element of”. These relations allow to link the different concepts types in the representation of an experience. An example is that a *Context* “**require**” an *Analysis*, which “**generate**” a *Solution*. In the *Context* we have that a **WorkOrder** has an “**object**” **FunctionalLocalization** and “**concern**” to a **Technician**. An *Analysis* is described by the concept **Cause**. Finally, the *Solution* graph defines that a **TypeOfIntervention** “**concern**” a **MaintenanceAction**.

## II.2 Rules Database Generation

This module applies a data mining process on the Experience Databases in order to discover association rules. An association rule is formally defined as a relation between two attributes of experience X and Y (antecedent and consequent) contained in the Experience database, based on conditional probability  $P(X|Y)$  and  $P(X \cap Y)$ . The association rules are selected according thresholds. After the association rules generation, the authors propose a semi-automatic process for assessment and interpretation of the rules.

## III. XMILE - A EXPERT SYSTEM FOR MAINTENANCE LEARNING FROM TEXTUAL REPORTS

The reports of software incidents and maintenance results are normally described in natural language, by

phone, e-mail, chat, or by tools for IT services monitoring, such as SpiceWorks<sup>1</sup>, OTRS (Open Ticket Request System)<sup>2</sup>, GLPI (Gestionnaire Libre de Parc Informatique)<sup>3</sup>, among others. As an IT service management needs the experiences databases in order to reuse the experiences and to improve the decisions related to the maintenance activity, we propose XMILE, an expert system based on NLP and machine learning techniques that automatically extracts the main attributes of textual incident and maintenance reports, generating knowledge in software maintenance. Initially, in this section, we define the ontology of the XMILE system and, after, we detail its architecture.

## III.1 XMILE Ontology

Inspired on the generic model of an experience, defined by [2], we instantiate this model for IT maintenance experiences. An *IT Experience* consists of three parts: *Context*, *Analysis* and *Solution*. In the *Context* part, the new concepts are: (1) *User* - represents the user that report the incident or the user that usually operates the IT system, and is defined by the following attributes: (a) *Admission Date*, representing the date of the admission of the user in the company or function; and (b) *Last Training Date*, representing the date of last training that the user participated. These attributes aim to capture the user experience in the use of the system, as we believe that the user experience impacts on the number of incidents; (2) *Text Report* - represents the textual report of software incident or software maintenance.

We propose the following types of *Fault Zone*: (a) *Equipment* - any hardware device that, connected directly or indirectly to a computer, adds new functionality or whatever is required for a task; (b) *Net access* - infrastructure (hardware, software and protocols, configuration) that enables LAN, WAN or Wi-Fi network connectivity; (c) *System functionality* - software or function of the IT system that performs the functional requirements of the users.

In the *Analysis* part, the causes of an *IT Experience* are: (a) *Lack of Maintenance* - indicating that there was failure to perform preventive maintenance on computers, printers, networks, and other devices; (b) *Lack of Training* - difficulty of the user to understand how the IT System work and how to reach its objectives; (c) *IT System Failure* - interruption of one or more tasks due to errors in IT system operating; (d) *IT System Error* - interruption of one or more tasks due to errors in IT system coding; (e) *Lack of Permission* - lack of permission granted to the user to produce or extract some information from the IT system; (f) *Communication Problem* - interruption in the provision or exchange of information on the internal network or with the Internet.

In the *Solution* part, the new concepts and instances are:

*Type of Intervention* - (a) *Infrastructure* - components and services (hardware and basic software) that provide the basis for sustaining all the information systems of an organization; (b) *IT System* - Automated or manual model of processes that use information technologies and

<sup>1</sup> <https://www.spiceworks.com>, accessed 03/14/2018

<sup>2</sup> <https://www.otrs.com>, accessed 03/14/2018

<sup>3</sup> <http://glpi-project.org>, accessed 03/14/2018

that are responsible for collecting and transmitting data that are useful for the development of products or services of companies, organizations and other projects; (c) Technical Support – Intellectual (knowledge), technological (hardware or software updates) and material, for the purpose of solving technical problems; (d) User Support – clarification of doubts, complaints, requests for services or support in solving problems.

**Maintenance Action** – (a) IT System update – application of security patches, configurations, features, and other new or revised items that will change the current system; (b) IT System correction – adjustment in programs where you change the system default behavior; (c) IT Training onsite – acquisition of knowledge, skills and competences as a result of vocational training or teaching practical skills related to specific useful skills; (d) Infrastructure repair – installation, monitoring and updating of servers, printers and other devices in order to maintain the high availability and normal operation of the system.

### III.2 The XMILE System

Figure 1 presents the architecture of the XMILE system with three components: Pre-processing, NLP and Machine Learning.

#### Pre-Processing Component

In the Pre-Processing component are performed the text mining and cleaning, data selection and data transformation. In order to prepare the data in an adequate format for the next NLP and Machine Learning components, we adopt here a data structure with the past experiences databases, based on the ontological knowledge (see Section III.1). This data structure is a triplet  $D = (O, I, R)$ , in which  $D$  is the database,  $O$  is a set of objects or transactions (i.e. each maintenance experience),  $I$  is a set of attributes (i.e. concepts defined in the XMILE ontology) and  $R \subseteq O \times I$  is a binary relation between  $O$  and  $I$ . Thus, each maintenance experience  $O$  in  $D$  is represented by a set of concepts in  $I$  related by binary relations in  $R$ . It's important to note that a text of an incident report  $t \in T$  is related with the concept *Text Report* in  $I$ . Next, we detail each phase:

- **Text Cleaning and Mining** - in this phase, the text is mined and cleaned by NLP processors like tokenizers and lemmatizers, then, stopwords and special characters are removed, and all letters are turned in uppercase. This phase aims at improving text quality, especially because the users and technicians write in an informal incomplete way. This is done by using NLP techniques that transform the tokens (word or expression) into their lemmas or canonical forms.
- **Data Selection** – in this phase, the values of each object in  $O$  is selected from the databases and associated to a concept in  $I$ . For example, given the incident “*Please check for TGA and SG must be separate information and program assumes the same information*”, reported by the `userId = 9874`, which has data of admission = 09/12/2007, and was trained last time 10/04/2014. So, we will have the following concept and values: `User.admissionDate = 09/12/2007`; `User.lastTrainingDate = 10/04/2014`

- **Data Transformation** – in this phase, the same data is transformed and normalized. For example, if in the original database, the date was in long format, we can transform in short format.

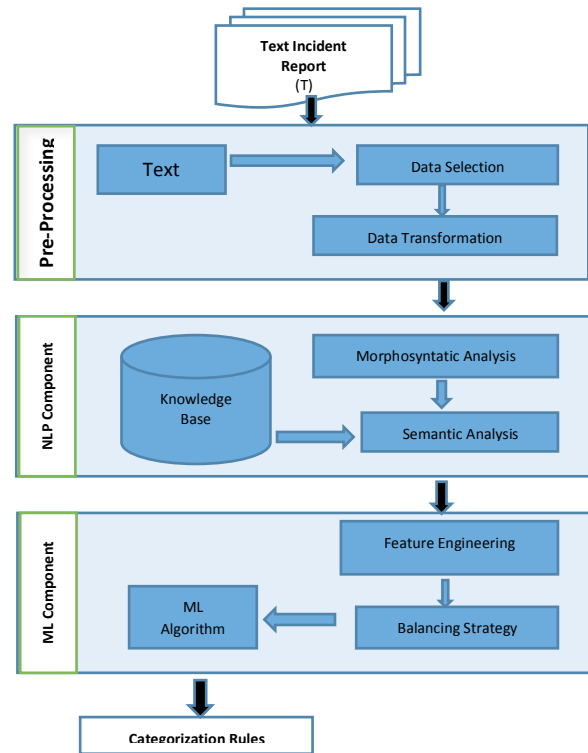


FIGURE 1. THE XMILE'S ARCHITECTURE.

#### NLP Component

- **Morphologic Analysis** - this component performs the morphological analysis or shallow parsing of the texts. Morphological analysis consists of the separation of sentences and grammatical classification (noun, verb, adjective, adverb, etc) and their modifiers of gender, number, time and verbal mode. Shallow parsing is applied to facilitate and normalize the search for word meaning (disregarding verbal, gender and number variations). In this component, we use a parser according to the language (e.g Freeling parser [10], for Portuguese Language, and Stanford Core NLP Toolkit [11], for English Language).
- **Semantic Analysis** - this component was developed to enrich the vocabulary with synonyms from a knowledge base like WordNet [12] or InferenceNet [13].

#### Machine Learning Component

This component is responsible for the calculation of the features (time of use of the equipment, time of the user in the organization, and quality of the internet/intranet connectivity), the balancing of the training and test sets, and the execution of Machine Learning algorithms, in order to learn rules for the categorization of attributes: Intervention type, Faulty zone, Cause and Maintenance Action.

#### IV. EXPERIMENTAL EVALUATION

In this experimental evaluation we want to verify the performance of XMILE in order to extract the information about the incident from textual report (in Natural Language).

In order to develop a golden standard, we selected 2819 textual incidents reports related to IT systems of a Brazilian company of automobile. Two IT technicians annotated each incident report with one of the **Type of Intervention** and **Cause** of the incident. In this work, we focus on these attributes because they represent the more important information to be extracted from incidents report, related to the Analysis and Solution of an Experience Database.

We defined two evaluation scenarios:

- SCENARIO 1 – we send to the Machine Learning algorithms only the bag of words of the textual reports (lemma of the relevant words)
- SCENARIO 2 – we send to the Machine Learning algorithms the bag of words and the following features: (i) time of use of the equipment; (ii) time (in years) of the user in the function; (iii) quality of connection of the Intranet or Internet.

In both scenarios, were executed three ML algorithms - Random Forest, J48 and Naïve Bayes, with 10-cross fold validation and the balancing strategy SMOTE [14]. Table 1 and 2 present the results in terms of F1-Measure (harmonic average between precision and recall) for the attribute Type of Intervention and Cause, respectively.

TABLE 1. RESULTS FOR ATTRIBUTE TYPE OF INTERVENTION (F1-MEASURE)

	J48	NAIVE BAYES	RANDOM FOREST
<b>SCENARIO 1</b>			
Infrastructure	0.907	0.849	<b>0.914</b>
IT System	<b>0.896</b>	0.869	<b>0.896</b>
Technical Support	0.858	0.780	<b>0.878</b>
User Support	0.862	0.782	<b>0.863</b>
<b>Average</b>	<b>0.881</b>	<b>0.820</b>	<b>0.888</b>
<b>SCENARIO 2</b>			
Infrastructure	0.907	0.811	<b>0.912</b>
IT System	<b>0.900</b>	0.791	<b>0.900</b>
Technical Support	0.855	0.795	<b>0.865</b>
User Support	0.855	0.785	<b>0.862</b>
<b>Average</b>	<b>0.880</b>	<b>0.796</b>	<b>0.885</b>

TABLE 2. RESULTS FOR ATTRIBUTE CAUSE (F1-MEASURE)

	J48	NAIVE BAYES	RANDOM FOREST
<b>SCENARIO 1</b>			
Lack of Maintenance	0.723	0.670	<b>0.772</b>
Lack of Training	0.742	0.646	<b>0.764</b>
IT System Failure	0.746	0.789	<b>0.773</b>
IT System Error	<b>0.739</b>	0.702	0.738
Lack of Permission	0.803	0.701	<b>0.821</b>
Communic Problem	0.926	0.769	<b>0.942</b>
<b>Average</b>	<b>0.777</b>	<b>0.726</b>	<b>0.799</b>
<b>SCENARIO 2</b>			
Lack of Maintenance	0.712	0.670	<b>0.743</b>
Lack of Training	<b>0.750</b>	0.684	0.740
IT System Failure	<b>0.773</b>	0.722	0.744
IT System Error	0.696	0.700	<b>0.726</b>
Lack of Permission	0.787	0.712	<b>0.804</b>
Communic Problem	0.916	0.809	<b>0.917</b>
<b>Average</b>	<b>0.775</b>	<b>0.715</b>	<b>0.777</b>

According to the results, we can observe that the best ML algorithm is the Random Forest with F1-measure = 0,888 and 0.799, respectively. An interesting result is that the additional knowledge about the context (additional features about time in the function, equipment age, and quality of connectivity) does not influence the results.

#### V. CONCLUSION

In this paper, we propose XMILE – an expert system based on Natural Language Techniques that extract automatically the main attributes from a textual report that describes a software incident. We evaluate XMILE in a set of 2819 textual incidents reports related to IT systems of a Brazilian company of automobile and the expert system achieved 89% and 80% (f1-measure evaluation metric) for the main attributes – Type of Intervention and Cause of an incident. As future works, we intend to evolve XMILE to the other attributes of the ontology and to verify additional features that can improve its performance.

#### REFERENCES

- [1] BEZERRA, G.; Pinheiro, V.; ALBUQUERQUE, A. Incident Management Optimization through the Reuse of Experiences and Natural Language Processing In: 9th International Conference on the Quality of Information and Communications Technology (QUATIC), 2014, 2014, Guimaraes.
- [2] Ruiz, P Potes, FOGUEM, B Kamsu, GRABOT, BERNARD (2014). Generating knowledge in maintenance from Experience Feedback. Knowledge-Based Systems 68 (2014) 4–20.
- [3] J. Sowa, Conceptual Structures: Information Processing in Mind and Machine, Addison-Wesley Publishing Company, Reading, MA, 1984.
- [4] M. Chein, M.L. Mugnier, Graph-Based Knowledge Representation: Computational Foundations of Conceptual Graphs, Springer Publishing Company, Incorporated, 2008.
- [5] F. Fürst, F. Trichet, Axiom-based ontology matching, Expert Syst. 26 (2) (2009) 218–246.
- [6] M. Uschold, M. Grüninger, Ontologies: principles, methods and applications, Knowl. Eng. Rev. 11 (2) (1996) 93–136.
- [7] T.R. Gruber, Toward principles for the design of ontologies used for knowledge sharing, Int. J. Hum.–Comp. Stud. 43 (5–6) (1995) 907–928.
- [8] R. Studer, V.-R. Benjamins, D. Fensel, Knowledge engineering: principles and methods, Data Knowl. Eng. 25 (1–2) (1998) 161–197.
- [9] J.Breuker, A cognitive science perspective on knowledge acquisition, Int. J. Hum.–Comp. Stud. 71 (2) (2013) 177–183.
- [10] Padró, Lluís e Evgeny Stanilovsky. 2012. Freeling 3.0: Towards wider multilinguality.
- [11] C.D. Manning, M. Surdeanu, J. Bauer, et al. The Stanford CoreNLP Natural Language Processing Toolkit. In Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, 2014.
- [12] Miller, G. A. (1995). Wordnet: a lexical database for english. Communications of the ACM, 38(11):39–41.
- [13] V. Pinheiro, T. Pequeno, V. Furtado and W. Franco. InferenceNet.Br: Expression of Inferentialist Semantic Content of the Portuguese Language. In: T.A.S. Pardo et al. (eds.): PROPOR 2010, LNAI 6001(90–99). Springer, Heidelberg, 2010.
- [14] N.V. Chawla, K.W. Bowyer, L.O. Hall and W.P. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. JAIR 16, 2002, pp. 321–357.

# Effort Estimation in Agile Software Development: an Updated Review

Emanuel Dantas, Mirko Perkusich, Ednaldo Dilorenzo, Danilo F. S. Santos, Hyggo Almeida, Angelo Perkusich  
Intelligent Software Engineering (ISE) Group, Federal University of Campina Grande (UFCG)  
Campina Grande, Paraiba, Brazil, 58429-140  
{emanuel.filho, mirko.perkusich, ednaldo.dilorenzo} @ifpb.edu.br, {danilo.santos, hyggo, perkusich} @embedded.ufcg.edu.br

**Abstract**— One of the main issues of an agile software project is how to accurately estimate development effort. In 2014, it was published a Systematic Literature Review (SLR) regarding this subject. The authors of this SLR analyzed works from 2001 to 2013 and reached the number of 25 relevant papers. Therefore, the goal of our work is to provide an updated review of the state of the art based on this reference SLR work. We applied a Forward Snowballing approach, in which our seed set are the former SLR and its selected papers. We identified changes in this new review comparing it with the reference SLR: XP methodology was mentioned in just a few works; Use Case Points (UCP) method and Case Points as size metric were not found. We also observed a strong indication of solutions based on Artificial Intelligence and Machine Learning methods for effort estimation in Agile Software Development (ASD). Finally, we identified that in the reference SLR there is a gap in terms of agreement on suitable cost drivers. Thus, in our updated review, we applied Thematic Analysis in the selected papers and identified a representative set of 10 cost drivers for effort estimation.

**Keywords**- Agile Software Development; Effort Estimation; Forward Snowballing.

## I. INTRODUCTION

In Agile Software Development (ASD), planning is carried iteratively. Project scope is continuously refined and prioritized following principles of Just In Time (JIT) management. According to Silva et al. [17], effort is one of the most important factors to prioritize requirements and features in ASD. It is also important to negotiate the scope of releases with the stakeholders.

Effort estimation in Agile Software Development (ASD) is an active research area. In 2014, a Systematic Literature Review (SLR) [20], in which data from 25 papers reporting 20 studies were analyzed and aggregated, was used to describe the state of the art related to estimation techniques, effort predictors and applied to ASD. The authors concluded that there were several gaps in the literature, such as the low level of accuracy of the techniques and little consensus on appropriate cost drivers.

Since 2014, the scientific community has been very active on the area of effort estimation in ASD. For instance, in 2015 Lenarduzzi et al. [10] proposed a mechanism to improve effort accuracy using functional size metrics.

On the other hand, in 2016 Grapenthin et al. [5] concluded that annotating the risks associated with user story during planning poker increases estimation accuracy.

There are several literature reviews published in the scientific community about effort estimation [2, 13, 16]. However, it is clear that the theme continues to be challenging and a subject of further studies given the difficulty of finding accurate solutions to the problem. In this context, the objective of this article is to present an updated overview of the state of the art on effort estimation in the context of ASD. For this purpose, we applied the Forward Snowballing technique [4] to find out relevant studies since the reference review of Usman et al. [20].

As contribution of our review, 24 new relevant papers were selected. Some findings from the reference review remain actual, but other questions have been raised in our research. In special, a significant amount of these new works have used techniques of Artificial Intelligence or Machine Learning to support effort estimation in ASD, which contributed to better estimation accuracy.

Another important implication of our review was the identification of an increasing use of cost drivers during effort estimation. Cost Drivers are personal or project factors that influence the value of estimates. Usually the works use different nomenclatures to represent the same factor. Based on this, in our research we used a Thematic Analysis approach [7] to map these factors.

The remainder of this paper is organized as follows. The section II presents more details of works related to effort estimation and the section III discusses the research method. The section IV presents our findings, and section V discusses the results of our research. The section VI has our final remarks, discussing potential future works.

## II. RELATED WORK

In this section, we presented more studies and details related to field of effort estimation. Sehra et al. [16] present a research of software estimation methods. This research evaluated 1178 papers between 1996 and 2006, many contributions were cited, but did not present specific findings for companies that use agile methods.

DOI reference number: 10. 18293/SEKE2018-003

Some works presented evidences that effort estimation is a task critical for project planning [13, 15], especially in agile software. These studies focused on methods of estimating effort in ASD. However, these researches did not explore the levels of accuracy of the approaches and how the use of cost drivers could be used to solve the problem.

Bilgaiyan et al. [2] indicated that computing techniques could be used to solve the problem of the effort estimation in ASD projects. After reviewing the literature, they found works that use techniques like Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Artificial Neural Network (ANN), and Fuzzy Inference Systems (FIS). However, no evidence of benefits of these techniques was presented and whether the methods were validated in the industry.

The systematic review published in 2014 by Usman et al. [20] investigated works from 2001 to December 2013, resulted in a complete state of the art guide on effort estimates in ASD. In another work [19] the same authors presented results of a survey representing the state of the practice.

Based on these related works, the new analysis presented in our article proposes an update of the state of the art review presented in Usman et al. [20], and differs from these related works in the following ways:

- The context of our study is ASD, whereas some of these reviews do not restrict this scenario;
- Most of these reviews focused solely on effort estimation methods, whereas ours also focused on the predictors (costs drivers) used in effort estimation;
- Most of these reviews do not bring information about data validation, whereas our work informs the domain, accuracy metrics and accuracy level achieved;
- Comparing to Usman et al. [20], effort estimation has been a relevant topic in ASD and they only evaluated papers until 2014. So, there is a need for an update.

### III. REVIEW METHOD

According to Kitchenham and Charters [8], systematic review is an evidence-based technique that uses a well-defined, unbiased and repeatable methodology to identify, analyze and interpret all the relevant papers related to a specific research question, subject area, or phenomenon of interest. It has been used to explore the state of the art of several areas such for ordering the product backlog [18], software requirements prioritization [1] and metrics [9].

The reference study [20] described an extensive SLR of peer reviewed studies focusing on effort estimation in ASD and followed the guidelines developed by Kitchenham and Charters [8]. Since our goal is to update it, we applied the Forward Snowballing approach [4] following the guidelines presented in Wohlin [22]. In our update, we followed the same research questions from the reference SLR and used the same inclusion and exclusion criteria in the evaluations.

### A. Research Questions

The following research questions (RQ) were investigated:

**RQ1:** What techniques have been used for effort or size estimation in ASD?

**RQ1a:** What metrics have been used to measure estimation accuracy of these techniques?

**RQ1b:** What accuracy level has been achieved by these techniques?

**RQ2:** What effort predictors (size metrics, cost drivers) have been used in studies on effort estimation for agile software development?

**RQ3:** What are the characteristics of the dataset/knowledge used in studies on size or effort estimation for agile software development?

**RQ4:** Which agile methods have been investigated in studies on size or effort estimation?

**RQ4a:** Which development activities (e.g. coding, design) have been investigated?

**RQ4b:** Which planning levels (release, iteration, current day) have been investigated?

### B. Search strategy

The first step of the snowballing involves the identification of a set of studies as a starting point (seed set) [6]. In the context of updating SLRs, key studies already exist, and should be the results of the previous SLR [4].

All papers selected in the reference SLR were submitted to the procedure of Forward Snowballing. In this process, we used Google Scholar<sup>1</sup> and Scopus<sup>2</sup> to analyze all the citations of these papers. Forward snowballing is conducted by examining papers published in the interval of the 2014 to December 2017. In the reference SLR [20] inspected works until December 2013.

The cited papers were forwarded to the study selection phase (Figure 1). It is important to note that citations to books, dissertations and theses were not considered. Initially, a basic evaluation is performed by analyzing only paper's title and abstract. Papers that pass this stage go to a selection of advanced evaluation where every paper is read. The analysis is performed by two reviewers who evaluate the paper according to the criteria for inclusion and exclusion (see criteria in [20]).

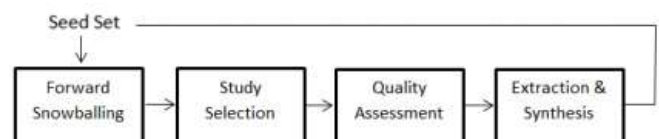


Fig. 1: Review Steps

Finally, papers categorized as relevant after the advanced evaluation are subjected to a procedure of quality assessment and data extraction. It is important to emphasize that

<sup>1</sup> <https://scholar.google.com.br>

<sup>2</sup> <https://www.scopus.com/>



snowballing is an iterative process, that is, at the end of an evaluation cycle, a new one is started using as seed set the resulting papers of the previous cycle. The process ends when no more new citations are found.

#### IV. RESULTS

This new review produced new evidences, accounting for changes in the practice and research in effort estimation in ASD, and also reinforces some of the results of the reference SLR, increasing the general confidence on its findings. In this section we describe the details of the whole review process and the results for each of the research questions.

After applying Forward Snowballing, the total number of found works was 262. Then, we performed an initial discard and reached the number of 120 relevant works. In this process we removed dissertations, theses, books and duplicate papers. Table 1 describes the number of studies of the different stages of the update review.

The next steps are the basic and advanced evaluation of the 120 selected papers. In the basic, two reviewers evaluated the title and abstract in accordance with the criteria for inclusion and exclusion. Only 36 papers followed for advanced evaluation.

TABLE 1. No. Of Papers during Snowballing 01

Papers	Search
a. Search results	262
b. After initial discard	120
c. After basic evaluation	36
d. After advanced evaluation	24
e. Excluded on low quality score	04
f. Final Papers Snowballing 01 (d-e)	20

After full paper reading in the advanced review, only 24 papers were compliant with the inclusion and exclusion criteria according to reviewers. Among those, 04 did not achieve the required score in quality assessment. In the end, after first cycle of the snowballing, we ended up with 20 selected papers.

The papers from the first evaluation were submitted to a new process of Forward Snowballing. The resulting papers from each evaluation phase of the second cycle are described in Table 2.

TABLE 2. No. Of Papers during Snowballing 02

Papers	Search
a. Search results	50
b. After initial discard	20
c. After basic evaluation	11
d. After advanced evaluation	06
e. Excluded on low quality score	02
f. Final Papers Snowballing 01 (d-e)	04

This was the last cycle of this study, since the papers of this second cycle did not have news citations. After adding papers from the last cycle to our database, we achieved the number of 312 papers analyzed, where 24 papers compose the resulting state of the art update.

Out of the 24 selected papers, conference proceedings provided 16 papers (62,5%) and journals 8 articles (37,5%) of relevant studies, these proportion range similar to those obtained in the reference SLR. We observed that publications about effort estimation have been growing in recent years. There were 04 papers published in 2014, the same for 2015, while in 2016 and 2017 there were 08 papers in each year.

For more details, we created a webpage<sup>3</sup> to provides a summary of the information extracted from each paper. It is important to note that some of these papers are from the same authors and might represent a single study. As a result, we had a total of 24 papers mapped to 15 primary studies. In the following subsections we presented the results of the extracted data related to our study’s research questions.

##### A. (RQ1) Estimation Techniques

Planning Poker was the most cited (09 papers) estimation technique, while Expert Judgment was also reported in some works (04 papers). Differently from the reference SLR [20], Use Case Points (UCP) Method has not been cited in any paper in this review

In this update, we observed a strong trend towards the use of intelligent techniques to estimate or support the estimation of effort. In the context of this work, an intelligent technique is defined as a technique that captures knowledge from data or individuals, discovers knowledge or automates routine tasks.

Half of the 24 papers described intelligent techniques for decision making in the process of effort estimation. Machine Learning (08 papers) Bayesian Networks (03 papers) and Optimization Algorithms (01 paper) were cited. Some works use an intelligent technique together with one traditional techniques (Planning Poker or Expert Judgment).

1) *Accuracy Metrics:* Question 1a investigates which prediction accuracy metrics were used in the works. As in the reference SLR, Mean Magnitude of Relative Error (MMRE) is the most frequently used accuracy measure. In Table 3, we showed the number of papers based on the used prediction accuracy metrics.

TABLE 3. Accuracy metrics used

Metrics	Papers IDs	F
MMRE	P1, P2, P3, P4, P6, P7, P8, P13, P15, P16, P17, P18, P19	13
PRED(25)	P6, P7, P8, P13, P15, P18, P19	7
PRED(8)	P16, P19	2
BRE	P5, P9	2
MSE	P6, P7	2

<sup>3</sup> <https://goo.gl/1ei1Sa>

Not Used	P11, P12, P20, P21, P22, P23, P24	7
Other	P14 (MRE); P10 (comparison with actual); P16 (MdMRE); P18 (MAE)	4

In this review we verified a decrease of the use of magnitude of relative error (MRE). Furthermore, comparing with the reference SLR, new metrics were cited such as Mean Square Error (MSE) in P6 and P7 works. Mean Absolute Error (MAE), Mean Squared Error (RMSE), Relative Absolute Error (RAE) and Relative Squared Error (RRSE) was cited in paper P18.

2) *Accuracy Level Achieved*: Question 1b looks into the accuracy levels achieved by different techniques. Table 4 shows the works that had the best results, in other words, the best levels of accuracy by technique.

TABLE 4. Accuracy achieved by techniques

Technique	Accuracy Achieved % (Paper IDs)
Planning Poker	MMRE: 16-61 (P1) BRE: 35 (P5)
Expert Judgment	MRE: 20-90 (P14) MMRE: 22 (P15) PRED (25): 73.13 (P15) BRE: 38.7-78.5 (P9)

We observed that even in the best results reported in Table 4, the level of accuracy is not good. Which in most cases did not turn out to meet the 25% threshold [3]. However, the works that use intelligent techniques presented better results. In the Table 5 we show a sample of these works and the levels of accuracy achieved.

TABLE 5. Accuracy achieved by Intelligent techniques

Intelligent Technique	Accuracy Achieved % (IDs)
Machine Learning	MMRE: 2.93 (P8) PRED (7.19): 100 (P8)
Bayesian Networks	MMRE: 6.21 (P18) PRED(25): 100 (P18)
Optimization Algorithms	MMRE: 5.69 (P16) MdMRE: 3.33 (P16) PRED (8): 66.67 (P16)

### B. (RQ2) Effort Predictors

1) *Size Metrics*: Since Planning Poker was the estimation technique most found, it was no surprise that the most reported size metric was story points. In short, 17 papers used story points, 03 studies have used Function Points, and other papers did not report the size metric. The result this question was similar to that found in the reference SLR, except we did not find papers using the metric Use Cases Points.

2) *Cost Drivers*: The works presented different factors that influence the estimation process. In general, most papers describe project factors. A specific paper uses people factors (P21). Many papers also cited people's factors and project factors (Both) as important in the process of estimating effort

in ASD. Only 25% (6 papers) of the works do not mention cost drivers in their finding.

We identified an increase on studies that report on cost drivers, and a trend for some of these factors. The reference SLR reports that there is little agreement on suitable cost drivers for ASD projects. We observed that multiple works use different classifications for mentioning the same predictor. Therefore, we used thematic analysis to classify the cost drivers identified in the selected papers. In Table 6 we present the result of this process.

TABLE 6. Cost Drivers

Cost Driver	Papers IDs	F
Quality Requirement	P1, P5, P8, P11, P12, P14, P15, P18, P20	9
Task Size	P3, P4, P8, P14	4
Integration	P1, P4, P8, P16	4
Priority	P2, P5, P10, P22	4
Complexity	P4, P5, P11, P12, P18, P22, P24	7
Delay Stakeholders	P8, P11, P12, P16	4
Team composition	P8, P11, P12, P16, P21	5
Work Environment	P8, P11, P12, P16, P21	5
Experience	P10, P11, P12, P20, P21, P23, P24	7
Technical Ability	P11, P12, P18, P20, P21	5

Quality Requirement was the most cited factor with 37,5% (9 papers). The papers report that the clarity of requirements, the level of uncertainty or ambiguity, and the characteristics of the application domain are crucial in the estimation process. Furthermore, regarding project factors, other cost driver cited was complexity with 29,1% (7 papers). Here, we consider all complexity related with technology or business solution. Factors related to the level of integration of components and tools, task size, priority and business value, and finally the delay response of Stakeholders were cited by many works.

Regarding people factors, the level of experience of the team was the most cited factor with 29.1 % (7 papers). Factors related to the work environment, team composition and technical ability were also cited, such as the ability with technologies or communication and management skills

### C. (RQ3) Characteristics of the Dataset or Knowledge Used

RQ3 looks into the domain (industry or academic). In short, industry remains the most reported domain used by the studies (14 papers). However, we can see that compared with the reference SLR, there was a considerable growth in the number of works describing validation in academic (7 papers) environment (i.e., 29.2% vs 13.6%).

We also analyzed details about the type of dataset used herein; cross-company was cited in only one paper, all others used within-company data. As in the reference SLR, we

believed these results are quite interesting as they suggest that within the scope of ASD, companies have focused on their own project data, rather than looking for data from cross-company datasets.

#### D. (RQ4) Characteristics of the Dataset or Knowledge Used

This research question is designed to identify the specific agile methods used in effort estimation studies in an ASD context. Agile methods are concrete approaches to materialize the manifesto's<sup>4</sup> values and principles towards agility [11]. According to this review, Scrum is the most frequent method used (11 papers) as software development methodology in the ASD context. Some works (04 papers) use some combined features of Scrum and XP. Finally, others papers don't describe explicitly the method adopted, simply describe the method as being something with regular deliveries, fast customer feedback and emphasis on development rather than documentation.

1) *Development Activity*: Question 4a investigates the development activities (Analysis, Design, Implementation or Testing) to which the effort estimate applies. In this update of the literature, only one of the 24 works exclusively uses one activity of the development cycle, in the case of Implementation (P4). All other works either do not explicitly describe which activities or mentions that the effort estimate matches the functionality completely, from its analysis until it is ready for delivery.

2) *Planning Level*: Question 4b investigates the Planning Level. Out of the 24 works only 2 refer to release planning (P11, P12), meanwhile the others did not mention or report that the studies evaluated estimates of effort at each Iteration.

## V. DISCUSSION

We presented in Subsection A, a comparison between the new findings and the results of the reference SLR published in 2014 by Usman et al. [20] checking the progress on questions researches. While in Subsection B we presented new discoveries, suggest lines of research and threat to validity.

### A. Comparison with reference SLR

The results of this study address four research questions, which explore aspects related to effort estimation in agile software development projects. Regarding estimation techniques, planning poker is the most commonly cited method. However, expert Judgment was also found in the research. These techniques are effective when the team has similar experiences in the past when working in new project.

Comparing with the reference SLR, we observed a decrease in the use of Expert Judgment (i.e., 16.7% vs 20%), and an

increase use of intelligence techniques to support the effort estimation (i.e., 50% vs 27.5%). In particular, we did not mention new works citing Use Case Points (UCP) method, which was reported in 3 primary studies in the reference SLR.

In Usman et al. [20], MMRE was the most popular metric of accuracy, which was also observed in this update. Unlike the reference SLR, where it was widely cited, MRE was cited by only in one study in this review. BRE metrics cited in the reference SLR also remain in evidence and are reported in this work. Some works justified the choice of the BRE because of criticism of the MRE regarding its lack of balance [12].

The research on size metrics showed that story point remains the most largely used measure, mainly motivated by being the measure used with planning poker. In this review we did not find records of the use of use cases points as size metrics, which is a consequence of not finding mentions to UCP method. Some papers used characteristics of the functional size of the features to help in the process of estimates of effort in agile environments. However, the results reported by the authors has moderate significance. The size metric Lines of Code cited in 1 of the primary studies of the reference SLR was not found in this update.

Usman et al. [20] reported that the low accuracy in effort estimation could be related to the lack of clarity in cost drivers and that new studies in this context needed to be carried out. We identified an increase in the use of Cost Drivers in the studies. Only 25% (6) of papers do not use some cost driver.

In the reference SLR, industrial datasets are used in most of the studies. Although this persists in the new review, we identified an increasing number of works which were validated in the academic domain. We believed that the works that were validated only in the academic domain requires future works to replicate the proposals on projects in the industries, so that the conclusions can be valid of practitioners.

Usman et al. [20] believed that some effort should be made to make cross-company datasets available for ASD context. However, this was not found in this review. Only one paper validated the data in cross-company environment, all others in within-company environments. We believed that the greatest difficulty in this context is that effort measures are relative and subjective to teams. This hinders a common measure in cross-company with different teams.

We only found 4 works describing the use of Extreme Programming (XP) and all of them had XP used in combination with Scrum. In the reference SLR, XP was cited 7 and Scrum, 8. In our update, we found a total of 15 papers that reported to use Scrum (i.e., 4 that use XP and Scrum and 11 that use Scrum).

We believed that this increase is a consequence of, today, Scrum being the most popular agile method in industry, as reported by VersionOne [21]. Many works did not explicitly set out which agile method was they used. Regardless of the agile methodology, we agree with Usman et al. [20] that the activities when they are estimated refer to the complete

---

<sup>4</sup> <http://agilemanifesto.org>

development effort. The design, implementation and testing activities are performed very closely to each other. As in the reference SLR, the works reported in this review mostly deals with planning at the level of iterations.

### B. Implications for research and practice

An important gap cited by Usman et al. [20] is the lack of studies that show good estimation accuracy. We identified in this update that accuracy remains a challenge in most of the papers analyzed. However, an enhancement is clearly observed in works that use intelligent techniques in this context (see Table 5).

We observed an increasing usage of intelligence techniques for effort estimation in ASD. Half of the 24 works uses some Artificial Intelligence or Machine Learning technique. These works use historical data and expert knowledge to support decision-making. Since some of them validated their approaches only in academia, there is a need for further researches with replication of these techniques in the industry.

It was also mentioned by Usman et al. [20] that the lacking of the consensus in costs drivers is a reason for poor accuracy levels. In this research, we used thematic analysis to categorize the predictors that were identified in the primary studies. The result showed 10 factors (see Table 6), in which five are related to projects and five to people. We believed that further case studies in industry evaluating these cost drivers are needed.

A potential threat to validity is, as for any systematic literature review, if we were not able to cover all primary studies. We used Forward Snowballing because we considered unlikely that a relevant paper published in 2014 or later does not refer to any of the papers results of the SLR from Usman et al. [20]. In regard to the quality of the selection of the study and data extraction, we used a systematic approach where which each paper was evaluated by at least two reviewers, to avoid reviewer bias and human errors.

## VI. CONCLUSION

This study presents an update review of a reference systematic literature review on effort estimation in Agile Software Development [20]. Forward Snowballing was used to look for the most relevant works in this theme since the year of 2014. The seed set used for beginning the search were the relevant works listed in the reference SLR and the reference SLR itself. After two evaluation cycles using Forward Snowballing and 312 works evaluated we selected 24 relevant papers.

Some considerations of the reference SLR are still valid and current. However, we identified new trends. We believe that further investigation into the use of the cost drivers and

intelligence techniques is needed in effort estimation, given the possible benefits for a best management of agile projects. Further efforts in academia and industry are need to be made in this direction.

## REFERENCES

- [1] Achimugu, P., Selamat, A., Ibrahim, R., Mahrin, M.N.: A systematic literature review of software requirements prioritization research. *Information and Software Technology* 56(6), 568–585, 2014.
- [2] Bilgaiyan, S., Mishra, S., Das, M.: A Review of Software Cost Estimation in Agile Software Development Using Soft Computing Techniques. *International Conference on Computational Intelligence and Networks (CINE)*, 112–117, 2016.
- [3] Conte S. D., Dunsmore V.Y.S. H. E.: *Software engineering metrics and models*. In: Benjamin-Cummings Publishing Co, 1986.
- [4] Felizardo, K.R., Mendes, E., Kalinowski, M., Souza, E.F., Vijaykumar, N.L.: Using Forward Snowballing to update Systematic Reviews in Software Engineering. *International Symposium on Empirical Software Engineering and Measurement - ESEM '16*, 1–6, 2016.
- [5] Grapenthin, S., Book, M., Richter, T., Gruhn, V.: Supporting Feature Estimation with Risk and Effort Annotations. *42nd Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2016*, pp. 17–24, 2016.
- [6] Jalali, S., Wohlin, C.: Systematic Literature Studies: Database Searches vs. Backward Snowballing. *International Symposium on Empirical Software Engineering and Measurement*, pp. 29–38, 2012.
- [7] John Wiley and Sons: Introduction to Qualitative Research Methods: *The Search for Meanings*. New York. John Wiley and Sons, 1984.
- [8] Kitchenham, B., Charters, S.: Guidelines for performing Systematic Literature Reviews in Software Engineering. *Engineering* 2, 1051, 2007.
- [9] Kupiainen, E., Mantyla, M.V., Itkonen, J.: Using metrics in agile and lean software development – a systematic literature review of industrial studies. *Information and Software Technology* 62, 143–163, 2015.
- [10] Lenarduzzi, V., Lunesu, I., Matta, M., Taibi, D.: Functional size measures and effort estimation in agile development: A replicated study. *Business Information Processing*, vol. 212, pp. 105–116, 2015.
- [11] Melo O., Santos C., Katayama V., E., Corbucci, H., Prikladnicki, R., Goldman, A., Kon, F.: The evolution of agile software development. *Brazil. Journal of the Brazilian Computer Society*, 523–552, 2013.
- [12] Miyazaki, Y., Takanou, A., Nozaki, H., Nakagawa, N., Okada, K.: Method to estimate parameter values in software prediction models. *Information and Software Technology* 33(3), 239–243, 1991.
- [13] Munialo, S.W., Muketha, G.M.: A Review of Agile Software Effort Estimation Methods, 612–618, 2016.
- [15] Schweighofer, T., Kline, A., Pavlic, L., Hericko, M.: How is Effort Estimated in Agile Software Development Projects? Sqamia, 2016.
- [16] Sehra, S.K., Brar, Y.S., Kaur, N., Sehra, S.S.: Research patterns and trends in software effort estimation, 2017.
- [17] Silva, A., Perkusich, A.: A systematic review on the use of Definition of Done on agile software development projects, 2017.
- [18] Silva, A., Ramos, F., Silva, A.: Ordering the Product Backlog in Agile Software Development Projects : A Systematic Literature Review *International Conference on Software Engineering & Knowledge Engineering –SEKE*, 2017.
- [19] Usman, M., Mendes, E., Borstler, J.: Effort estimation in agile software development: A survey on the state of the practice. *International Conference on Evaluation and Assessment in Software Engineering. EASE '15*, pp. 12–11210. ACM, New York, NY, USA, 2015.
- [20] Usman, M., Mendes, E., Weidt, F., Britto, R.: Effort estimation in Agile Software Development: A systematic literature review. *ACM International Conference Proceeding*, 82–91, 2014.
- [21] VersionOne: 11th Annual State of Agile Development Survey Results. <https://versionone.com/pdf/VersionOne-11th-Annual-State-of-AgileReport.pdf>. Accessed in: 02-12-2017, 2017.
- [22] Wohlin, C.: Guidelines for snowballing in systematic literature studies and a replication in software engineering. *International Conference on Evaluation and Assessment in Software Engineering - EASE '14*, pp. 1–10, 2014.

# SIDD – SCRUM ITERATION DRIVEN DEVELOPMENT: AN AGILE SOFTWARE DEVELOPMENT AND MANAGEMENT PROCESS BASED ON SCRUM

Tayse Virgulino Ribeiro<sup>1</sup>      Cristina D’Ornelas Filipakis Souza<sup>2</sup>      Heloise Acco Tives Leão<sup>3</sup>

<sup>1,2</sup>Lutheran University Center of Palmas - CEULP/ULBRA, Palmas – TO – Brazil

<sup>3</sup>Department of Computer Science - University of Brasília (UnB) – Brasília-DF, Brazil

E-mail: {tayse1000, filipakis, heloise.acco}@gmail.com

## Abstract

*Scrum Iteration Driven Development is an agile, Scrum-based process that allows the management and development of software, making possible an improvement in the software development process, with the objective of assisting in the management and planning of software projects. The proposal of Scrum Iteration Driven Development (SIDD) is to be an agile process capable of directing development and management activities. Therefore, considers the context of the company, to help them follow the principles of this process in its entirety, without the need for complementary techniques. Finally, to disseminate the results of the work, an interactive presentation area of the SIDD process was developed with information related to the software development and management process, with the intention of being a guide to assist the users in the project management process<sup>1</sup>.*

**Keywords** - Software Engineering, Agile Methodology, Scrum, Development Process, SIDD.

## 1 INTRODUCTION

Pressman [3] states that software engineers should strive to produce and use techniques and tools to develop high-quality systems. Currently, the reality for large and small companies is the difficulty to understand the concepts of Software Engineering that leads to misuse of software development practices.

Therefore, this work emphasizes a specific agile software development methodology, called Scrum. According to Sabbagh [5], Scrum is an agile, simple and lightweight framework used to manage the development of complex products. Identifying it as a framework means that it is able

to solve a problem in a particular domain, and to solve this problem, your applications needs to be working according to project standards.

In this way, the present work aims to create an agile process, allowing the development and management of software in an iterative way. This process is based on a qualitative and quantitative research carried out with 20 companies from the city of Palmas, Tocantins - Brazil, in April 2017 by Ribeiro [4], addressing the interviewees about the use of software development practices. This research aimed to identify the approach of the use and the evolution of the companies in the context of software development process, to obtain a better understanding of the use of agile practices. Therefore, to propose a software management and development process in order to provide improvement in the software development process.

Based on this, the problem that this paper seeks to solve is related to the creation of a Scrum-based agile process for software development and management in order to provide improvement in the software development process of companies. Raising as hypothesis, if there is an understanding of the company profile analysis and the approach to using Scrum, then it is possible to create a Scrum-based agile process for software development and management in order to provide improvement in the software development process.

Thus, the proposal to use the SIDD process has positive aspects in the aid of the administration of project management activities, such as the creation of specific artifacts that aid in the management and development of the software. However, the process tends to lose agility face other agile processes, because there is an addition of events and artifacts construction in its tasks. This addition is what introduces the management context to the process.

<sup>1</sup>DOI reference number: 10.18293/SEKE2018-102

## 2 THEORETICAL FRAMEWORK

After the “software crisis” in the mid-1990s, agile methodologies became known as methods of improving development processes. In the year 2001, an Agile Manifesto was written by Kent Beck and sixteen other software professionals. This manifesto generated twelve principles that explain the concept of agility in the area of software development. From that year, the term “Agile Methodologies” became popular in this field.

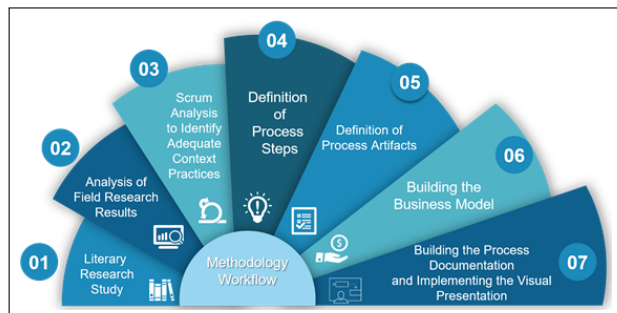
According to Sharma and Hasteer [6], in the world of software development industry, traditional software development has moved to the agile model over the years. In this field there are many methodologies, such as Scrum, Extreme Programming, Crystal, FDD (Feature-driven development), DSDM (Dynamic Systems Development Method, etc. The agile process developed in this work emerged as a technique to aid in software development.

Khmelevsky, Li and Madnick [2] complement that agile practices such as Scrum, which allow a better dynamic of concentration between the teams and an agile delivery, have become one of the easiest and most desirable techniques to work with. On the other hand, it should be noted that it is necessary to evaluate the type of project to which the technique will be applied, since the process may not be directly applicable.

The use of an agile process is directly related to the application environment and the approach type. Therefore, Srivastava, Bhardwaj and Saraswat [7] state that the use of an agile method has a different approach and perspective for each project.

## 3 METHODOLOGY

The composition of the work development methodology elaborated for design the SIDD process is shown in Figure 1. In the study phase of the theoretical reference was car-



**Figure 1. Methodology of the developed stages of the project.**

ried out a study of the main concepts related to the use of

agile development processes, as well as quality approaches to software processes. After that, a qualitative-quantitative research was carried out in 20 software development companies in Palmas-TO, Brazil in the year 2017. From the analysis of the results obtained in the research it was possible to start planning the business model, which is the third stage of the work methodology.

A business model was created for the process, in order to describe aspects related to the use of the management and development process in the job market. In this way, were verified the benefits that this process will provide for its customers, the resources necessary for its use to be possible and the way in which the communication with the client will be established.

After the construction of the business model, an analysis of the Scrum development process was carried out to identify suitable practices for the scenario. This analysis was carried out in order to understand the scenario of software development in private companies, with the purpose of defining a proposal for a software management and development process appropriate to the companies.

Faced with this, the definition of the steps of the SIDD Process was started with the purpose of presenting the definition of the workflow of the software development and management process that is developed in this work.

With the steps delineated, the definition of the artifacts was started, elucidating all the artifacts used for the composition of the process and presentation of the documentation. And finally, the visual representation of the SIDD process was created. In this phase the flow is presented in a graphical way, relating the roles to the stages in which they are involved.

## 4 QUALI-QUANTITATIVE RESEARCH

A qualitative-quantitative research was carried out in 20 software development companies in Palmas-TO, Brazil in the year 2017 with the objective of evaluating the evolution and needs of the utilization of development processes in the companies and agencies of this city. According to what was collected in the research, where it can be seen that seven companies did not adopt a specific software development practice, two adopted traditional development practices, while eleven adopted agile practices. In addition, 100% of companies that used agile practices used the Scrum development process as a basis, but with adaptations.

In some companies/agencies there was adaptations related to term and events that are approached in agile practices. Six of the companies, made adaptations of sprints and backlog times in project planning, changing the schedule, as they can not meet the deadline initially reported. One of these adaptations, for example, refers to the project progression meetings (Daily's - Scheduled Scrum Meetings).

The adjustments made at the meetings were not specified. But the adaptations can be related to two points, which are: not holding meetings or holding meetings more consistently (e.g. a meeting per day period rather than at the end of the day). Some adaptations are not possible to identify, since the partial way of use was not informed and in other cases the responsible person was not able to respond. It is worth mentioning that all adaptations should be evaluated, since accession to a methodology basically means following its principles.

Based on this, it is necessary to take some aspects into account before adopting a development practice. The company / organization must make a diagnosis in the sector, to verify how its management style behaves and if they have trained individuals. After identifying these characteristics, the entire industry and the company must commit to the plans that will be carried out, since this is a significant influence. According to the theoretical reference, it is possible to observe some important criteria when adopting a software development practice. According to Awad [1] and Stoica, et al. [8], there are a number of factors influencing the traditional methodology, such as: methodology approach; maturity of the company/body and team; team composition; level of project knowledge; perspective of project change, communication, culture, documentation and the project's return on investment.

Thus, a proposal for a new agile software development process capable of directing the development and management activities, besides taking into account the context of the company, can contribute for the companies follow the processes in its entirety, without the necessity for complementary techniques.

## 5 SCRUM ITERATION DRIVEN DEVELOPMENT

The process of development and management of SIDD software was obtained through the case study carried out, in which the presentation flow was created with the aim of facilitating the presentation of the software development and management process. Figure 2 presents, graphically, the workflow of the process developed in this work.

The division of the process takes place in the accomplishment of the events, in the construction of the artifacts and in the participation of their respective roles. The SIDD process consists of the following events: Planning Meeting, Sprint Planning, Planning Poker (optional), Sprint, Daily SIDD and Sprint Review.

In the execution of these events the following artifacts are generated: Project Model Canvas, Product Backlog, Sprint Backlog, Object Model(optional), Sequence Diagram (optional), Testing Session, Definition of Done and Product Increment.

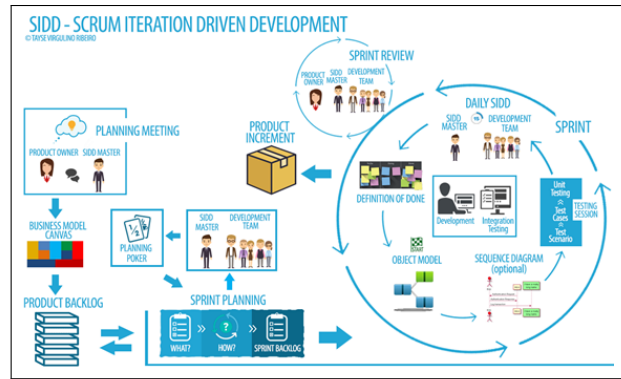


Figure 2. Structure of the Process.

Thereby and to carry out the SIDD cycle, the participation of SIDD Team is required. This team consists of the following roles: Product Owner, SIDD Master and Development Team. These roles aim to achieve results with quality and agility, in a proactive and collaborative way. The performance of a role is directly related to a set of executed events and their developed artifacts.

The participation of the Product Owner in the SIDD process begins in Planning Meeting. This meeting discusses the need of the customer and what will be generated of value. With this, the Product Owner and SIDD Master work to define the Project Model Canvas. Based on this, the items that make up the Product Backlog are defined. Finally, it participates in conjunction with the SIDD Master and the Development Team in the performance of the Sprint Review activity, assisting in the evaluation regarding the fulfillment of the Sprint objective, characterizing the delivery of the product. SIDD Master is responsible for leading and facilitating the work of the Development Team in order to promote organizational skills, communication and continuous improvement in the development process. SIDD Master is one of the only roles that is present all the time in the product design process because it aims to monitor the progress of all activities and keep the project always aligned with the needs of the Product Owner.

Its participation begins in the Planning Meeting. After that, the Product Owner and SIDD Master work to define the Project Model Canvas. As a consequence, the items that make up the Product Backlog are defined. In addition, Sprint Backlog planning is done in the Sprint Planning event. In the act of this planning the SIDD Master and the Development Team are also performing the Planning Poker. After that, at the execution of Sprint, the Development Team contributes in the accomplishment of the daily meeting, known like Daily SIDD. During the execution of Sprint, SIDD Master participates in assisting and leading the Development Team in the construction of the artifacts that aid in the development of the product, such as: Object

Model(optional), Sequence Diagram (Optional), Test Session, Definition of Done and Product Increment.

The Development Team is responsible for developing the product. Therefore, the team should contemplate some characteristics, such as: specialty in the field, self-organization, focus, motivation, discipline, agility and teamwork. The Development Team begins its participation in the Sprint Planning event. In this event, you define what will be done from the beginning of the project execution. At the time of this planning, the team also conducts Planning Poker. After that, in the execution of Sprint the Development Team contributes in the accomplishment of the Daily SIDD. During the execution of Sprint, the team participates in the generation of the same artifacts that aid in product development.

To disseminate the results of the work, an interactive presentation area of the SIDD process was developed, with information related to the software development and management process, which can be accessed through the following link <http://metodologiasidd.com.br/>. This area was presented according to the structure of the process, which is performed in the accomplishment of the events, in the construction of the artifacts and in the participation of their respective roles. The development of the site was mainly aimed at the development of the process guide area (<http://metodologiasidd.com.br/guide.html>), in which it is possible to obtain the orientation of the Scrum Iteration Driven Development process, presenting the main areas of the process, which are: events, artifacts, and roles.

## 6 CONCLUSION

In the analysis of the results of the qualitative-quantitative research carried out by Ribeiro [4], it was possible to observe that the companies that worked with the practice of agile development did not necessarily meet all the principles of a methodology, since much of its use was made in a partial way. Besides that, it was noted that the use of Scrum as a methodology was standard in the software development companies in the region. Thus, this research had the goal of questioning respondents on the use of software development practices analysis of the information obtained.

In this way, the proposal of a new agile process capable of directing the activities of development and management, in addition to taking into account the company context, can assist companies to follow the principles of this process in its entirety, without need for complementary techniques. Based on this, the objective of this study was to provide an improvement in the software development process of enterprises. Therefore, in addition to explaining the concepts of agile methodology and the steps of the process Scrum, this work consisted in modeling an agile process based on Scrum that allows software management and development.

From this, the Scrum Iteration Driven Development, an agile process based on Scrum, was developed, which allows the management and development of software, making possible an improvement in the software development process, with the objective of assisting in the management and planning of software projects.

Therefore, the use of the SIDD is beneficial in assisting the management of project management activities, such as creating specific artifacts that assist the management and development of software. However, the process tends to lose agility face other agile processes, because there is an addition of events and artifacts construction in its tasks. This addition is what introduces the management context to the process.

## References

- [1] M. Awad. A comparison between agile and traditional software development methodologies, this report is submitted as partial fulfillment of the requirements for the honours. In *The University of Western Australia*. Citeseer, 2005.
- [2] Y. Khmelevsky, X. Li, and S. Madnick. Software development using agile and scrum in distributed teams. In *2017 Annual IEEE International Systems Conference (SysCon)*, pages 1–4, April 2017.
- [3] R. Pressman. Engenharia de software: uma abordagem profissional [internet], 2011.
- [4] T. V. Ribeiro. *Quantitative Research on Software Development Processes Used by Companies in Palmas-TO*. 2017.
- [5] R. Sabbagh. *Scrum: Gestão ágil para projetos de sucesso*. Editora Casa do Código, 2014.
- [6] S. Sharma and N. Hasteer. A comprehensive study on state of scrum development. In *2016 International Conference on Computing, Communication and Automation (ICCCA)*, pages 867–872, April 2016.
- [7] A. Srivastava, S. Bhardwaj, and S. Saraswat. Scrum model for agile methodology. In *2017 International Conference on Computing, Communication and Automation (ICCCA)*, pages 864–869, May 2017.
- [8] M. Stoica, M. Mircea, and B. Ghilic-Micu. Software development: Agile vs. traditional. *Informatica Economica*, 17(4):64, 2013.



# Investigating the Effects of Agile Practices and Processes on Technical Debt - The Viewpoint of the Brazilian Software Industry

Viviane Coelho Caires  
PPGCOMP, Salvador University  
Federal Institute of Bahia-IFBA  
Jequié/Vitória da Conquista, Brazil  
viviane.aires@ifba.edu.br

Nicolli Rios  
Department of Computer Science,  
Federal University of Bahia  
Salvador, Brazil  
nicollirios@gmail.com

Johannes Holvitie  
TUCS - Turku Centre for Computer  
Science & University of Turku  
Finland  
jjholv@utu.fi

Ville Leppänen  
TUCS - Turku Centre for Computer  
Science & University of Turku  
Finland  
ville.leppanen@utu.fi

Manoel G. de Mendonça Neto  
Department of Computer Science,  
Federal University of Bahia  
Fraunhofer Project Center @ UFBA  
Salvador, Brazil  
manoel.mendonca@ufba.br

Rodrigo Oliveira Spínola  
PPGCOMP, Salvador University  
Fraunhofer Project Center at UFBA  
Salvador, Brazil  
rodrigo.spinola@unifacs.br

**Abstract**—The current scenario of software development is characterized by a wide adoption of agile methodologies, which define processes and practices that address a range of problems faced by development teams. However, there is still little information on how these methodologies deal with technical debt(TD). This work presents the results of a replicated survey(originally executed in Finland) whose goal was to investigate which agile practices and processes are sensitive to TD. Despite this replication allows different types of analysis, the focus of this paper will be on the analysis of the effects of the agile practices and processes on TD from the perspective of the Brazilian software industry, where the study was replicated. At total, 62 practitioners from different organizations answered the questionnaire. The results indicated that participants already had a good knowledge about TD, instances of TD reside in the software implementation and are caused due to deficiencies in its architecture, the size of a debt item is proportional to its impact on the project, and, refactoring and iteration have the most positive effect on TD. This replication also contributes to the investigated topic through the accumulation of evidence about the findings, thereby increasing the level of confidence in results.

**Keywords**—*Technical debt; agile methodology; survey; replicated study.*

## I. INTRODUCTION

Technical debt (TD) represents the effects of immature artefacts that bring short-term benefits in terms of increased productivity and lower costs, but which may need to be adjusted later with interest during software development [1, 2, 3]. TD is usually incurred when development teams have to choose between to evolve the system considering quality standards or to put it to run in the shortest possible time, using minimum resources. As TD is incurred in a project, the effort required to eliminate it is cumulative and its payment tends to become more complex. Different types of debt may occur during the phases of a software development process and the used methodology can affect their presence [4]. An inadequate

management of TD can bring significant losses to a software project [7].

The current scenario of software development is characterized by a wide adoption of agile methodologies, which define processes and introduce practices that address a range of problems currently faced by development teams [5]. However, there is still little information on how these methodologies accommodate the concept of TD. To shed some light in this discussion, Holvitie *et al.* [9] conducted a survey with practitioners from Finland on how TD issues relates to agile software processes and practices. They investigated participants' level of knowledge on TD, how TD manifests itself in their projects, and what processes and practices of agile development are sensitive to it. In general, the study pointed out that the processes and practices that are closest to the implementation and maintenance activities are perceived as having the most positive effects on the control of TD. In addition, the authors also identified that TD items usually come from problems in the software architecture.

Although the surveys' results are valid, the work of Holvitie *et al.* [9] is also limited by some issues. The main one is that the data were collected from development companies based in Finland and, therefore, the results may reflect only the local scenario. To deal with this, an international consortium involving researchers from Finland, Brazil and New Zealand worked together to replicate the survey in their respective countries. The goal of this set of replications was investigate whether the findings of the Holvitie's study are reproducible. Results from the whole gathered data were reported in [10]. However, as the results presented in [10] did not consider the specificities of each involved country (the whole dataset was analyzed as an only instance), an in-depth analysis of the results of each individual replication is still missing. This kind of analysis can reveal hidden details that could not be perceived in a generic look at the data. More specifically, it can reveal how software practitioners perceive impacts that agile practices

and processes have in TD considering the local software industry reality that usually differs from other places when we consider variables like the size of organizations and development teams, and the size and duration of software projects.

This work presents the results of the replication<sup>1</sup> of the study of Holvitie *et al.* [9] in Brazil, a country located in another continent and with a culture different from Finland. Despite the fact of this replication allows different types of analysis (for example, comparison between the results from each country), the focus of this work is to discuss the results of the replication in Brazil. Thus, we will present an analysis on the effects of the agile software development practices and processes on TD from the perspective of the Brazilian software industry. We will discuss the answers to the following research questions: **RQ1** - *What is the level of knowledge of respondents about TD?*; **RQ2** - *Which agile software development practices and processes are sensitive to TD?*; and **RQ3** - *How does TD manifest itself in the participants' work?*

To replicate the survey, we used a web-based questionnaire that was answered by 62 practitioners (mostly characterized by professionals with more than 6 years of experience) from 62 different software organizations. In general, the results indicated that the participants already had a good knowledge about the concept of TD, but some of them are still not familiar with the term. In another finding, we could observe that many instances of TD reside in the software implementation and are caused due to deficiencies in its architecture. We also identified that the size of a debt item is proportional to its impact on the project. Finally, considering all analyzed agile software development practices and processes, most of respondents indicated that refactoring (practice) and iteration (process) have the most positive effect on TD.

In addition to this introduction, this paper has five more sections. In Section II, the replication of the survey in Brazil is described. The effects of the agile software development practices and processes in TD are presented in Section III. Section IV discusses the obtained results. Next, limitations of the study are presented in Section V. Finally, Section VI presents some final remarks.

## II. SURVEY REPLICATION - BRAZIL

### A. Survey

The goal of the research performed by Holvitie *et al.* [9] was to investigate which agile software development practices and processes are sensitive to TD. They conducted a survey, structured in three groups of questions, considering a population of practitioners.

The first group of questions aims to establish the level of knowledge of the respondents about software development and how they perceive TD in their projects. For this, the research questions were defined as, **for an individual**: (**RQ1.1**) does work experience, (**RQ1.2**) do used agile development practices, or (**RQ1.3**) do associated project responsibilities correlate with

what the respondent perceives his/hers assumed or actual TD knowledge to be?; (**RQ1.4**) in which mediums has he/she seen or heard the term TD be used?; (**RQ1.5**) in which situations has he/she or his colleagues applied the concept of TD?, and; (**RQ1.6**) in which situations does he perceive the use of the TD concept as helpful? During this first stage, the authors also present the McConnell's definition of TD [7], ensuring that all participants know the term.

In the second stage, there is a set of questions about which agile development practices and processes are used by respondents in their projects and how they realize that their use affects TD. We established that the XP practices together with Scrum processes cover the components of agile software development well in addition to being highly popular [11, 12]. Questions of this stage intend to answer the following research questions: **are there certain agile software development practices or processes for which** (**RQ2.1**) their effect on technical debt is seen to be significantly positive, neutral or negative?; (**RQ2.2**) it is seen that they (do not) cover the team's or the project's development management needs?, and; (**RQ2.3**) it is seen that they (are not) able to cover TD issues that require management?

In the third stage of the survey, participants are asked to cite particular instances of TD and, from that concrete instance, answer the following research questions: **for a concrete instance of technical debt**, (**RQ3.1**) in which phase of the software development it was observed?; (**RQ3.2**) what are the causes for its emergence?; (**RQ3.3**) is it legacy?; (**RQ3.4**) is its size dynamic?, and; (**RQ3.5**) does its effects correlate with its size?

In total, the questionnaire has 37 questions (35 objective and 2 subjective) and collects the following information: (i) participants' knowledge on software development; (ii) organizational details (such as participants' role in the project, number of projects developed by the company, number of people involved in a given project); (iii) agile development processes and practices that are applied; (iv) interviewee's knowledge on TD; (v) perception of the development phases affected by TD; and (vi) an example of an artifact affected by TD, the size of that debt item and its perceptible effects. The survey, available at <http://soft.utu.fi/tds16/questionnaire.pdf>, was developed as a web-based form in order to increase the response rate and minimize data manipulation errors. Google Forms platform was used for building, distributing, and collecting survey data.

### B. Survey Brazil

When we decided to replicate the survey in Brazil, it was already designed and all the instruments were available. Therefore, in this section we focus on the details of how we planned and operated the replication in Brazil. Further information on the design of the survey can be found in [9].

To plan the survey replication in Brazil, we held a couple of discussions with the general organizers. During the discussions, the online questionnaire was presented and some general guidelines for conducting the survey were provided. Thus, the configuration of the environment was performed and then the participants were invited by e-mail to contribute with the

<sup>1</sup> Replication based on previous insights is widely recommended in the experimental paradigm [13]

research. Participants were selected through software associations or local industry contacts. In this process, we tried to reach practitioners spread out in different regions in Brazil.

In total, 62 professionals from different software development organizations answered the survey. Regarding the size of the organizations in terms of number of employs, 44% of the respondents work on organizations with over 250 employs. A significant number (30%) of answers were also obtained from participants of organizations that have between 10 and 50 employs. 10% of the participants work on small companies with less than 10 employs. Finally, 16% of the respondents indicated that work on companies that have between 51 and 250 employs. The development teams in which participants are involved in are mainly characterized as small teams (42%, 2-5 members). 25% of the respondents work on teams that have between 6-10 members. We also had answers from teams with over 20 members (10%). The other participants are part of very small or middle size teams.

The length of projects in which participants are working on has the following distribution: 1-3 months (23%), 4-6 months (31%), and over 6 months (38%). Regarding development iteration length, the answers are distributed as follows: one week or less (18%), 2-3 weeks (26%), 1 month (10%), 2 months (5%), over 2 months (10%), no iteration (20%). Finally, concerning respondent level of experience, approximately 15% of the respondents have less than 3 years of software development experience, slightly more than 20% have between 3 and 6 years, and 65% have more than 6 years. The average time to complete the questionnaire was 15 minutes.

To ensure a standardized data analysis in relation to the work of Holvitie *et al.* [9] and make possible a future comparison between them, we forwarded the responses to the general organizers, which applied the same analyzes carried out in the study performed in Finland. Then, they returned the results and we could interpret them.

### III. RESULTS

In this section, we discuss the survey results concerning (i) what is the level of knowledge on TD of the participants, (ii) what agile software development practices and processes are perceived as sensitive to TD, and (iii) how TD manifests itself in their work.

#### A. RQ1 - What is the level of knowledge of respondents about TD?

Research questions grouped by RQ1 are focused on participant's perception on the concept of TD. For this, participants are initially asked about how they perceive their knowledge on TD, followed by a request for them to (optionally) describe their definition of the term. These answers were classified according to respondents' work experience (RQ1.1), applied software development techniques (RQ1.2) and assumed roles (RQ1.3). There was no significant difference between the distributions of these variables. Thus, the most general one is presented here. From Figure 1, it is observed that 32% of the respondents considered having a good or very good definition of TD, however, almost 50% of them

indicated that they did not know the term or had a poor definition of it.

Next, the McConnell's definition of TD [7] was presented and, then, the respondent was asked to indicate how close to this concept was his initial understanding. The results are represented in Figure 1 and indicate that about 70% of respondents reported that their definition were close to or very close to the definition extracted from the technical literature. Besides, slightly more than 20% reaffirmed not knowing the term or having a poor definition of it. These data indicate that participants were initially reticent about their understanding on concept of TD, but that most of them (80%) really already knew it. Other surveys performed in the area have pointed out this same behavior [6][9].

Complementing this analysis, Figure 2 presents the relationship between the experience of survey participants, their previous knowledge on TD and their knowledge after the definition be presented in the questionnaire. We can see that for interviewees with less than 3 years of experience, 5% had a good or very good definition for TD, 8% had a poor or very poor definition, and 2% reported not knowing the term. After reading the definition presented in the survey, the percentages passed to 11%, 2% and 2%, respectively. For participants who had experience between 3 and 6 years, 3% indicated having a good or very good definition for TD, 16% a poor or very poor definition and 2% indicated not knowing the term. After reading the definition, the percentages passed to 10%, 2% and 10%, respectively. Finally, for the most experienced participants (more than 6 years of experience), 25% reported having a good or very good definition for TD, 29% had a poor or very poor definition, and 11% reported not knowing the term. This percentage changed to 49%, 10% and 6%, respectively, after reading the TD concept presented.

Then, the respondents were asked where they had either

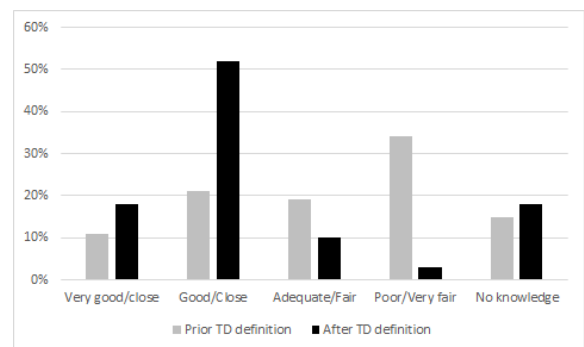


Figure 1. Distribution for perceived TD knowledge

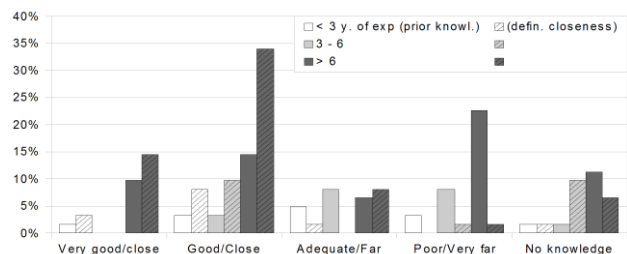


Figure 2. Relationship between interviewees' experience and knowledge on TD concept

seen or heard the term TD used (RQ1.4). The questionnaire provided seven initial options that can be observed in Figure 3. We can see that more than 50% have seen the term in the technical literature. Surprisingly, about 40% of the respondents reported the term has been used in work meetings. It is also important to mention that over 15% of respondents never had heard the term before.

Finally, closing the analysis of RQ1, a mapping of common

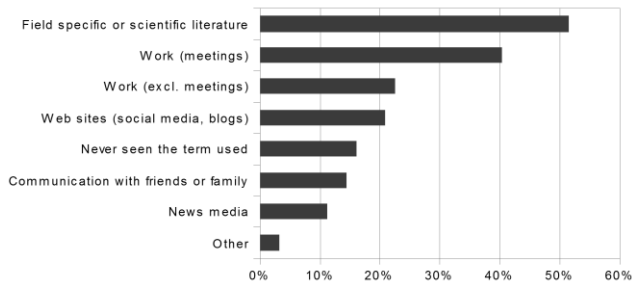


Figure 3. Technical debt usage in different mediums

decision situations in development is shown in Figure 4. We asked participants whether, for each situation, he/she or a co-worker had already applied the concept of TD (RQ1.5) and whether the use of the concept would have been useful (RQ1.6). The data show that more than half of the respondents realized the utility of using the concept of TD in all situations and only 5% reported that its use would not bring gains.

Still on Figure 4, 27% of respondents reported that they have already applied the concept of TD in unforeseen situations, almost 20% in decisions about development infrastructure, about 20% in integrated resources, and 35% in conduction of software development. From the perspective of a co-worker, 27% reported that a colleague had already used the term TD in unforeseen situations, 18% in issues involving development infrastructure, 10% in integrated resources, and almost 30% in the conduction of software development. It is worth mentioning that more than 50% of respondents never used the TD concept in decision-making in any of the situations.

**Finding 1:** The concept of TD is already known by a large part of the population represented in this study. On the other hand, practitioners are still assimilating the concept.

**Finding 2:** The usefulness of using the TD concept in development activities is recognized.

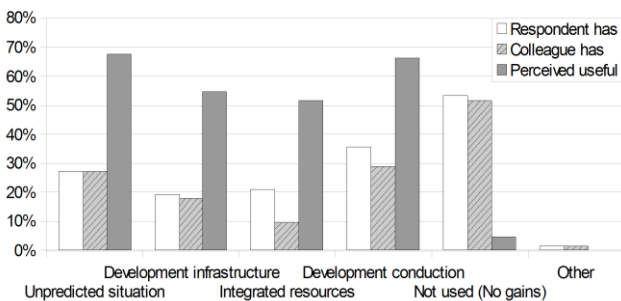


Figure 4. Respondent's application and perceived usefulness of applying the TD concept

**B. RQ2 - Which agile software development practices and processes are sensitive to TD?**

The RQ2 group of questions is focused on investigating the effects of agile software development practices and processes on TD. For this, initially the questionnaire presented a list of 11 agile development practices [11] and, for each of them, the interviewee should indicate how positively/negatively it could impact the TD in the project (RQ2.1). The results presented in Figure 5 demonstrate that practices used during implementation phase (simple design, TDD, coding standards, refactoring, continuous integration, and pair programming) are considered by more than half of the interviewees as having a positive or very positive effect on TD. More specifically, refactoring was indicated as the practice that has the most positive effect.

Afterwards, we asked participants about the effect of agile development processes on TD (RQ2.1). For this, a list of six processes [12] was considered. As we can see in Figure 6, all processes (iteration planning meetings, iterations, iteration backlog, iterations reviews/retrospectives, daily meetings, and product backlog) were considered to have very positive or positive effects on TD. The iteration process was considered the most positive among them.

We also asked if the combination of agile techniques that participants used were adequate for the team's or the project's management needs (RQ2.2) and if the techniques were able to cover all aspects that require management (RQ2.3). For singular practices, processes and their adoption rates, not a single combination could be identified for which the difference in their management or cover characteristics was statistically significant.

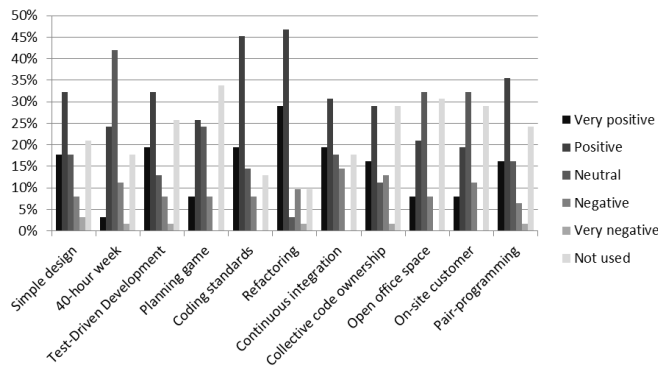


Figure 5. Perceived effect of agile software development practices on TD

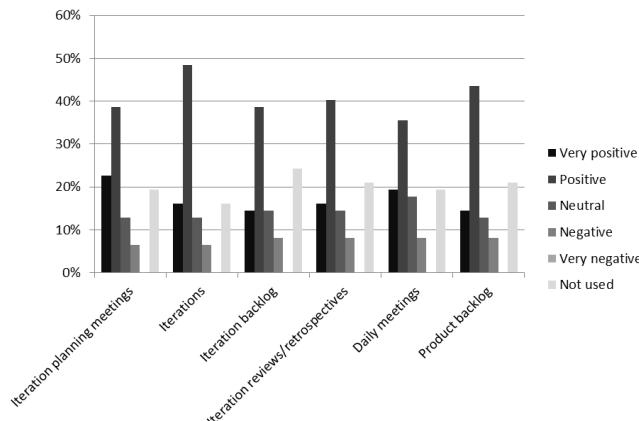


Figure 6. Perceived effect of agile development processes on TD

**Finding 1:** In general, agile practices have a positive effect on the TD. Among the analyzed practices, refactoring was considered to have the most positive effect.

**Finding 2:** In general, agile processes have a positive effect on the TD. Among the considered processes, iteration was indicated as having the most positive effect.

*RQ3 - How does TD manifest in the participants' work?*

The group of questions RQ3 is focused on the analysis of situations that TD affected the progress of projects in which the participants were involved. When asked about in which phase of the software development the TD was observed (RQ3.1), as we can see in Figure 7, 77% of respondents stated that the implementation phase is the most affected, followed by design phase. Although the testing phase was reported as the least affected, its percentage is still relevant.

We also investigated the causes that led to the occurrence of debt (RQ3.2). To do this, from a previously defined list of causes [2], the participant should indicate which of them he/she considered pertinent. In Figure 8, we can see that the causes most often indicated by participants were inadequate architecture and inadequate structure, followed by violation of best practices or style guides, and inadequate testing and documentation. This result is aligned with findings reported by Ernst *et al.* [6] that also pointed to problems in architecture as the main source of TD in software projects.

In addition to this question, when asked about the source of TD instances (RQ3.3), most participants (50%) stated that TD instances came from the legacy from an earlier team/individual who previously worked on the same project/product. 18% of the participants indicated that their source is in the legacy from an unrelated project/product of the organization, and 14% stated that the source is in the legacy from outside the organization. Only 18% of participants answered that the source is not from legacy activities.

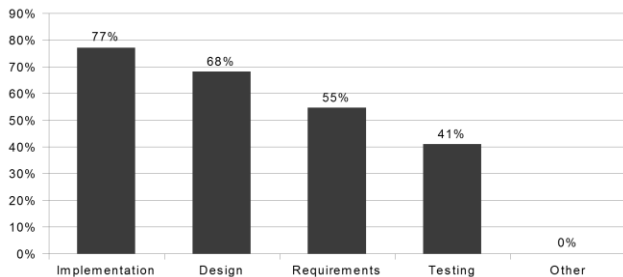


Figure 7. Distribution of TD by project phases

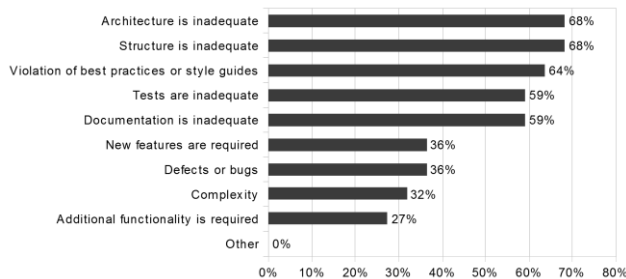


Figure 8. Indicated causes for concrete instances of TD

Then, when asked about the relationship between the continued development of a component and the size of the debt located in that component (RQ3.4), most of participants (82%) reported that the continued development would contribute to the increase in the size of the debt, while only 18% reported that this would lead to a reduction in the size of the debt. None of the respondents indicated that there would be a large decrease or no change.

Finally, when asked about the correlation between the size of a TD item and the effects that it causes in software development (RQ3.5), about 70% of the respondents answered that the size is directly proportional to the magnitude of the effects, 5% stated that it is not proportional, and another 5% answered that it is inversely proportional. Other 23% reported that the size is somehow proportional to effects magnitude. Thus, for most respondents, the larger is the size of a debt item, the greater is the effect it brings to the project.

**Finding 1:** The implementation phase is, usually, the most affected by TD.

**Finding 2:** Inadequate software architecture and internal structure are the main causes of TD.

**Finding 3:** Debt items tend to come from the legacy of a team/individual who previously worked on the same project/product.

**Finding 4:** Continued development in a software item contributes to the increase in TD's size on that item.

**Finding 5:** The larger the size of a TD item is, the greater is the effect it brings to the project.

#### IV. DISCUSSION

This work presented the results of a replicated survey in Brazil. For RQ1, we found that the concept of TD is already known by a good part of the population represented in this study. Regarding RQ2, we observed that, in general, agile software development practices and processes have a positive effect on TD. In this item we highlight the refactoring practice and the iteration process, which were considered as having the most positive effect. Finally, for RQ3, the data indicated that the implementation phase is the most affected by debt items, and problems associated with the architecture and internal structure of the software are the main causes of TD. These results justify, at a certain extension, the fact that agile practices that have a more positive effect on TD are directly related to coding activities.

Another result from RQ3 indicated that continued development in a software item contributes to the increase in TD in that item. This is an interesting result because if, on the one hand, continuous work on an item opens opportunities for improvements in its internal structure (that can lead to payment of debt items), on the other hand, if we do not explicitly manage TD, these opportunities can be lost and, as consequence, the debt size can reach higher levels. Finally, participants also reported that there is proportionality between debt size and the effects it brings to the project. These two

results reinforce the importance of making explicit the management of the TD items.

#### A. Relation to previous work

The results of this replication indicated that the population has different characteristics from the original study [9]: (i) Finland sees a majority in the smaller organization size categories and Brazil is in the middle ground with highs in medium and large categories; (ii) Finland had average iteration length of two to three weeks whereas Brazil is more evenly distributed (from 1 week to more than 2 months), and (iii) projects in Brazil tend to be longer and Finland's shorter. Thus, on the perspective of the population characterization, we could say that this replication contributes to the original study by expanding the sample from the organization spectrum. Besides, it also indicates that the obtained results reflect particularities of the Brazilian local scenario of agile software development. Concerning participants' level of experience, country-wise deviation is almost non-existent.

Regarding results for research questions, despite in general both executions pointed out to the same direction, we also could detect particularities. For example, respondents from both countries answered that common agile practices and processes are sensitive to technical debt. However, while in Brazil refactoring and iteration are considered as having the most positive effects on TD, in Finland, participants indicated coding standards and iteration reviews/retrospectives. A more detailed analysis of differences and similarities between the results obtained with the execution of the survey in Finland and its replication in Brazil is out of the scope of this paper, being part of the next steps of this research.

#### V. STUDY LIMITATIONS

Some limitations apply to this study. One of them is related to the cultural influence of the region where the survey was performed. Although the questionnaire was answered by participants from different companies of different sizes and based in different regions from Brazil, yet the "Brazilian way" of developing software may have influenced the responses. A detailed analysis considering this aspect is outside the scope of this paper, but it is a future work that will be carried out by the authors considering the data obtained with the execution of the survey in Finland and its replication in Brazil.

A second limitation that affects this study is related to the lack of control over the participants invited to participate in the research. It could happen that only developers interested in the TD area participate of the study. This might bias the results towards a more positive view of technical debt knowledge. However, about 50% of the respondents initially indicated that they were not familiar with the concept and thus we assume that this positive bias is not significant.

Finally, although the number of responses (62) can be considered good, yet the data cannot be generalized to represent practitioners from Brazilian software industry. Still, they provide valuable indicators on the research questions raised. Their analysis together with the data obtained from the original execution of the study in Finland will allow a greater level of confidence in the results.

#### VI. FINAL REMARKS

This work is aligned with a growing concern of the software engineering community: the replication of empirical studies. It contributes to the generation of knowledge in a given topic through the accumulation of evidence about the findings, thereby increasing the level of confidence in results [13].

Specifically, this replicated study investigated the perception of practitioners on TD concept, the effects of agile software development practices and processes on it, and how TD manifests itself in practice in the Brazilian software industry. The reached results, described in details on Sections III and IV, contribute to the improvement of the body of knowledge that has been built around the Technical Debt Landscape [3] [8]. The next steps of this research include a country level comparison of the obtained results.

#### ACKNOWLEDGMENT

This work was partially supported by the CNPq Universal grant 458261/2014-9, by the State of Bahia's SECTI-Fraunhofer-UFBA cooperation agreement 2012-1, and by the RESCUER project Grant: 490084/2013-3.

#### REFERENCES

- [1] C. Seaman & Y. Guo (2011), Measuring and Monitoring Technical Debt, *Advances in Computers* 82, 25-46.
- [2] P. Kruchten; R. Nord & I. Ozkaya (2012), Technical Debt: From Metaphor to Theory and Practice, *Software*, IEEE 29(6), 18-21.
- [3] C. Izurieta; A. Vetro; N. Zazworka; Y. Cai; C. Seaman & F. Shull (2012), Organizing the technical debt landscape, in *Managing Technical Debt (MTD)*, 2012 Third International Workshop on, pp. 23-26
- [4] N.S.R. Alves, T.S. Mendes, M.G. Mendonça, R.O. Spínola, and C. Seaman, Identification and management of technical debt: A systematic mapping study, *Information and Software Technology*, Volume 70, February 2016, Pages 100-121, ISSN 0950-5849.
- [5] R. C. Martin, *Agile software development: principles, patterns, and practices*. Prentice Hall PTR, 2003.
- [6] N.A. Ernst, S. Bellomo, I. Ozkaya, R.L. Nord, and I. Gorton. 2015. Measure it? Manage it? Ignore it? software practitioners and technical debt. In *Proc. of the 2015 10th Joint Meeting on Foundations of Software Engineering*. ACM, New York, NY, USA, 50-60.
- [7] S. McConnel, "Managing Technical Debt," *Construx Software*, Version 1. Available from: <http://www.construx.com>, 2008
- [8] N.S.R. Alves, R.S. Araújo, R.O. Spínola. A Collaborative Computational Infrastructure for Supporting Technical Debt Knowledge Sharing and Evolution. In: *Americas Conference on Information Systems*, 2015, Puerto Rico.
- [9] J. Holvitie; V. Leppanen & S. Hyrnsalmi (2014), Technical Debt and the Effect of Agile Software Development Practices on It - An Industry Practitioner Survey, in *MTD 2014*, pp. 35-42.
- [10] J. Holvitie, S.A. Licorish, R.O. Spínola, S. Hyrnsalmi, S.G. MacDonell, T.S. Mendes, J. Buchan, and V. Leppänen. Technical debt and agile software development practices and processes: An industry practitioner survey. *Information and Software Technology*, 2017, ISSN 0950-5849.
- [11] N. Kurapati, V. S. C. Manyam, K. Petersen, Agile software development practice adoption survey, in: *Agile processes in software engineering and extreme programming*, Springer, 2012, pp. 16{30.
- [12] D. West, T. Grant, Agile development: Mainstream adoption has changed agility 2 (41).
- [13] F. Shull, J.C. Carver, S. Vegas, and N. Juristo. 2008. The role of replications in Empirical Software Engineering. *Empirical Software Engineering*. 13, 2 (April 2008), 211-218.

# Timing Analysis for Microkernel-based Real-Time Embedded System

Rongfei Xu, Li Zhang

*School of Computer Science and Engineering  
Beihang University  
Beijing, China*

Ning Ge

*School of Software  
Beihang University  
Beijing, China  
gening@buaa.edu.cn*

Jing Jiang

*School of Computer Science and Engineering  
Beihang University  
Beijing, China*

**Abstract**—Currently, more and more application-specific operating systems (ASOS) are applied in real-time embedded systems. With the development of microkernel technique, the ASOS is usually customized based on the microkernel using the configurable policy, which has various alternatives. In the design of the real-time embedded system (RTES) based on such ASOS, evaluating its timing performance at the early design stage is helpful to guide the designer towards choosing the most appropriate policy. However, the existing works lack a uniform approach to support analyzing the various alternatives of the configured policy. To solve this problem, this paper presents a general-purpose timing analysis approach for the ASOS-based RTES. In the analysis, a timing analysis tree is proposed to characterize the tasks and the ASOS in the RTES. Then, each of the alternative policies in the ASOS is refined by the uniform execution rules in the tree. Finally, the task's response time under the various alternative policies is analyzed by a traversal of the timing analysis tree using a uniform way. In the case study, we take the scheduling policy as an example to show the use of our approach on a real-life robot controller system.

**Index Terms**—real-time embedded system, microkernel-based RTOS, application-specific operating system, alternative policy, timing analysis

## I. INTRODUCTION

In real-time embedded systems (RTES), the real-time operating system (RTOS) is usually used to manage the tasks in the system, and directly impacts their timing performance [1]. The RTESs in various domains may suffer from the general-purpose operating system (OS) due to their specific characteristics. Currently, many works are aimed at the application-specific operating systems (ASOSs) to enhance the performance of a certain application [2], e.g., microkernel architectures are representative ASOSs. Nowadays, more and more practical RTOSs are designed based on a microkernel, such as QNX, Integrity, and FreeRTOS. A microkernel [3] is a minimalistic kernel that contains the near-minimum amount of functions and features required to implement an OS, it adopts the "separation of mechanism and policy" principle. Such principle makes it convenient to build arbitrary OS services using the configurable policy. When customizing an ASOS, every configurable policy has various alternatives, each of

which has a different influence on the response time of the task. Hence, in this work, we are interested in the timing analysis of the design of the RTES, which is implemented on a customized ASOS based on the microkernel with various alternatives for the configurable policy.

In the real-time systems, the timing analysis approaches can be divided into two categories: dynamic and static. The dynamic approaches, which include the simulation and model checking, suffer from the efficiency problem when applied to the case mentioned here. For the simulation, each alternative policy requires generating a policy-dedicated simulation model, which is not feasible for a general-purpose. For the model checking, it also needs to concern the policy-dedicated rules throughout the task model [4]. Thus, we resort to the static analysis approaches, which include three classes [5]: structure-based, path-based, and the technique using implicit path enumeration (IPET). Both the path-based approach [6] and the IPET [7] are inadequate for our case due to they do not consider the OS. As for the structure-based approach, it can only support the specific function of the OS [8], [9], and is inadequate to analyze the various alternative policies here in a general purpose way.

In this paper, we propose a timing analysis approach specific for the RTES based on a microkernel-based ASOS, which is customized by the configurable policy that has various alternatives. In order to perform the timing analysis for the various alternatives uniformly, we first propose a structure of timing analysis tree, which is used to characterize the tasks and the ASOS in the RTES. Then, we define a canonical form of the execution rules to refine the various alternatives in the timing analysis tree. Based on the execution rules, we finally propose a general-purpose analysis technique by a traversal of such timing analysis tree for the various alternatives. In the case study, we take the scheduling policy as an example to show the use of our approach on a real-life robot controller system. Comparing with the state-of-the-art methods, the superiority of our approach is that it simplifies the analysis by fixing the tasks and the ASOS mechanisms, and only replacing the part of the configurable policy.

This paper is organized as follows: Sect. II discusses the related works; Sect. III introduces the background and overview of our approach; Sect. IV proposes the timing

This paper is supported by the National Natural Science Foundations of China (No. 61672078 and No. 61732019)  
DOI:10.18293/SEKE2018-095

analysis approach; Sect. V evaluates our approach on a real-life case; and Sect. VI gives some concluding remarks and perspectives.

## II. RELATED WORKS

Currently, the timing analysis of the RTES includes two different classes of methods [5], that is the dynamic methods and the static methods.

The dynamic methods rely on the simulation or the model checking. For the simulation-based methods, the works [10], [11] mapped the MARTE model to the SymTA/S model for timing analysis based on formal scheduling analysis techniques and symbolic simulation; the work [12] proposed a simulation-based timing analysis depending on a more detailed system model, which described the execution control flow at the code level. When used in our case, the simulation-based methods need a model transformation (or refinement) for each alternative policy, which is inflexible. For the model checking, the work [13] presented an analysis method for the worst-case execution time (WCET) using UML-MARTE model checker, which was aimed at detecting wrong software designs and refined the correct ones with respect to WCET; the work [14] mapped the activity diagram of UML into the priority time Petri net (PTPN) to enhance the formal schedulability test of given real-time tasks; The work [15] mapped the workload model of real-time systems into a Petri Nets formalism to generate all transactions for the timing analysis. However, as for our case, the model checking method needs to specify the policy-dedicated rules throughout the task model, which is flexible or even impossible.

The static methods include three classes [5]: structure-based, path-based, and techniques using implicit path enumeration (IPET). In the path-based method [6], the execution time is determined by analyzing the paths in the task. In IPET [7], the control flow and the basic-block execution time are combined into the constraints to analyze the execution time of the task. Both the path-based method and the IPET don't consider the OS's functions in the execution of the task. In the structure-based method [16], the execution time is analyzed in a bottom-up traversal of the syntax tree of the task. The syntax tree takes the functions or subtasks of a task as the nodes, so the interactions between the nodes can be used to concern the OS's functions, such as synchronization [17], instruction cache locking [9], etc. However, the structure-based method can't support analyzing the various realizations of the function in a general purpose way. For example, the work [17] proposed three analysis methods for the three instruction cache locking strategies, i.e. static locking, semi-dynamic locking and dynamic locking.

## III. BACKGROUND AND OVERVIEW

A real-time embedded application is usually designed as a set of tasks managed by the RTOS [18], i.e. the ASOS here. The microkernel-based ASOS includes three basic mechanisms that cover the essential functions of the microkernel, i.e. the task scheduling, the inter-process communication (IPC),

and the resource access [3]. Each mechanism can be extended using a set of alternative policies. The task consists of a sequence of functional blocks with some system calls [19]. The system call is realized by the basic system calls in the ASOS. The functional block is used to realize an independent function and composes the execution sequence of a task [19].

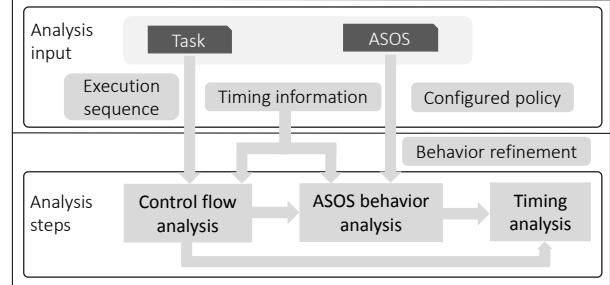


Fig. 1. Overview of our approach

The overview of our approach is shown in Fig. 1. The RTES design includes the tasks and the ASOS. The response time of each task is analyzed based on its control flow, which is characterized by the execution sequence of the task. Such control flow is influenced by the ASOS behavior, which varies with different configured policy. Here, we propose to refine the ASOS behavior at the analysis stage. Besides, the timing information needs to be specified for the tasks and the ASOS. Specifically, the worst-case execution time (WCET) is pre-defined for each functional block in the tasks and each basic system call in the ASOS. For each alternative policy, the timing analysis of the RTES design is implemented by combining the control flow and the ASOS behavior to analyze the tasks' response time.

## IV. TIMING ANALYSIS APPROACH

In our approach, we define an extensible timing analysis tree (ETAT) to characterize the task and the ASOS, where the alternative policy can be replaced flexibly (i.e. extensible). If a new policy is configured, the only part needs to be modified in the ETAT is the policy node together with its child node.

### A. Extensible Timing Analysis Tree (ETAT)

In this section, we first define the semantics for the extensible timing analysis tree (ETAT); then, we propose a canonical form to define the execution semantics for the ETAT, which is used to refine the ASOS behavior to perform the timing analysis. Based on the proposed canonical form, we introduce the execution rules for the three mechanisms in ASOS, i.e. scheduling, IPC and resource access.

1) *Definition of ETAT*: In the RTES, each task is modeled as an ETAT, which consists of a set of nodes and edges. The node is defined as  $TreeNode = (time\_cost, component\_attribute)$ , where the  $time\_cost$  attribute records the time cost of the represented component, the  $component\_attribute$  attribute characterizes the attributes of the represented component. There are three types of nodes in the ETAT as follows:



- *object* node specifies the tasks and the functional blocks.
- *operation* node specifies the ASOS behaviors, including mechanisms and configurable policies
- *parameter* node specifies the basic system calls and execution rules for the ASOS behavior.

The various relationships between the nodes are defined as different types of edges in the ETAT. Each type of edge can only exist between a pair of certain type of nodes. The are five types of edges, which are listed as follows:

- *use*: A task or a functional block uses the mechanism or the policy in the ASOS; The execution rules and the basic system calls are used by the mechanism or the policy.
- *realize*: A policy is realized based on the mechanism.
- *consist*: A task consists of a set of functional blocks.
- *sequence*: The successor of a functional block in the execution sequence is its sub-sequence.
- *operate*: The mechanism or the policy operates on the task or the functional block.

2) *Definition of the Execution Semantics*: The ASOS behavior is refined by the execution rules in the ETAT, which define the operating actions and the timing actions for the behavior. Specifically, the operating action expresses the operation for this behavior; the timing action indicates there is a time cost for the operation. A canonical form for the execution rules is defined as

$$State \xrightarrow{[Condition]/Action} State' \quad (1)$$

where *State* represents the current state of a task, *Condition* means the condition affecting the execution of the task, and *Action* is the operating or the timing action for the task triggered by the satisfaction of conditions.

The execution rules for *scheduling* mechanism are defined as shown in Fig. 2.

- Four basic states of a task (running, ready, blocked, suspended) are represented by *St\_Run*, *St\_Ready*, *St\_Block* and *St\_Suspend*, respectively.
- The set of conditions consists of *Cond\_Preempted*, *Cond\_First\_Run*, *Cond\_Wait\_Event*, *Cond\_Event\_Arrive*, and *Cond\_Time\_Out*.
  - *Cond\_Preempted* represents the condition that makes a task be preempted.
  - *Cond\_First\_Run* represents that the task is selected to run first among the tasks in the ready queue.
  - *Cond\_Wait\_Event* represents that the task is waiting for an event.
  - *Cond\_Event\_Arrive* represent that the waited event arrives.
  - *Cond\_Time\_Out* represents that the waiting is time-out.
- The operating actions (i.e. running, readying, blocking and suspending) are represented by *Act\_Run*, *Act\_Ready*, *Act\_Block* and *Act\_Suspend* respectively, and the timing action is defined as *Act\_Timing*.

The execution rules for *resource access* mechanism are defined as shown in Fig. 3.

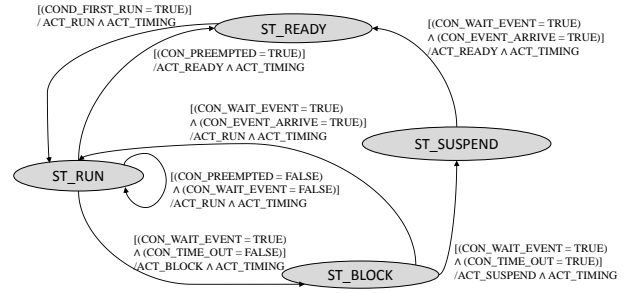


Fig. 2. Execution rules for scheduling

- Two basic states of *St\_Run* and *St\_Block* are involved.
- The conditions of *Cond\_Request\_Resource* and *Cond\_Req\_Resource\_Available* are used.
  - *Cond\_Request\_Resource* represents that the task requests a resource during its execution.
  - *Cond\_Req\_Resource\_Available* represents that the requested resource is available right now.
- The operating actions include *Act\_Run*, *Act\_Block*, *Act\_Check\_Resource*, *Act\_Get\_Resource* and *Act\_Timing*. Among them, *Act\_Check\_Resource* is to check whether the resource is available, *Act\_Get\_Resource* is to obtain the available resource.

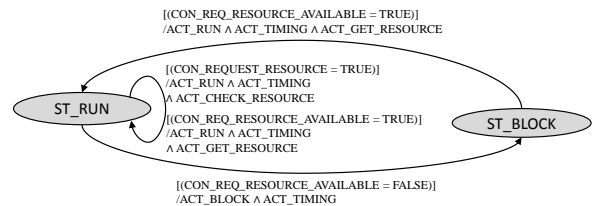


Fig. 3. Execution rules for resource access

The execution rules for *IPC* mechanism are defined as shown in Fig. 4.

- Two basic states of *St\_Run* and *St\_Block* are involved.
- The conditions of *Cond\_Request\_Communication* and *Cond\_Req\_Connect\_Setup* are used.
  - *Cond\_Request\_Communication* represents that the task requests a communication with other task during its execution.
  - *Cond\_Req\_Connect\_Setup* represents that the connection for the requested communication is set up.
- The operating actions include *Act\_Run*, *Act\_Block*, *Act\_Connect\_Setup*, *Act\_Communicate* and *Act\_Timing*. Among them, *Act\_Connect\_Setup* is to set up the connection, *Act\_Communicate* is to communicate with other task.

## B. Timing Analysis for ETAT

The response time of a task consists of the scheduling time, the interaction time (with other functional blocks), and the WCET of the functional blocks in this task. Both the

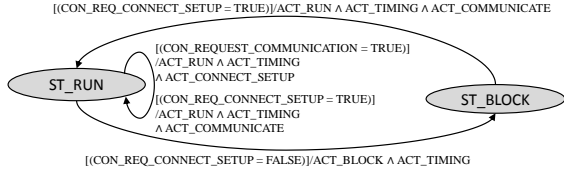


Fig. 4. Execution rules for IPC

scheduling time and the interaction time rely on the ASOS behavior, and consist of the WCET of the basic system calls. Such response time is analyzed by a traversal of the ETAT of this task. During the traversal, the time cost of the root node (i.e. the task node) indicates the current execution time of the task, and is updated once the time cost of the scheduling node or that of each functional block node is worked out. The time spent at each object node of functional block is analyzed based on its operation child-node. The time spent at each operation node (include scheduling and interaction) is analyzed based on its parameter child-node. When the ETAT is completely traversed, the time cost of the root node indicates the response time of the task.

```

function timing_analysis_process (T)
1: visit the root node rn of T
2: if (rn.traversalsFinished == true) then
3:   return
4: else
5:   visit the scheduling child-node sc of rn
6:   OperationNodeAnalysis_forScheduling
7:   update rn.time_cost
8:   if (task.state != "executable") then
9:     return
10:  else
11:    visit the functional block child-node fb of rn
12:    set fb as the functional block node to be analyzed fb_ta
13:    while (fb_ta != null)
14:      if fb_ta.executionCondition is satisfied then
15:        ObjectNodeAnalysis_forFunctionalBlock
16:        update rn.time_cost
17:      else
18:        return
19:      end if
20:      set the functional block child-node of fb_ta as fb_ta.
21:      visit the node fb_ta
22:    end while
23:    set rn.traversalsFinished = true
24:    return rn.time_cost as the response time
25:  end if
26: end if
  
```

Fig. 5. Timing analysis process

Given an ETAT  $T$ , the timing analysis process is shown in Fig. 5. First, visit the root node of  $T$  to check whether  $T$  is completely traversed (L. 1). If not, visit the scheduling child-node of the root node to check whether the task is executable

(L. 5). Then, analyze the time cost of the scheduling child-node, and update the time cost of the root node (L. 6,7). If the task is executable, we visit the functional block child-node (say  $fb$ ) of the root node, then visit the functional block child-node (say  $fb'$ ) of  $fb$ , then visit the functional block child-node of  $fb'$ , ..., until all functional block nodes are visited (L. 20). For each functional block node, we check its execution condition and analyze its time cost based on the operation node (if exists) and the parameter node (the analysis procedure will be introduced later), then update the time cost of the root node (L. 14-16). When all the functional block nodes are visited, the task is set as completely traversed, the time cost on the root node indicates the response time of the task.

Next, we specifically introduce the scheduling node and the functional block node mentioned above. The time cost of the scheduling node is the time spent at the scheduling operation. The time cost of the functional block node includes the time spent at the object itself and at the interaction operation (if exists). Therefore, we focus on the two types of nodes, i.e. the *object* node and the *operation* node. According to the definition of ETAT, the basic structures of *object* node and *operation* node are summarized in Fig. 6. For the *object* node, it has a child node of the *object* type with a *consist* (for task) or *sequence* (for functional block) edge between them. If the *object* node has a scheduling operation or an interaction operation, an *operation* node is generated as its another child node with the *use* edge. For the *operation* node, it has a child node of *parameter* with the *use* edge. If the *operation* node has an extended operation (for policy), the *realize* edge is used to link them. If the *operation* node has an other operand, a child node of the *object* type is generated for the operand with the *operate* edge.

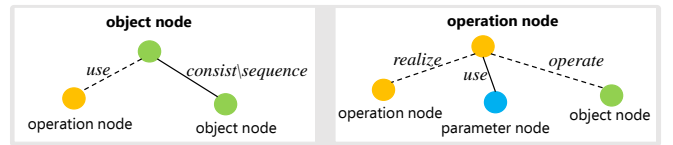


Fig. 6. Basic structure of *object* node and *operation* node

The timing analysis for the *object* node and the *operation* node is presented as follows. As the *object* node of the task is the root node to record the time cost, we focus on the *object* node of the functional block here. For ease of illustration, we call such a child node that has a *use* edge with its father node as the *use* child-node in brief (the same for other edges).

- C1: For the *object* node of functional block, its time cost includes the time spent at itself and at its *use* child-node (if exists). The time spent at the functional block itself is specified by the WCET value in its *component\_attribute*. The *use* child-node is actually the *operation* node, whose time cost is analyzed by the way in C2.
- C2: For the *operation* node, its time cost includes the time spent at its *use* child-node, *realize* child-node (if exist) and *operate* child-node (if exist). The time spent at the *use* child-node is the time cost of the system call, which is

analyzed based on the execution rules (the timing actions particularly) and the WCET of basic system calls. If this *operation* node has a *realize* child-node, the *realize* child-node is actually an *operation* node, whose time cost is analyzed by the same way. If this *operation* node has other operands except for its father node (as the *operation* node is used by its father node, the father node is one operand of this operation), the *operate* child-node is actually an *object* node, whose time cost is analyzed by the same way as C1.

## V. CASE STUDY

### A. Experimental setup

In this section, we will illustrate the application of our approach to a real-life robot controller system [20]. The robot controller system (RCS), which consists of three tasks, is used to keep the robot operating normally. Among the tasks, the *balance* task is to keep the balance of the robot by calculating the input from the gyroscope and the inclinometer; the *navigation* task is to avoid obstacles during the process of going to the destination; the *remote* task is to receive a remote command via the infrared. The services of the infrared sensor, the gyroscope and the inclinometer are realized by the interrupt service routines (ISR), which are corresponding to *infrared\_isr*, *gyro\_isr* and *inclino\_isr* respectively. To implement such RCS, we use the  $\mu$ C/OS-II kernel [21] to configure the ASOS. The  $\mu$ C/OS-II kernel implements a static priority scheduling policy, and has an optional policy of round robin scheduling. In this case study, we analyze the timing performance of the three tasks in the RCS to assess these two scheduling policies.

For the ASOS, the WCET of the basic system calls in the  $\mu$ C/OS-II kernel given in [22] is used in this case study. About the two scheduling policies in the ASOS, the time slice of the round robin (RR) scheduling is set as 10 thousands CPU cycles, the priorities (P) for the three tasks in the static priority (SP) scheduling are set as:  $P(balance) = 4$ ,  $P(navigation) = 6$ ,  $P(remote) = 5$ . For the tasks, their timing requirements are represented by the deadline (D), and set as (in one thousand CPU cycles):  $D(balance) = 200$ ,  $D(navigation) = 40$ ,  $D(remote) = 4000$ . Within the tasks, the functional blocks (FBs) together with their WCET are set as shown in Table. I.

### B. Experimental process and results

First, we define the execution rules for the two scheduling policies. As shown in Fig. 7, these execution rules refine the preempted condition in the scheduling mechanism (as shown in Fig. 2). Specifically, for the SP scheduling, an arbitrary task  $T$  is preempted when there exists a ready task with a higher priority than  $T$ ; for the RR scheduling, the task  $T$  is preempted when the time slice for  $T$  is used up. It should be noted that the  $CON\_PREEMPTED$  in the execution rules of scheduling mechanism (as shown in Fig. 2) is set by the actions of  $ACT\_SET\_PREEMPTED\_TRUE$  or  $ACT\_SET\_PREEMPTED\_FALSE$  in the execution rules of the two scheduling policies.

TABLE I  
WCET SETTINGS FOR FUNCTIONAL BLOCKS (IN ONE THOUSAND CPU CYCLE)

Task	Function block	WCET
balance	Initialization	5
	GetInfoFromGyro	10
	GetInfoFromInclino	10
	Calculation	30
	KeepBalance	50
navigation	Initialization	1
	SendDetector	3
	FindObstacle	8
	AvoidObstacle	5
remote	Initialization	10
	GetInfoFromInfrared	800
	ExecuteCommand	3000

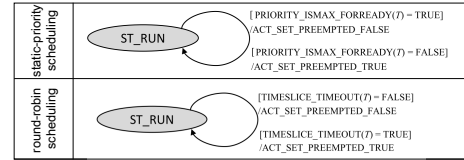


Fig. 7. Execution rules for two alternative scheduling policies

Then, we construct the timing analysis trees for the three tasks as shown in Fig. 8, where the *object* nodes, the *operate* nodes and the *parameter* nodes are represented by the colors of green, orange and blue respectively in each timing analysis tree. As space is limited, the attributes of each node in the trees are not presented.

After the timing analysis, the response time of each task under the two scheduling policies is presented in Table. II. As seen, the static priority scheduling can meet the deadline of the tasks, while the round robin scheduling can not. This case study takes the two scheduling policies as an example to illustrate the feasibility of our approach. Without loss of generality, any other scheduling policies can also be analyzed based on the timing analysis trees in Fig. 8 by defining their execution rules.

TABLE II  
TIMING ANALYSIS RESULTS (IN MILLISECONDS)

Task	SP scheduling	RR scheduling
<i>balance</i>	158	304
<i>navigation</i>	3845	5091
<i>remote</i>	36	47

## VI. CONCLUSION AND PERSPECTIVE

In the domain of the real-time embedded system, more and more application-specific operating systems (ASOS) are customized based on the microkernel using the configurable policy. The existing methods usually need an individual timing analysis for each alternative policy. To simplify the analysis, we propose a general-purpose timing analysis approach for such ASOS-based RTES design. A real-life robot controller system is used as a case study to show the feasibility of our

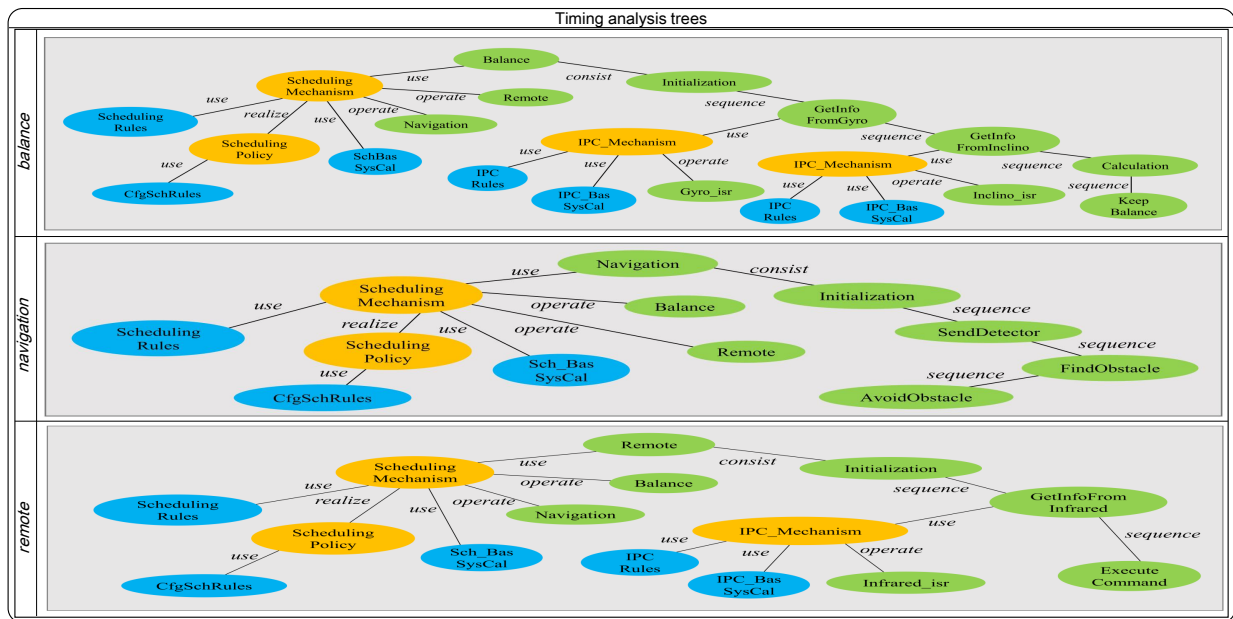


Fig. 8. Timing analysis tree for the tasks

approach. Currently, our approach only supports the configurable policies of the three basic aspects, i.e. scheduling, inter-process communication, and resource access. With the RTES is becoming more and more complex, the more functions are needed by the ASOS, such as network management, file system, etc. In the near future, we will extend our approach to support more configurations in the ASOS.

## REFERENCES

- [1] J. Schneider, "Why you cant analyze rtoss without considering applications and vice versa," *2nd WS Worst-Case Execution-Time Analysis*, 2002.
- [2] Y. Sun, Y.-F. Ai, and G.-S. Yang, "An optimal scheduling algorithm for vehicular application specific operating systems," in *Computer Science and Software Engineering, 2008 International Conference on*, vol. 2. IEEE, 2008, pp. 184–189.
- [3] J. Liedtke, "Towards real microkernels," *Communications of the ACM*, vol. 39, no. 9, pp. 70–77, 1996.
- [4] E. M. Clarke, W. Klieber, M. Novek, and P. Zuliani, *Model Checking and the State Explosion Problem*. Springer Berlin Heidelberg, 2011.
- [5] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, R. Heckmann, T. Mitra *et al.*, "The worst-case execution-time problem: overview of methods and survey of tools," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 7, no. 3, p. 36, 2008.
- [6] F. Stappert and P. Altenbernd, "Complete worst-case execution time analysis of straight-line hard real-time programs," *Journal of Systems Architecture*, vol. 46, no. 4, pp. 339–355, 2000.
- [7] A. Ermedahl, "A modular tool architecture for worst-case execution time analysis," Ph.D. dissertation, Acta Universitatis Upsaliensis, 2003.
- [8] G. Aupy, C. Basseur, and L. Marchal, "Dynamic memory-aware task-tree scheduling," in *Parallel and Distributed Processing Symposium (IPDPS), 2017 IEEE International*. IEEE, 2017, pp. 758–767.
- [9] T. Liu, M. Li, and C. J. Xue, "Instruction cache locking for multi-task real-time embedded systems," *Real-Time Systems*, vol. 48, no. 2, pp. 166–197, 2012.
- [10] M. Hagner and U. Goltz, "Integration of scheduling analysis into uml based development processes through model transformation," in *Computer Science and Information Technology (IMCSIT), Proceedings of the 2010 International Multiconference on*. IEEE, 2010, pp. 797–804.
- [11] M. Hagner and M. Huhn, "Tool support for a scheduling analysis view," in *MARTE workshop at DATE*, vol. 8, 2008, pp. 41–46.
- [12] M. Bohlin, Y. Lu, J. Kraft, P. Kreuger, and T. Nolte, "Simulation-based timing analysis of complex real-time systems," in *Embedded and Real-Time Computing Systems and Applications, 2009. RTCSA'09. 15th IEEE International Conference on*. IEEE, 2009, pp. 321–328.
- [13] N. Ge, M. Pantel, and B. Berthomieu, "A flexible wcet analysis method for safety-critical real-time system using uml-marte model checker," 2016.
- [14] Y. H. Kacem, A. Mahfoudhi, A. Magdich, C. Mraidha, and W. Karamti, "Using mde and priority time petri nets for the schedulability analysis of embedded systems modeled by uml activity diagrams," in *Engineering of Computer Based Systems (ECBS), 2012 IEEE 19th International Conference and Workshops on*. IEEE, 2012, pp. 316–323.
- [15] M. Naija, S. B. Ahmed, and J.-M. Bruel, "New schedulability analysis for real-time systems based on mde and petri nets model at early design stages," in *Software Technologies (ICSOF), 2015 10th International Joint Conference on*, vol. 1. IEEE, 2015, pp. 1–9.
- [16] A. Colin and I. Puaut, "Worst case execution time analysis for a processor with branch prediction," *Real-Time Systems*, vol. 18, no. 2-3, pp. 249–274, 2000.
- [17] R. Simmons and D. Apfelbaum, "A task description language for robot control," in *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, vol. 3. IEEE, 1998, pp. 1931–1937.
- [18] Y. Harada, K. Abe, M. Yoo, and T. Yokoyama, "Aspect-oriented customization of the scheduling algorithms and the resource access protocols of a real-time operating system family," in *Smart City/SocialCom/SustainCom (SmartCity), 2015 IEEE International Conference on*. IEEE, 2015, pp. 87–94.
- [19] F. Verdier, B. Miramond, M. Maillard, E. Huck, and T. Lefebvre, "Using high-level rtos models for hw/sw embedded architecture exploration: case study on mobile robotic vision," *EURASIP Journal on Embedded Systems*, vol. 2008, no. 1, p. 349465, 2008.
- [20] T. Braunl, "Eyebot: a family of autonomous mobile robots," in *Neural Information Processing, 1999. Proceedings. ICONIP'99. 6th International Conference on*, vol. 2. IEEE, 1999, pp. 645–649.
- [21] Micrium, "μc/os-ii real-time kernel," <http://www.micrium.com/products/>, 2017.
- [22] M. Lv, N. Guan, Y. Zhang, R. Chen, Q. Deng, G. Yu, and W. Yi, "Wcet analysis of the μc/os-ii real-time kernel," in *Computational Science and Engineering, 2009. CSE'09. International Conference on*, vol. 2. IEEE, 2009, pp. 270–276.

# Schedulability Analysis of Real-time Tasks with Precedence Constraints

Rongfei Xu, Li Zhang

School of Computer Science and Engineering  
Beihang University  
Beijing, China

Ning Ge

School of Software  
Beihang University  
Beijing, China  
gening@buaa.edu.cn

Xavier Blanc

LaBRI, UMR5800  
University of Bordeaux, Bordeaux INP, CNRS  
Talence, France

**Abstract**—The timing requirements of real-time systems can be guaranteed by the well-designed scheduling. The analysis of such scheduling inputs an abstract task model of the system and outputs a diagnostic regarding the practicability of the timing requirements. Task models have evolved from periodic models to more sophisticated graph-based ones, among which the digraph real-time (DRT) task model is the most applicable because of its good expressiveness and analysis efficiency. However, the DRT model can't support the precedence constraints within or between tasks. In this paper, we propose a new task model, called the DRTPC model, that extends the DRT model to support the precedence constraint. Further, based on our model, we present a uniprocessor schedulability analysis algorithm for the static priority scheduling, and introduce an optimization technique to improve the analysis efficiency. Our experiments show that, despite the high computational complexity of the problem, our approach scales very well for large sets of tasks with precedence constraints.

**Index Terms**—real-time system, schedulability analysis, task model, precedence constraint, static priority scheduling

## I. INTRODUCTION

The timing requirements of real-time systems can be guaranteed by the well-designed scheduling. The analysis of such scheduling is to assess the schedulability regarding the practicability of the timing requirements [1], i.e. the system's tasks can complete by the deadline. In the design of real-time systems, an abstract task model is usually used as an input for the schedulability analysis, and specifies the tasks' timing constraints (duration, start constraint, etc.) [2]. The precedence constraints within or between tasks, which commonly exist in real-time systems [3], [4], directly affect the schedulability and need to be concerned in the task model.

Let us take a simple example of a robot controller system to explain the precedence constraint. This system consists of three periodic tasks: the *navigation* task is to go to the destination by continuous movements and to avoid the obstacles when finding them, the *detect* task is to detect the location of the obstacles, and the *balance* task is to keep the balance of the robot. The *navigation* task needs the obstacle's location to do the job of finding the obstacle (*FO*). The location is collected by the job of receiving the location of an obstacle (*RL*) in the

*detect* task. The job of computing the adjustment for a balance (*CA*) in the *balance* task needs the data from the inclinometer, which is collected by the job of reading the inclinometer (*RI*) in the same task. Finally, the adjustment is used by the job of controlling balance (*CB*) to balance the robot. There are then three precedence constraints: *RL* precedes *FO*, *RI* precedes *CA*, and *CA* precedes *CB*.

Since the well-known Liu and Layland task model [5] appeared, a large number of task models, ranging from the relatively simple periodic and sporadic ones to the more complex graph-based ones, have been proposed [2]. There is a contradicting goal of expressiveness and analysis efficiency for these task models. For example, the Petri net [6] and the timed automata [7] are powerful to allow accurate modeling and easy to support the precedence constraint, but their analyses are based on the model checking, which can result in the exponential computational complexity. On the other hand, some works have specified the precedence constraint by extending the tractable task models, such as the recurring real-time task model [8], the sporadic task model [9], the dynamic offsets task model [10], [11], etc. Although these extended task models have a relatively good analysis efficiency, they are limited to the specific tasks and not for general-purpose ones. Therefore, the existing task models supporting the precedence constraint are inadequate. Due to good expressiveness and analysis efficiency, the currently proposed digraph real-time (DRT) task model [12] is promising according to a thorough survey [2]. Specifically, the DRT model can specify both the sporadic and periodic tasks, and is tractable (i.e. in pseudo-polynomial time) to be analyzed for large sets of tasks. However, the DRT model is based on the assumption that the tasks are independent of each other, so it is not yet to support the precedence constraint.

In this paper, we extend the DRT model by specifying the precedence constraint to get a new task model called DRTPC (Digraph Real-Time task model with Precedence Constraint). Further, we present a uniprocessor schedulability analysis algorithm for the static priority scheduling in the DRTPC model. This technique is capable of analyzing the schedulability by considering the interferences caused by both the priority-based preemption and the precedence constraint. In addition, we introduce an optimization technique for the schedulability

This paper is supported by the National Natural Science Foundations of China (No. 61672078 and No. 61732019)  
DOI:10.18293/SEKE2018-116

analysis to improve its efficiency. Our experiments show that, the proposed approach scales very well for large sets of tasks with precedence constraints.

In the remainder of this paper, we first discuss the related works in Sect. II. Then, we describe the DRTPC model in Sect. III and present the schedulability analysis algorithm for DRTPC in Sect. IV. Finally, the efficiency and scalability of our approach are evaluated in Sect. V, and Sect. VI gives some concluding remarks and perspectives.

## II. RELATED WORKS

In the past decades, the abstract task models that specify the tasks' timing constraints have been studied intensively in the real-time scheduling [1], such as the multiframe (MF) task model [13], generalized multiframe (GMF) task model [14], recurring real-time (RRT) task model [15], etc. The DRT model [12] considered in this paper is a generalization of the above models. Specifically, the DRT model can specify the branching and loop structures in real-time systems, which makes it capable of modeling both the sporadic and periodic tasks. Besides, some efficient methods of schedulability analysis have been proposed for the DRT model [16]. However, the DRT model can only support the independent tasks.

Currently, some works have been done to concern the dependence between tasks in the DRT model, such dependencies include three classes: inter-release time constraint [17], shared resource [18], [19] and synchronization [20], [21]. Specifically, the works [17] and [20] extended the edge in the DRT model to specify the global inter-release separation constraint and the synchronous execution respectively. The works [18], [19] and [21] extended the vertex in the DRT model to specify the maximal duration of resource access, the semaphore that guards the shared resource, and the synchronization operation respectively. However, it lacks an approach to concern the dependence caused by the precedence constraint, which is considered in this paper for the DRT model.

In the context of formal modeling and analysis, the precedence constraint is supported by two classes of works. The first class is based on the more expressive task models. For example, the work [7] specified the timed automata to deal with the precedence and resource constraints between the real-time tasks; the work [6] identified the precedence constraint properties of Petri net and tested whether it was feasible to execute a workflow with the specified temporal constraints. Due to the analysis complexity of automata or Petri net, the above-mentioned works suffer from the state space-explosion problem. The second class extends a simple task model to support the precedence constraint. For example, the work [9] considered the real-time system with the implicit precedence constraints between the intra-task jobs based on the sporadic task model; the work [8] represented the recurrent precedence-constrained tasks to be executed on multiprocessor platforms, where each recurrent task was modeled by a directed acyclic graph (DAG); the works [10], [11] addressed the schedulability analysis of the tasks with precedence relations in the distributed real-time systems based on the model of tasks with dynamic

offsets, which was specific for the distributed systems; the work [22] proposed an approach to scheduling the tasks with pipeline precedence constraints in the distributed real-time systems, which were described by the directed acyclic graph (DAG). Due to limited expressiveness of the task models to be extended, the above works can't support the general purpose tasks. Therefore, it is necessary to extend the DRT model to support the precedence constraint.

## III. TASK MODEL

In this section, we first introduce the DRT model, then define our DRTPC model to support the precedence constraint based on the DRT model, and explain its semantics.

### A. DRT Model

A real-time system is usually designed as a set of tasks, each of which consists of a sequence of functional blocks (called jobs here) [23]. A task model characterizes a task by the execution sequences and the timing constraints of its jobs. According to the definition of DRT model in [12], each task is characterized by a directed graph  $G(T)$ . The vertices of  $G(T)$  represent the jobs in the task. Each vertex is labeled by an ordered pair of  $(WCET, RD)$ , which represents the worst-case execution time (WCET) demand and the relative deadline of the corresponding job respectively. The directed edges of  $G(T)$  represent the orders (from start to end) in which the jobs are released. Each edge is labeled by the inter-release interval time between two jobs. In such graph  $G(T)$ , a vertex may have multiple edges or a loop edge, which makes the DRT model can specify the branching and loop structures.

*Example 1:* For the example of the robot controller, it consists of three tasks. For the *navigation* task, it includes three jobs (go to the destination *GD*, find an obstacle *FO* and avoid the obstacle *AO*) and four release orders (from *FO* to *AO*, from *AO* to *GD*, from *GD* to *FO* and to itself). For the *detect* task, it includes two jobs (send a detection *SD* and receive the location of the obstacle *RL*) and two release orders (from *SD* to *RL*, and from *RL* to *SD*). For the *balance* task, it includes three jobs (read the inclinometer *RI*, compute the adjustment *CA*, and control the balance *CB*) and three release orders (from *RI* to *CA*, from *CA* to *CB*, and from *CB* to *RI*). If we don't consider the precedence constraints, the DRT model of this example is characterized as shown in Fig. 1. For example, the job *AO* in the *navigation* task is set as having a WCET demand of 1 time unit and a relative deadline of 3 time units. The job *AO* is set as being released at 35 time units later than the release time of *FO*.

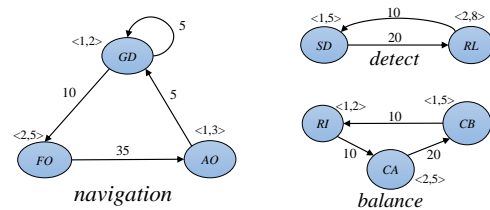


Fig. 1. DRT model of the robot controller.

## B. DRTPC Model

The DRTPC model is defined based on the DRT model to support the precedence constraint, which constrains that the execution of a job shouldn't start until all of its precedent jobs (if exist) are finished. In the DRTPC model, each task is expressed by a directed graph  $G'(T)$ , which describes the timing constraints of each job and the precedence constraints between jobs in this task. Each vertex of  $G'(T)$  is labeled by a triple  $(PJ, WCET, RD)$ , which represents a set of precedent jobs, the worst-case execution time, and the relative deadline of the corresponding job respectively. The directed edge of  $G'(T)$  has the same meaning as the DRT model, i.e. the inter-release interval time between jobs. It should be noted that the precedent job of a job may be in the same task model or in a different task model, which is called intra-task or inter-task precedence constraint respectively.

*Example 2:* The DRTPC model of the robot controller considering the precedence constraints is shown in Fig. 2.

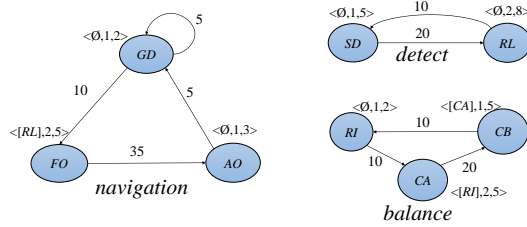


Fig. 2. DRTPC model of the robot controller.

The semantics of the DRTPC model is defined based on the execution paths generated by the task model. Such execution path is specified by a job sequence, where each job is considered from two parts: timing constraint and precedence constraint. Formally, we use a 3-tuple  $(RT, DT, AD)$  to denote a job that is released at the time  $RT$ , with the duration  $DT$  and the absolute deadline  $AD$ .

A job sequence is said to be valid, if and only if an arbitrary job in the sequence satisfies the following two conditions:

*Condition 1:* For the timing constraint, it should satisfy the three sub-conditions [12]: The duration is equal to the WCET of the job; the absolute deadline is equal to the release time plus the relative deadline of the job; the time between the release time of the job and that of an other job is greater than the inter-release interval time between them.

*Condition 2:* For the precedence constraint, it should satisfy the two sub-conditions: For the intra-task precedence constraint, all of the precedent jobs (if exist) of the job should be included in the same job sequence; For the inter-task precedence constraint, all of the precedent jobs (if exist) of the job should be included in the job sequences of other tasks, which contain these precedent jobs.

*Example 3:* We take the task model in Fig. 2 as an example to illustrate the valid job sequences within one period. For the *balance* task, the job sequence of  $(RI, CA, CB)$  with the timing constraints of  $(0, 1, 2)$ ,  $(10, 2, 15)$ , and  $(30, 1, 35)$  respectively is valid, because the precedent job of  $CA$  (i.e.  $RI$ ) is included

in this job sequence. For the *detect* task, a valid job sequence is  $(SD, RL)$  with the timing constraints of  $(0, 1, 5)$  and  $(20, 2, 28)$  respectively. For the *navigation* task, the job sequence of  $(FO, AO)$  with the timing constraints of  $(0, 2, 5)$  and  $(35, 1, 38)$  respectively is valid, because the precedent job of  $FO$  (i.e.  $RL$ ) is included in the job sequence of the *detect* task.

In the next section, we will take an example of the three valid job sequences to introduce the schedulability analysis for the DRTPC model.

## IV. SCHEDULABILITY ANALYSIS

In this section, we present the schedulability analysis for the static priority scheduling in the DRTPC model. For such scheduling, each task in the real-time system is assigned a unique priority. The jobs have the same priority as their task. For each job in the execution paths of the task, the job can be executed only if no job with a higher priority exists in the system. When all jobs in the execution paths meet their deadline after the scheduling, this task is considered as schedulable.

As the DRTPC model (DRTPC in short) is based on the DRT model (DRT in short), we first introduce the schedulability analysis for the static priority scheduling in the DRT, which only concerns the interference on the scheduling caused by the priority-based preemption. Then, we propose our schedulability analysis algorithm for the DRTPC, which extra concerns the interference caused by the precedence constraint.

### A. Schedulability Analysis for DRT

The schedulability analysis for the DRT is based on evaluating the request function [16], which represents the maximum accumulated workload of all jobs that the job sequence may generate during a time interval. The request function is defined as follows.

*Definition 1* (Request Function [16]). For the job sequence  $\sigma = (v_0, v_1, \dots, v_n)$  of an arbitrary execution path  $\pi$  in the task model, its request function before time  $t$  is defined as

$$rf_{\pi}(t) = \max(dt(\pi') | \pi' \text{ is prefix of } \pi \text{ and } g(\pi') < t) \quad (1)$$

where  $dt(\pi) = \sum_{i=0}^n dt(v_i)$ ,  $g(\pi) = \sum_{i=1}^n g(v_{i-1}, v_i)$ ,  $dt(v_i)$  is the duration of job  $v_i$ , and  $g(v_{i-1}, v_i)$  is the inter-release interval time between job  $v_{i-1}$  and job  $v_i$ .

The schedulability of a job with respect to a set of interfering tasks is specified based on the request function. To analyze the schedulability, we first define  $\Pi_T$  as the set of all execution paths for an arbitrary task  $T$  in the task model. Then, for a set of tasks  $\Gamma = (T_1, T_2, \dots, T_n)$ , let  $\Pi(\Gamma) = \Pi_{T_1} \times \Pi_{T_2} \times \dots \times \Pi_{T_n}$  be the set of all path combinations, namely  $\Pi(\Gamma) = \{ (\pi_1, \dots, \pi_n) | \pi_1 \in \Pi_{T_1}, \dots, \pi_n \in \Pi_{T_n} \}$ . Finally, the schedulability is judged based on the following theorem.

*Theorem 1* ([16]): A job with duration  $dt$  and absolute deadline  $ad$  is schedulable under a set of interfering tasks  $\Gamma$  if and only if

$$\forall (\pi_1, \dots, \pi_n) \in \Pi(\Gamma) : \exists t < ad : dt + \sum_{T_i \in \Gamma} rf_{\pi_i}(t) \leq t \quad (2)$$

This theorem shows that, a necessary and sufficient condition for the schedulability of a job is that during its release time and (absolute) deadline, there exists a time instant  $t$  at which this job and all of the interfering jobs released before  $t$  are finished. For the static priority scheduling in the DRT, the interfering jobs are the jobs with a higher priority.

### B. Schedulability Analysis for DRTPC

As the DRTPC extends the DRT to support the precedence constraint, the interference caused by such precedence constraint needs to be extra considered in the schedulability analysis for the DRTPC. We first take an example of the execution of the robot controller to present the interferences in the DRTPC.

*Example 4:* We give a scenario that the tasks in the robot controller have a priority relationship as  $P(balance) > P(detect) > P(navigation)$ . The sample execution of the tasks with the execution paths in *Example 3* is shown in Fig.3.

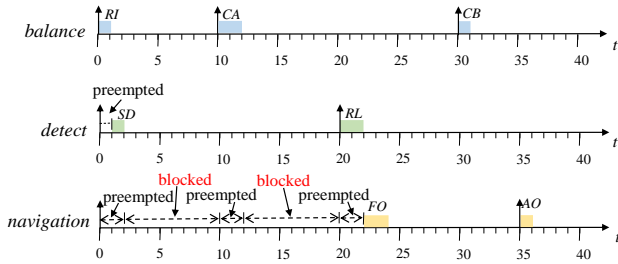


Fig. 3. A sample execution of the execution paths in *Example 3*

It can be found from the example that there are two interferences during the scheduling in the DRTPC: preemption interference caused by the jobs with a higher priority and block interference caused by the precedent jobs. The schedulability analysis for the DRT only considers the interference workload caused by the preemption. Here, we additionally study the workload resulting from the precedence constraint.

During the scheduling, if the precedent jobs of a job haven't been all finished, then this job will be blocked. The block relationship between a job  $j$  and its precedent job  $k$  is summarized as the following three cases:

- The finish time of job  $k$  is before the release time of job  $j$ . In this case, job  $j$  will not be blocked by such a precedent job.
- The finish time of job  $k$  is after the absolute deadline of job  $j$ . In this case, job  $j$  will be blocked until its deadline, which makes the block time infinite.
- The finish time of job  $k$  is between the release time and the absolute deadline of job  $j$ . This case includes two sub-cases:
  - If the release time of job  $k$  is before that of job  $j$ , then the block time is the time interval between the release time of job  $j$  and the finish time of job  $k$ .
  - If the release time of job  $k$  is after that of job  $j$ , then the block time is the time interval between the release time and finish time of job  $k$ .

Then, the workload of an arbitrary job under a set of interfering tasks in the DRTPC consists of three parts: duration of the job, preemption interference, and block interference. Based on *Theorem 1*, an arbitrary job with duration  $dt$  and absolute deadline  $ad$  is schedulable under a set of interfering tasks  $\Gamma$  if and only if

$$\forall(\pi_1, \dots, \pi_n) \in \Pi(\Gamma) : \exists t < ad : dt + \sum_{T_i \in \Gamma} rf\_P_{\pi_i}(t) + \sum_{T_i \in \Gamma} rf\_B_{\pi_i}(t) \leq t \quad (3)$$

Where  $\sum_{T_i \in \Gamma} rf\_P_{\pi_i}(t)$  and  $\sum_{T_i \in \Gamma} rf\_B_{\pi_i}(t)$  represent the accumulated workload caused by the preemption interference and the block interference respectively. This formula shows that, a necessary and sufficient condition for the schedulability of a job in the DRTPC is that during its release time and (absolute) deadline, there exists a time instant  $t$  at which this job and all of its precedent jobs together with the jobs with a higher priority released before  $t$  are finished.

Finally, we introduce the schedulability analysis procedure for a task  $T$  with an arbitrary execution path  $JS$  (i.e. a job sequence), which is shown in Fig.4.

```

function schedulability_analysis ( $T, JS$ )
1: for each job  $j \in JS$  of  $T$  do
2:   compute  $\sum_{T_i \in \Gamma} rf\_P_{\pi_i}(t)$  for  $j$ 
3:   compute  $\sum_{T_i \in \Gamma} rf\_B_{\pi_i}(t)$  for  $j$ 
4:   if  $\forall(\pi_1, \dots, \pi_n) \in \Pi(\Gamma)$ :
5:      $\exists t < ad: dt + \sum_{T_i \in \Gamma} rf\_P_{\pi_i}(t) + \sum_{T_i \in \Gamma} rf\_B_{\pi_i}(t) \leq t$  then
6:       isSchedulable( $j$ ) = true
7:     else
8:       isSchedulable( $j$ ) = false
9:     end if
10: end for
11: for each job  $k \in JS$  of  $T$  do
12:   if isSchedulable( $k$ ) == false then
13:     return false
14:   end if
15: end for
16: return true

```

Fig. 4. Procedure of schedulability analysis.

*Procedure 1:* First, compute the accumulated workload caused by the preemption interference  $\sum_{T_i \in \Gamma} rf\_P_{\pi_i}(t)$  and the block interference  $\sum_{T_i \in \Gamma} rf\_B_{\pi_i}(t)$  for every job  $j$  in the job sequence  $JS$  of the task  $T$  (L. 2, 3). Then, sum the duration of the job  $j$ , the  $\sum_{T_i \in \Gamma} rf\_P_{\pi_i}(t)$ , and the  $\sum_{T_i \in \Gamma} rf\_B_{\pi_i}(t)$  to check whether the condition of the request function in (3) is satisfied (L. 4). If there exists some time  $t$  that makes the condition satisfied, then the job  $j$  is considered as schedulable (L. 5). Only all of the jobs in  $JS$  are schedulable can the execution path  $JS$  be considered as schedulable (L. 10-15).

### C. Optimization for Analysis

The schedulability analysis proposed in Sect. IV-B considers all path combinations of the interfering tasks  $\Gamma$ , which leads to a high computational complexity. Here, we optimize this



issue by checking each path  $\pi_i$  (i.e. job sequence) in the path combinations  $\Pi_\Gamma$  before the schedulability analysis, to distinguish the path combinations that must be tested.

The check follows two principles: if the path  $\pi_i$  is checked to make the task  $T$  (to be analyzed) unschedulable, then all of the path combinations needn't be analyzed anymore, and this task is considered as unschedulable directly; if the job in path  $\pi_i$ , which is a precedent job of some job in the job sequence of task  $T$ , is checked to cause no block, then remove this precedence constraint when analyzing the accumulated workload caused by the precedence constraints. The check only involves such jobs in path  $\pi_i$  that are the precedent jobs of some jobs in the job sequence of task  $T$ . Here, we assume an arbitrary job of a job sequence of task  $T$  and its precedent job in path  $\pi_i$  as the job  $j$  and the job  $k$  respectively. Then, the condition of removing the precedence constraint is that the priority of  $k$  is higher than that of  $j$ , and the release time of  $k$  is before that of  $j$ ; the condition of unschedulable is that the priority of  $k$  is lower than that of  $j$ , and the release time of  $k$  is after that of  $j$ .

## V. EVALUATION

It has been proven that the DRT method can have the pseudo-polynomial complexity and good expressiveness comparing with existing methods. Here, we only need to evaluate that whether our based-DRT approach also has a tractable complexity. We first analyze the efficiency and scalability to evaluate whether it is practical to be used for large sets of tasks with precedence constraint. Besides, we also investigate the relationship between the schedulability and the number of precedence constraints in the task set, which is helpful to design the precedence constraints in real-time systems.

### A. Experimental Setup

We implement our approach using the Javascript programming language running on a standard desktop computer with the 3.3GHz CPU and 8 GB RAM. For the random task set generation, we consider three types of tasks (namely small, medium, and large tasks) referring to [16], which have the parameter ranges as shown in Table I. For each task, one of the three types is randomly selected, then the task parameters are chosen randomly from the corresponding intervals.

TABLE I  
TASK PARAMETER RANGES

Task Type	Small	Medium	Large
Vertices	[3,5]	[5,9]	[7,13]
Branching degree	[1,3]	[1,4]	[1,5]
$p$	[50,100]	[100,200]	[200,400]
$e$	[1,2]	[1,4]	[1,8]
$d$	[25,100]	[50,200]	[100,400]

The workload generated by the tasks (instead of their numbers) has a direct relationship to the experimental results. Here we use the utilization to represent such workload, which is defined as the ratio of the sum of duration over the sum of

inter-release separation time in the tasks [12]. Our experiment is implemented under a given task set utilization. In order to generate a task set with a desired utilization, random tasks are generated and added to the task set until the desired utilization is achieved. Besides, we use the ratio of precedence (RP) to represent the ratio of the jobs with the precedence constraint over the whole jobs in the task set, and use the acceptance ratio to represent the ratio of the schedulable tasks over the whole tasks in this experiment.

### B. Experimental Results

#### (1) Efficiency and scalability

First, we explore how much our optimization can help in the complexity reduction. For four representative ratios of precedence in the task set, the comparison of the total path combinations and the combinations that must be tested after the optimization is depicted in Fig. 5. As seen, the reduction obtained by the optimization is considerable compared to the number of total combinations. It is also observed that the task set with a higher ratio of precedence (RP) has the fewer total combinations, as well as the tested combinations. This is because a larger number of precedence constraints in the task set makes fewer paths (i.e. job sequences) valid (see *valid job sequence* in Sect. III-B).

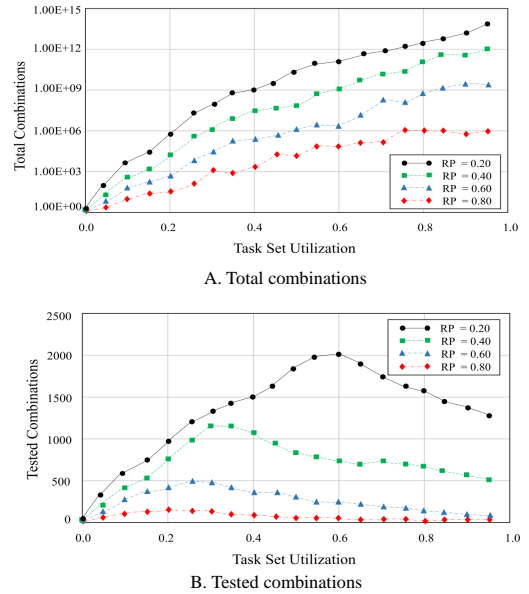


Fig. 5. Reduction of the path combinations need to be tested

Then, we evaluate the efficiency and scalability of our approach through varying the ratio of precedence in the task set with a step of 0.05. Figure 6 shows the comparison of average run-time for the two analyses (with and without the optimization) under the task set utilization of 0.5. As seen, increasing the ratio of precedence in the task set causes that the analysis without the optimization becomes very lengthy. This is because that although the higher ratio of precedence means the fewer valid path combinations, the path combinations

with more precedence constraints can sharply increase the computation. In contrast, the analysis with the optimization has a good efficiency, and scales very well with the increasing ratio of precedence. This is because that with the ratio of precedence increasing, more and more tasks can directly be checked as unschedulable in advance, and do not cost the analysis run-time anymore.

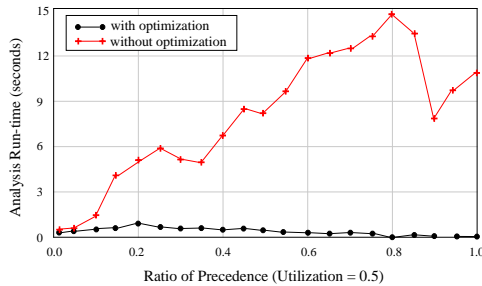


Fig. 6. Average run-time of the proposed methods

## (2) Schedulability

Here, we investigate the relationship between the schedulability and the ratio of precedence in the task set. We find that a high ratio of precedence makes most of the tasks unschedulable, so we only present the results of four low ratios of precedence as shown in Fig. 7, i.e.  $RP = 0.00, 0.05, 0.10$  and  $0.20$ . As seen, with the utilization increasing, the task set under the four ratios of precedence all exhibit a lower and lower acceptance ratio. This is because that the higher utilization makes the tasks more difficult to be arranged. It is also observed that a higher ratio of precedence causes the decline curve of acceptance ratio steeper. This is because more precedence constraints in the task set lead to more blocks, which makes more tasks unschedulable.

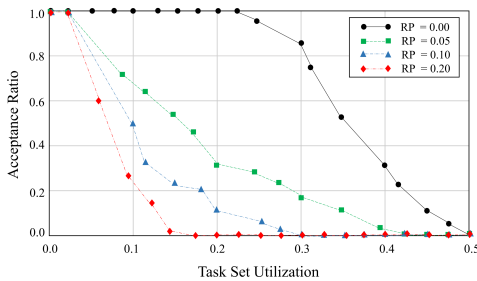


Fig. 7. Schedulability under different ratios of precedence

## VI. CONCLUSION AND PERSPECTIVES

With real-time systems are becoming more and more complex, the precedence constraints between or within their tasks directly impact the schedulability. In this paper, we propose an extension of the DRT task model to specify the precedence constraints, then propose a uniprocessor schedulability analysis algorithm for the static priority scheduling in our model. In addition, we introduce an optimization method for the analysis to improve its efficiency. Our experiments show

that, the proposed approach can scale well for large sets of tasks with precedence constraint. Except for the precedence constraint, there are various constraints between the tasks affecting their schedulability, next we will extend our approach to more constraints. Besides, we will also extend our approach to support the multiprocessor in the future.

## REFERENCES

- [1] L. Sha, T. Abdelzaher, K.-E. Årzén, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, and A. K. Mok, "Real-time scheduling theory: A historical perspective," *Real-time systems*, vol. 28, no. 2-3, pp. 101–155, 2004.
- [2] M. Stigge and W. Yi, "Graph-based models for real-time workload: a survey," *Real-time systems*, vol. 51, no. 5, pp. 602–636, 2015.
- [3] J. Schlatow and R. Ernst, "Response-time analysis for task chains with complex precedence and blocking relations," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 5s, p. 172, 2017.
- [4] D. Prot and O. Bellenguez-Morineau, "How the structure of precedence constraints may change the complexity class of scheduling problems," *arXiv preprint arXiv:1510.04833*, 2015.
- [5] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM (JACM)*, vol. 20, no. 1, pp. 46–61, 1973.
- [6] N. R. Adam, V. Atluri, and W.-K. Huang, "Modeling and analysis of workflows using petri nets," *Journal of Intelligent Information Systems*, vol. 10, no. 2, pp. 131–158, 1998.
- [7] E. Fersman and W. Yi, "A generic approach to schedulability analysis of real-time tasks," *Nordic Journal of Computing*, vol. 11, no. 2, pp. 129–147, 2004.
- [8] S. Baruah, V. Bonifaci, A. Marchettispaccamela, L. Stougie, and A. Wiese, "A generalized parallel task model for recurrent real-time processes," in *IEEE Real-Time Systems Symposium*, 2012, pp. 63–72.
- [9] T. P. Baker and S. K. Baruah, "An analysis of global edf schedulability for arbitrary-deadline sporadic task systems," *Real-Time Systems*, vol. 43, no. 1, pp. 3–24, 2009.
- [10] M. G. Harbour, "Exploiting precedence relations in the schedulability analysis of distributed real-time systems," in *Real-Time Systems Symposium, 1999. Proceedings. the IEEE*, 1999, pp. 328–339.
- [11] R. Pellizzoni and G. Lipari, "Improved schedulability analysis of real-time transactions with earliest deadline scheduling," in *Real Time and Embedded Technology and Applications Symposium, 2005. RTAS, 2005*, pp. 66–75.
- [12] M. Stigge, P. Ekberg, N. Guan, and W. Yi, "The digraph real-time task model," in *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2011 17th IEEE*. IEEE, 2011, pp. 71–80.
- [13] A. K. Mok and D. Chen, "A multiframe model for real-time tasks," in *Proc. Real-Time Systems Symposium, Dec*, 1996, pp. 22–29.
- [14] S. Baruah, D. Chen, S. Gorinsky, and A. Mok, "Generalized multiframe tasks," *Real-Time Systems*, vol. 17, no. 1, pp. 5–22, 1999.
- [15] S. K. Baruah, "Dynamic- and static-priority scheduling of recurring real-time tasks," *Real-Time Systems*, vol. 24, no. 1, pp. 93–128, 2003.
- [16] M. Stigge and W. Yi, "Combinatorial abstraction refinement for feasibility analysis," *Real-Time Systems*, vol. 51, no. 6, pp. 1–36, 2013.
- [17] M. Stigge, P. Ekberg, N. Guan, and W. Yi, "On the tractability of digraph-based task models," in *Real-Time Systems (ECRTS), 2011 23rd Euromicro Conference on*. IEEE, 2011, pp. 162–171.
- [18] N. Guan, P. Ekberg, M. Stigge, and W. Yi, "Resource sharing protocols for real-time task graph systems," in *Real-Time Systems (ECRTS), 2011 23rd Euromicro Conference on*. IEEE, 2011, pp. 272–281.
- [19] Y. Zhuo, "Static priority schedulability analysis of graph-based real-time task models with resource sharing," 2014.
- [20] M. Mohaqeqi, J. Abdullah, N. Guan, and W. Yi, "Schedulability analysis of synchronous digraph real-time tasks," in *Real-Time Systems (ECRTS), 2016 28th Euromicro Conference on*. IEEE, 2016, pp. 176–186.
- [21] N. Guan, Y. Tang, J. Abdullah, M. Stigge, and W. Yi, "Scalable timing analysis with refinement," in *TACAS*, 2015, pp. 3–18.
- [22] P. Jayachandran and T. Abdelzaher, "Transforming distributed acyclic systems into equivalent uniprocessors under preemptive and non-preemptive scheduling," in *Euromicro Conference on Real-Time Systems*, 2007, pp. 233–242.
- [23] C. M. Krishna, *Real-Time Systems*. Wiley Online Library, 1999.

# *Conflict Management in the Collaborative Description of a Domain Language*

Claudia Litvak  
DIIT  
Universidad Nacional de La Matanza  
La Matanza, Bs. As., Argentina  
clitvak@unlam.edu.ar

Gustavo Rossi<sup>1</sup>, Leandro Antonelli  
Lifia, Facultad de Informatica  
Universidad Nacional de La Plata  
La Plata, Argentina  
{gustavo, lanto}@lifia.info.unlp.edu.ar  
<sup>1</sup>also CONICET

**Abstract**—The identification and specification of the requirements of a software system is a difficult task that has the goal of obtaining requirements as correct and complete as possible. It is extremely important that Requirements Engineers understand a domain language in order to write high-quality requirements. Moreover, they must describe (and discuss) the language in a collaborative way in order to consider the different points of view of all stakeholders to assure that the resulting requirements will have more chances to meet their needs. However, collaborative construction implies the occurrence of conflicts that are unavoidable because of ambiguity, overlapping and misunderstanding natural language descriptions. This article relies on the Language Extended Lexicon in order to describe the application domain. Although it is a semi-structured glossary and this characteristic helps to reduce the conflicts, our experience shows that conflicts arise anyway. Thus, in order to mitigate this problem, this article presents a catalogue with a set of conflicts that could appear during a collaborative construction of the Language Extended Lexicon and proposes alternatives for their resolution.

**Keywords**—requirements engineering, collaboration, conflicts, natural language models

## I. INTRODUCTION

Requirements Engineering is one of the initial stages of the Software Development Life Cycle. The goal of this stage is to acquire the knowledge and the requirements needed for the system to be built. Errors made in requirements specifications have a great impact towards the end of software development, since the cost of error correction increases as each stage progresses [1].

Several authors argue that the interaction of different stakeholders working collaboratively on the same problem improves the quality of the system requirements [2] [3]. Since different stakeholders have different concerns and different point of view, all of them working together will produce a richer model.

However, generating models collaboratively implies the emergence of conflicts that must be solved in order to build a consistent high quality model. The existence of a conflict is not a negative situation, in fact it might be positive since it

provides the possibility of improving the models, analyzing and discussing the different ideas observed and manifested by the conflict.

In this context, it is even more important, to define a basic language in order to interact and describe the needed models. There are two main kinds of languages: formal and natural language. Despite the introduction of ambiguity, the natural language has the advantage to be understood by all the stakeholders (technical and non technical).

Ambiguity means having two interpretations for the same word. For example, let's consider that the word "label" has two different meanings: (i) "It is the action of putting the brand of the product on the boxes of finished product"; and (ii) "It is the action of marking the price of each finished box of finished product". Imagine a situation where two stakeholders use the same word with different meaning: they would think they understand each other, but in fact, they want to transmit a different idea. An opposite situation could be the use of two different words, which in fact are synonyms and represent the same idea. In this case, both stakeholders can not know that they are talking about the same thing.

Our research is framed by the Language Extended Lexicon (LEL). The LEL is a model that uses Natural Language [4] to describe the vocabulary of the application domain. The LEL is a very convenient tool for stakeholders with no technical skills, although people with such skills will profit more from its use [5]. In particular, the convenience of the LEL as a tool arises from three significant characteristics: it is easy to learn, it is easy to use and it has good expressiveness. Goel [6] states that the LEL is widely used to capture the language to describe requirements. Moreover, it is a useful technique because can be understood by the stakeholders, and this characteristic encourage their active participation which is crucial in first steps of software development.

The LEL captures the terms (they are called symbols) and describes them with the name, the notion, and the behavioral responses. The name identifies the symbol; all synonyms that exist in the domain must be defined in this attribute. The notion describes the meaning (denotation) and the behavioral responses describe the relation of the symbol with other

symbols (connotation). Every LEL symbol belongs to one of four categories: Subject, Object, Verb, and State.

Antonelli [7] outlines a strategy to describe the LEL in a collaborative way. However, it is very difficult to produce a domain language specification when there are many actors involved [8]. In a collaborative context, all participants build a joint model, and as previously explained conflicts might emerge between the different viewpoints.

This paper presents an approach for the identification and resolution of conflicts that emerge when the LEL is developed collaboratively. The collaborative construction of the LEL means that different stakeholders propose symbols and provides definitions in an iterative way. This means that different people collaborate by making specific contributions: identifying the symbol that must be defined, or adding a definition. Nevertheless, in this context, it is necessary to have a full understanding of all the definitions. Our proposed approach consists in analyzing the whole glossary looking for conflicts and providing a solution for each conflict.

The paper is organized as follows: Section II provides the related work; Section III presents the conflicts, the proposed solutions, and a preliminary evaluation; finally, Section IV sets out the conclusions and future work.

## II. RELATED WORK

Different authors have studied the existence of conflicts in Requirements Engineering [9]. Literature covers a wide range of conflict types and stages of the requirement phase where conflicts can appear [10]. Bendjenna [11] states the importance of dealing with conflictive situations during Requirements Engineering, considering the variety of stakeholders with the common objective of obtaining a unique system. Aldekhail [12] presents a literature review related to requirements conflicts. Some publications have presented requirements conflict management in a web-based collaborative environment. The SOP project [13] has developed a wiki using the Volere Requirements Specification Template [14], seeking to pinpoint inconsistencies in requirements documents created with their tool. WikiWinWin [15] is a wiki front-end to the WinWin tool. Urbieta [16] presents an approach for detecting and solving inconsistencies and conflicts in web software requirements and shows a taxonomy for conflicts in Web applications requirements. Lutz [2] developed CREW-Space, a tool to support the co-located collaboration of several users to simultaneously interact through Android-enabled mobile devices. They use role playing to involve different stakeholders in a use case analysis. Azadegan [3] proposes two steps: (i) identifying relevant user requirements and (ii) voting for user requirements.

The problem of conflicts also appears when building domain ontologies collaboratively. Lexons with properties, restrictions and relationships are defined in ontologies. In the LEL, there are symbols with two specific attributes (notion and behavioral responses), and relationships between the symbols are hyperlinks to other symbols used to make the description. Also each symbol has a type. The most important difference between ontologies and our approach is that we analyze these definitions, while approaches with ontologies mainly analyze

the relationship between the elements. It was analyzed if there is overlapping in definition of the notion or the behavioral responses, or even if they are similar. If definitions are similar it could imply that synonyms were found. It is important to pay attention to homonyms, which are the same symbol referring to different things. Symbols (concepts) are naturally organized in a hierarchy way. This approach also analyzes how definitions are organized or repeated in such structure. In collaborative ontology engineering there is a great variety of methodologies [17], nevertheless, they do not analyze the definitions. Chen [18] proposes an approach that deals with classes and relations. They detect three kinds of conflicts: hard, soft and latent conflicts between the classes. On the subject of building ontologies collaboratively some studies apply the consensus method [19] [20]. It has been proved to be useful in conflict solution between objects. The most important problem in consensus-based collaboration, is defining when they get an agreement. Consensus quality concept [21] is defined to show, how they get a consensus, in the construction of the Vietnamese language dictionary with WordNet.

## III. CONFLICTS IN THE COLLABORATIVELY DEVELOPED LEL MODEL

This section presents the proposed conflict resolution approach and a preliminary validation. Section A describes the process to identify conflicts during the collaborative description of the LEL and presents a set of the conflicts that could arise. It is important to mention that these conflicts were identified from several real-life software systems descriptions. Section B shows each conflict and the actions to solve them. Finally, section C presents a preliminary evaluation.

### A. Our Approach in a Nutshell

The LEL is built in an iterative and incremental way, where different Requirements Engineers contribute to its description. With different points of view a conflict may arise. Thus, it must be identified and solved as soon as possible in order to obtain a consistent LEL (see Figure 1).

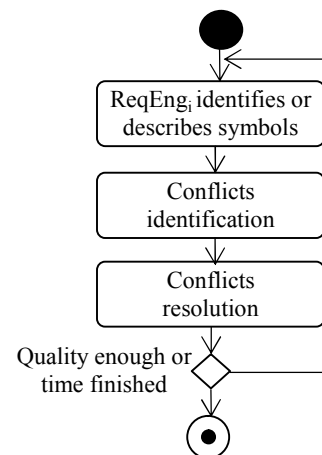


Figure 1. Process for conflicts resolution (Req Eng<sub>i</sub>: Requirements Engineer<sub>i</sub>)

The first step represents the action that every Requirements Engineer performs: identifying a symbol or contributing with the description of notion or behavioral response. Every action

can give origin to a conflict. For example, two different Requirements Engineers could define independently the previous Label symbols of Table 1 and 2. Thus, the whole glossary must be analyzed in order to identify conflicts. If a conflict is identified, it should be solved in order to assure the consistency of the LEL.

The list of conflicts was defined from the analysis of several real projects. The conflicts are grouped in categories in order to make the description clearer.

The first category is *Semantic conflicts*: These are conflicts that arise when there are differences in the meaning of the symbols. For example, Label refers to two different actions: the action of putting the brand (Table I) and the action of marking the price (Table II). Subcategories of Semantic conflicts are: (i) the same identification for elements with different meaning and the same syntactic classification; (ii) different identification for elements that refers to the same concept in the same way; (iii) different identification for elements that refers to the same concept in different way; (iv) different identification for elements that refers to the same concept with complementary information.

The second category is *Structural conflicts*: Structural conflicts arise when there is complete or partial repetition in the definitions, considering the description of the behavioral responses or the organization of the description in hierarchies. For example, let's consider that one symbol is a generic concept, and there is a specific term that specializes the previous one, and the last symbol repeats information described in the first one. Subcategories are: (i) different level of detail; (ii) descriptions duplicated in hierarchies.

The last category is *Syntactic conflicts*: These conflicts appear when the same symbol has different syntactic classifications. For example, Label can be an Object or a Verb. There is no subcategory.

*B. Catalogue of Conflicts and their Solutions*

This section describes the conflicts with more detail together with their proposed resolution. In order to illustrate the proposed approach, we chose "IP Etiquetas S.A.", a company that produces some kinds of sticky labels, either with barcodes, with specific brands or white ones. Underlined words are other LEL symbols.

The study was developed by means of a series of interviews carried out by different Requirements Engineers with several people in the company. A series of conflicts arose during the attempt to define the LEL model collaboratively. The total number of conflicts found was 17. For space reasons we show some of them in detail. The other conflicts refer to behavioral response conflicts and also conflicts generated when part of a description of notion or behavioral response defined by a requirement engineer is contained on the defined by other requirement engineer. Some examples include the Label symbol, which was considered by a Requirements Engineer as the verb meaning "attach a label," whereas another Requirements Engineer considers that Label is the produced label. A third engineer thinks the Label symbol means "attach the price tag," this being also a verb.

*1) The same identification for elements with different meaning and the same syntactic classification (Homonym)*

This conflict arises when there are two different entries that are identified with the same symbol, but they represent different things. For example, let's consider two different definitions of the symbol "Label" as described in Table I and Table II. The identification of both symbols is the same, since it is "Label". Nevertheless, both LEL entries refer to different things; one represents the action of putting the brand, while the other represents the action of marking the price.

TABLE I. LABEL SYMBOL

Symbol #: 10	Author: Req. Eng. 3	Type!: Verb
Name/s	Label	
Notion	- It is the action of putting the brand of the <u>product</u> on the boxes of <u>finished product</u> .	
Behavioral Response	-The <u>logo of the brand</u> is defined with the client and is previously established.	

TABLE II. LABEL SYMBOL

Symbol #: 10	Author: Req. Eng. 1	Type!: Verb
Name/s	Label	
Notion	- It is the action of marking the price of each finished box of <u>finished product</u> .	
Behavioral Response	-The <u>price per box</u> is previously established according to the total number required.	

Heuristic to detect the conflict: review all the LEL entries, identifying two or more entries with the same identification. Check the notion, in order to determine whether the entry is duplicated or they are different entries.

Solution: If the entry is duplicated merge both definitions. If the entries are different, specialize the identification in order to make clear that there are different entries: Label(1) and Label(2).

*2) The same identification for elements with different syntactic classification (Homonym)*

This conflict is similar to the previous one, but the difference relies on the type of the entries. For example, let's consider a new symbol "Label" with Verb classification (Table III), while the other "Label" symbols refers to Objects (Table I). The "Label" of object category refers to the end product manufactured by the company.

TABLE III. LABEL SYMBOL

Symbol #: 11	Author: Req. Eng. 2	Type!: Object
Name/s	Label	
Notion	- <u>Product</u> manufactured by the company	
Behavioral Response	-...	

Heuristic to detect the conflict: review all the LEL entries, identifying two or more entries with the same identification and different category.

Solution: Rename the symbols as Label(1) and Label(3).

3) *Different identification for elements that refer to the same concept in the same way (Synonym).*

This conflict arises when there are two different entries that are identified with different symbols, but they are described in the same way. For example, let’s consider two different entries “missing stock” and “insufficient raw material” as described in Table IV and Table V. Both refer to the same situation described identically. That is, “State of raw material stock when it is lower than the minimum stock level.”

TABLE IV. MISSING STOCK SYMBOL

Symbol #: 17	Author: Req. Eng. 2	Type <sup>1</sup> : State
Name/s	Missing stock	
Notion	-State of <u>raw material</u> stock when it is lower than the <u>minimum stock level</u> .	
Behavioral Response	-...	

TABLE V. INSUFFICIENT RAW MATERIAL SYMBOL

Symbol #: 9	Author: Req. Eng. 3	Type <sup>1</sup> : State
Name/s	Insufficient raw material	
Notion	-State of <u>raw material</u> stock when it is lower than the <u>minimum stock level</u> .	
Behavioral Response	-...	

Heuristic to detect the conflict: Compare all the notions of the different symbols checking for coincidences.

Solution: Define the elements as synonyms. In the example, “Missing Stock / Insufficient Raw Material element” must be defined as synonyms of the same entry.

4) *Different identification for elements that refer to the same concept in different way (Overlapping).*

This conflict arises when there are two different entries that are identified with different symbols, but they are described in different way. For example, let’s consider two different entries “insufficient raw material” as described in Table V and Table VI. Both refer to the same situation described similarly. One symbol is described as “State of raw material stock when it is lower than the minimum stock level.” while the other is described as “State of the stock of supplies when it must be changed to replenishment.” Both symbols refer to the same concept, and both descriptions are similar.

TABLE VI. INSUFFICIENT RAW MATERIAL SYMBOL

Symbol #: 8	Author: Req. Eng. 1	Type <sup>1</sup> : State
Name/s	Insufficient raw material	
Notion	-State of the stock of <u>supplies</u> when it must be changed to <u>replenishment</u> .	
Behavioral Response	-...	

Heuristic to detect the conflict: Compare all the notions of the different symbols checking for similarities.

Solution: Since both descriptions are similar, it must be agreed only one description. The other entry must be removed. In Tables V and VI, the same symbol with a different Notion is shown.

5) *Different level of detail.*

This conflict arises when there are different symbols overlapping concepts in a hierarchy structure not well defined. Let’s consider the situation of two different operators: (i) Rewinder Operator and (ii) Flexographic Printing Press Operator. One Requirements Engineer defines only one symbol named “Operator” with a general description considering both roles (i) and (ii). While other Requirements Engineer defines the two specific symbols (i) and (ii). In this situation, there are common characteristics to both roles; it should be described in a generic “operator” symbol, and then, the specific characteristics of both roles (i) and (ii) should be described in them.

TABLE VII. OPERATOR SYMBOL

Symbol #: 22	Author: Req. Eng. 3	Type <sup>1</sup> : Subject
Name/s	Operator	
Notion	-It is the technician in charge of operating the <u>production machines</u> .	
Behavioral Response	-...	

TABLE VIII. FLEXOGRAPHIC PRINTING PRESS OPERATOR SYMBOL

Symbol #: 9	Author: Req. Eng. 2	Type <sup>1</sup> : Subject
Name/s	Flexographic printing press operator	
Notion	-Is the technician in charge of operating the <u>flexographic printing press</u> .	
Behavioral Response	-...	

TABLE IX. REWINDER OPERATOR SYMBOL

Symbol #: 20	Author: Req. Eng. 2	Type <sup>1</sup> : Subject
Name/s	Rewinder operator	
Notion	-It is the person in charge of rewinding the <u>label</u> rolls. -It is the technician in charge of operating the <u>rewinding machine</u> .	
Behavioral Response	-...	

Heuristic to detect the conflict: Compare all the notions of the different symbols looking for possible hierarchy structures.

Solution: Identify the generic and specific terms of the hierarchy structure, and describe the specifics mentioning the generic. For example, in specializes symbols, refer to “Operator”, saying that “He is an Operator that ...”

6) *Different identification for elements that refer to the same concept with complementary information (Synonym with complementary information).*

This conflict arises when there are two different entries that are identified with different symbols, and they are described

with complementary information. For example, let's consider two different entries "Cash Flow" and "Monetary Flow" as described in Table X and Table XI. Both refer to the same situation. In this case "Cash Flow" describes more details in Notion, defining it as "the amount of cash inflows and outflows" and that "it is originated by payments issued or received" while "Monetary Flow" is defined by "the amount of cash inflows and outflows". Moreover, this situation could be observed in Behavioral Response.

TABLE X. CASH FLOW SYMBOL

Symbol #: 3	Author: Req. Eng. 5	Type <sup>1</sup> : Object
Name/s	Cash Flow	
Notion	-It is the amount of cash inflows and outflows. -It is originated by payments issued or received.	
Behavioral Response	-It is daily prepared by the Treasurer.	

TABLE XI. MONETARY FLOW SYMBOL

Symbol #: 13	Author: Req. Eng. 1	Type <sup>1</sup> : Object
Name/s	Monetary flow	
Notion	-It is the amount of cash inflows and outflows.	
Behavioral Response	-It is approved and registered by Treasurer. -It is used as a source of information when preparing the Sales Forecast.	

Heuristic to detect the conflict: Compare all the notions and Behavioral Response looking for common descriptions in different symbols checking for coincidences and differences.

Solution: Define the elements as synonyms; merging all the descriptions, that is, the whole description must be used: the common part, and the particularities of each symbol. In the example, "Cash Flow / Monetary flow" must be defined as synonyms of the same entry with the richer description in each case.

7) *Descriptions duplicated in in hierarchies*

This conflict arises when descriptions are duplicated in specific elements of the hierarchy instead of putting them in the generic element. For example, two specific elements have the same description in the behavioral responses. Thus, the objective of the hierarchy is to put the common descriptions in the generic element. The same problem could arise in the notion.

TABLE XII. OPERATOR SYMBOL

Symbol #: 22	Author: Req. Eng. 1	Type <sup>1</sup> : Subject
Name/s	Operator	
Notion	-It is the technician in charge of operating the <u>production machines</u> .	
Behavioral Response	- Send the finished order to the Plant Manager	

Let's consider the situation of two different operators: (i) Rewinder Operator and (ii) Flexographic Printing Press Operator. A requirements engineer has placed the same behavioral response on each specialized symbol and another

requirements engineer has defined a generic symbol, but the former did not realize that the generic symbol was the right place to put the description. The corresponding behavioral responses "Send the finished order to the Plant Manager" must be eliminated from each specialized, leaving this description only in the generic.

TABLE XIII. FLEXOGRAPHIC PRINTING PRESS OPERATOR SYMBOL

Symbol #: 9	Author: Req. Eng. 5	Type <sup>1</sup> : Subject
Name/s	Flexographic printing press operator	
Notion	-Is the technician in charge of operating the <u>flexographic printing press</u> .	
Behavioral Response	- Send the finished order to the Plant Manager	

TABLE XIV. REWINDER OPERATOR SYMBOL

Symbol #: 21	Author: Req. Eng. 5	Type <sup>1</sup> : Subject
Name/s	Rewinder operator	
Notion	-It is the person in charge of rewinding the <u>label rolls</u> . -It is the technician in charge of operating the <u>rewinding machine</u> .	
Behavioral Response	- Send the finished order to the Plant Manager	

Heuristic to detect the conflict: Compare all the notions and Behavioral Response of the different symbols looking for repetitions in the specific elements.

Solution: Move the repeated description from the specific elements to the generic one.

C. *Preliminary Evaluation*

In order to validate the conflicts proposed in this paper, we analyzed a LEL built collaboratively by 5 Requirements Engineers. We analyze the resulting LEL looking for the conflicts we proposed. Then, we present every report to Requirements Engineers who participated in the construction of the LEL to check whether they agree with the conflicts reported. Requirements Engineers have agreed in almost all the conflict reported. The following Table XV presents some figures for the 5 different participants.

TABLE XV. TOTAL OF CONFLICTS FOUND IN IP ETIQUETAS

Req. Eng.	Total of symbols described	Symbols with conflicts	Percentage
Req. Eng. 1	42	31	74
Req. Eng. 2	35	28	80
Req. Eng. 3	28	21	75
Req. Eng. 4	31	27	87
Req. Eng. 5	47	38	80

Table XV presents for each Requirements Engineers the number of symbols in which he participated in their description, the symbol with conflict identified by our approach and the percentage that it represents. This table shows that conflicts are very common.

#### IV. CONCLUSIONS AND FURTHER WORK

Requirements definition is one of the initial stages in the software development process and their products are the groundwork for subsequent stages. Thus, errors made in requirements stage will be replicated and deepened in subsequent stages. For this reason, it is extremely important to develop requirements models of the highest quality as possible. When requirements models are developed collaboratively, conflicts unavoidable will arise. Moreover, natural language descriptions are more plausible to give origin to conflicts.

A vast experience in working with a structured glossary, the Language Extended Lexicon (LEL), proves that such structure reduces the occurrence of conflicts. However engineers have observed that while building the LEL collaboratively produces a richer model, it also introduces conflicts. In our research, and by analyzing several application domains of real projects, a classification of conflicts was devised. A process and guides for their resolution has been described in this paper. Our approach with some examples of a real project was also illustrated.

A preliminary evaluation was presented; it showed the importance of identifying conflicts and the solutions for the conflicts proposed. The percentage of conflicts was between 74% and 87%, in the five groups that have been evaluated. It shows the importance of solving those conflicts for arriving to better quality models.

An experiment to validate the conflicts and their resolutions is being designed. This experiment will be conducted in a different country to validate in another context the findings presented in this paper.

A process to identify the conflicts and an automated suggestion of solutions is planned. This implementation will be based on two important modules: (i) a module of natural language processing and (ii) a module of machine learning.

#### REFERENCES

- [1] B.W. Boehm, *Software Engineering Economics*. Prentice Hall, 1981.
- [2] R. Lutz, S. Schäfer, and S. Diehl, "Using mobile devices for collaborative requirements engineering", *27th IEEE/ACM International Conference on Automated Software Engineering*, pp. 298–301. ACM, 2012.
- [3] A. Azadegan, X. Cheng, F. Niederman, and G. Yin, "Collaborative requirements elicitation in facilitated collaboration: report from a case study", *46th Hawaii International Conference on System Sciences*, pp. 569–578, ISSN 15301605, IEEE, 2013.
- [4] J. C. S. D. P. Leite, and A. P. M. Franco, "A strategy for conceptual model acquisition", *Requirements Engineering, IEEE International Symposium on*, pp. 243–246. IEEE, 1993.

- [5] A. d. P. A. Oliveira, J. C. S. d. P. Leite, L. M. Cysneiros and C. Cappelli, "Eliciting Multi-Agent Systems Intentionality: from Language Extended Lexicon to i\* Models", *Chilean Society of Computer Science, 2007. SCCC '07. XXVI International Conference of the*, Iquique, 2007, pp. 40–49, doi: 10.1109/SCCC.2007.20
- [6] S. Goel, "Transformation from LEL to UML", *International Journal of Computer Applications*, vol. 48, no. 12, 2012.
- [7] L. Antonelli, G. Rossi, and Oliveros A., "A collaborative approach to describe the domain language through the Language Extended Lexicon", *Journal of Object Technology*, vol. 15, no. 3, pp. 1–27, 2016.
- [8] N. Mulla, S. Girase S, "A new approach to requirement elicitation based on stakeholder recommendation and collaborative filtering", *International Journal of Software Engineering and Applications*, vol. 3(3), pp. 51–60, 2012, doi:10.5121/ijsea.2012.3305.
- [9] S. Easterbrook, "Resolving requirements conflicts with computer-supported negotiation", *Requirements engineering: social and technical issues*, vol. 1, pp. 41–65, 1994.
- [10] W. N. Robinson, S. D. Pawlowski, and V. Volkov, *Requirements interaction management*. ACM Computer Survey, vol. 35(2), pp. 132–190, 2003.
- [11] H. Bendjenna, P. J. Charrel, and N. E. Zarour, "Using AHP Method to Resolve Conflicts Between Non-Functional Concerns", *International Conference on Education, Applied Sciences and Management (ICEASM'2012)*, Dubai, UAE, pp. 26–27, 2012.
- [12] M. Aldekhail, A. Chikh, and D. Ziani, "Software Requirements Conflict Identification: Review and Recommendations", *International Journal of advanced computer science and applications*, vol. 7, no. 10, pp. 326–335, 2016.
- [13] B. Decker, E. Ras, J. Rech, P. Jaubert, and M. Rieth, "Wiki-based stakeholder participation in requirements engineering". *IEEE Software*, vol. 24(2), pp. 28–35, 2007.
- [14] J. Robertson, and S. Robertson, "Volere Requirements Specification Template". *The Atlantic Systems Guild*, 2012.
- [15] D. Yang, D. Wu, S. Koolmanojwong, Brown, A. W., and B. W. Boehm, "Wikiwinwin: A wiki based system for collaborative requirements negotiation", *Hawaii International Conference on System Sciences, Proceedings of the 41st Annual*, pp. 24–24. IEEE, 2008.
- [16] M. Urbietta, M. J. Escalona, E. R. Luna, and G. Rossi, G., "Detecting conflicts and inconsistencies in web application requirements", *International Conference on Web Engineering*, pp. 278–288. Springer, Berlin, Heidelberg, 2011.
- [17] E. Simperl, and M. Luczak-Rösch, "Collaborative ontology engineering: a survey", *The Knowledge Engineering Review*, vol. 29, no. 1, pp. 101–131, 2014.
- [18] Y. Chen, X. Peng, and W. Zhao, "An approach to detect collaborative conflicts for ontology development", *Advances in Data and Web Management*, pp. 442–454. Springer, Berlin, Heidelberg, 2009.
- [19] S. Karapiperis, and D. Apostolou, D. "Consensus building in collaborative ontology engineering processes". *Journal of Universal Knowledge Management*, vol 1(3), pp. 199–216, 2006.
- [20] N. T. Nguyen, "Advanced methods for inconsistent knowledge management". Springer, London (2008)
- [21] T. H. Duong, M. Q. Tran, and T.P.T. Nguyen, "Collaborative Vietnamese WordNet building using consensus quality", *Vietnam J ComputSci 2017*, vol 4:85, Springer Berlin Heidelberg, Print ISSN: 2196-8888, Online ISSN: 2196-8896, 2017.



# Belief Function Theory in Constraint Satisfaction Problems: a Unifying Approach\*

Aouatef Rouahi  
ISG of Tunis, Tunisia  
rouahi.aouatef@hotmail.fr

Kais Ben Salah  
University of Jeddah  
kaisuria@yahoo.fr

Khaled Ghedira  
Central University, Tunisia  
khaled.ghedira@isg.rnu.tn

## Abstract

*The Constraint Satisfaction Problem (CSP) is acknowledged as a simple declarative formalism for modeling well-defined decision problems. However, real-world problems are usually ill-defined, especially, under uncertain circumstances. In such situation, uncertainty evokes the need for flexibility or softness where we accept satisfying some constraints to some degree. Moreover, when the relevance of some constraints depends on other factors, we should prioritize those constraints. Eventually, the modeled uncertainty, as well as the expressed soft and prioritized constraints induce preferences over the solutions set. Previous work employing mathematical uncertainty theories are either uncertainty-based frameworks or preference-based ones and the only attempt to handle both uncertainty and preferences is performed using two uncertainty theories under a commensurability assumption. In this paper, we propose a unifying CSP extension, labeled Belief CSP, that deals jointly with all these four concepts, i.e., uncertainty, soft and prioritized constraints and preferences over the solutions set, by exploiting the expressiveness of the belief function theory.*

## 1 Introduction

The classical Constraint Satisfaction Problem (CSP) [6, 5] framework has carried high attention because of its simplicity and generality. In fact, every problem that can be described by a set of variables and a set of constraints among those variables can easily be cast as a CSP

However, decision problems tackled by the classical CSP are assumed to be well-defined so that all their features are precise and known with certainty. Hence, the CSP has proven unfit for reasoning under uncertainty where most of the problems are ill-defined, i.e., some problem's components are either beyond our control or cannot be assimilated

due to the lack of the required quantity and quality of knowledge or even ignorance.

From another side, yes-or-no reasoning makes no sense in uncertain contexts. In fact, uncertainty evokes the need for flexibility as the satisfaction of some constraints depends on ill-known components. In addition, the majority of the problems being solved are over-constrained, i.e., they do not lead to any solution. Therefore, it is necessary to soften some constraints in order to meet a solution, that is, accepting satisfying some constraints to some degree.

To boot, in real world problems, not all constraints are fully reliable. In other words, under uncertainty, the agent cannot entirely rely on some constraints that can be untrustworthy, or subject to change. This reliability affects the relevance of each constraint to the problem. Therefore, we should rank those constraints according to their reliability by assigning priorities.

As well, solutions cannot be equally preferable. In fact, uncertainty induces preference levels over the set of solutions.

In response to these issues, a variety of frameworks has been introduced. We are particularly interested in CSP extensions that employed mathematical uncertainty theories. Those extensions may be split in two trends: Uncertainty-based frameworks that focus on handling uncertainty in CSPs as suggested by the probabilistic CSP [4], which admits that the presence or the relevance of some constraints to the real problem may be uncertain.

Besides, the preference-based frameworks that focus on relaxing constraints in order to make an over-constrained problem solvable, such as the possibilistic CSP [10], which induces preference degrees over the constraints set to evaluate the necessity of the satisfaction of each constraint and the fuzzy CSP [3] that considers a constraint as a fuzzy set by making the satisfaction of a constraint by each labeling a matter of degree.

Yet, despite this variety of frameworks, there is no proposal that addresses the whole panel of those four notions mentioned above. We shall mention the work proposed in

\*10.18293/SEKE2018-133

[2] that combined two theories, that are possibility theory and fuzzy sets theory, to express both uncertainty and soft constraints in CSPs but under a commensurability assumption between uncertainty degrees and satisfaction degrees.

In this paper, we introduce a unifying CSP framework, labeled Belief Constraint Satisfaction Problem, that deals jointly with uncertainty, flexibility, priorities and preferences in CSPs employing one uncertainty theory, that is, the Belief Function Theory (BFT) [1, 7, 8], on account of its expressiveness. The BFT proposes a natural tool for imperfection<sup>1</sup> modeling as it allows handling uncertain as well as imprecise data.

The paper is organized as follows: The next section presents a formal definition of the BCSP and shows how uncertainty is handled and which components are involved; it also illustrates the prioritized and soft constraints modeling and preferences expressing. In the section 3 we will apply the BCSP to a simple agricultural land-use planning problem followed by a discussion of the main contribution and some further work.

## 2 Belief Constraint Satisfaction Problem

### 2.1 Preliminaries

A classical CSP is defined by a quadruplet  $(X, D, C, R)$  where  $X = \{x_1, \dots, x_n\}$  is a finite set of  $n$  variables, each  $x_i$  takes its values in a finite domain  $D_i$  such that  $D = \{D_1, \dots, D_n\}$ . The simultaneous assignment of values to a set of variables is called an instantiation and denoted by  $\theta$ .  $C = \{C_1, \dots, C_m\}$  is a finite set of  $m$  constraints where each constraint  $C_i$  is defined on a subset of variables  $S_i \subseteq X$  delimited its scope and by a relation  $R_i$  that specifies the set of permitted instantiations with respect to  $C_i$ ;  $R_i$  is a subset of the Cartesian product of the domains of the variables in  $S_i$  (i.e.,  $R_i \subseteq \times\{D_i|x_i \in S_i\}$ ). A solution of a CSP is a consistent complete instantiation, i.e., an instantiation of all the variables in  $X$ , so that, all the constraints in  $C$  are satisfied. A CSP is said to be consistent if and only if it has at least one solution, otherwise, it is said to be inconsistent.

### 2.2 Belief Constraint Satisfaction Problem (BCSP)

A Belief Constraint Satisfaction Problem is a quadruplet  $(X, D, C_\alpha, R)$ , where:

- $X = \{x_1, \dots, x_n\}$  is a finite set of variables;
- $D = \{D_1, \dots, D_n\}$  is the set of their domains;

<sup>1</sup>The term uncertainty is commonly used, in the literature, to refer to imperfection. However, according to the taxonomy established in [9] uncertainty is one aspect of imperfection, whereas, imprecision is its other aspect.

- $C_\alpha = \{(C_1, \alpha_1), \dots, (C_m, \alpha_m)\}$  is a finite set of belief constraints (shortened as Bf-constraints) where  $(1 - \alpha_i)$  is the priority of the Bf-constraint  $C_i$ .
- $R = \{R_1, \dots, R_m\}$  is a finite set of imperfect relations associated to  $C_\alpha$ .

#### 2.2.1 Imperfect Relations

Under uncertainty, we cannot doubtlessly state whether or not an instantiation of values to variables is permitted with reference to a given constraint. To model imperfect (uncertain and/or imprecise) relations, we rely on a basic belief mass (bbm) distribution over instantiations.

An imperfect relation  $R_i$ , defined by a pair  $(V_i, \Theta_i)$ , associates a valuation  $V_i$  (a bbm distribution), over the frame of discernment  $\Theta_i$  of the associated Bf-constraint obtained by the Cartesian product of the involved variables domains, i.e.,  $\Theta_i = \{D_{i1} \times \dots \times D_{ik}\}$ . The valuation  $V_i$  is defined as follows:

$$V_i = m_i : 2^{\Theta_i} \rightarrow [0, 1] \mid \sum_{I \subseteq \Theta_i} m_i(I) = 1$$

Where  $I$  is a singleton or a subset of instantiations. In the following, we shall restrict ourselves to normalized bbm distributions where  $m_i(\emptyset) = 0$ .

If we may define an instantiation as a logical relation between variables, the finite amount of support (complete, partial or ignorant) enclosed in the bbm (i.e.,  $m_i(I)$ ) and derived from the available pieces of evidence can be interpreted as the potentiality degree of a given relation to, actually, occur so that the potentiality of the associated Bf-constraint to be satisfied by such variables instantiation(s),  $I$ . A second reading interprets this bbm distribution as preference levels induced over instantiations. Formally, let  $\theta_1$  and  $\theta_2$  two subsets of  $\Theta_i$  (i.e.,  $\theta_1, \theta_2 \subseteq \Theta_i$ ),  $m(\theta_1) > m(\theta_2)$  means that  $\theta_1$  is more believable (certain) than  $\theta_2$  and hence  $\theta_1$  is, a priori, preferred to  $\theta_2$  for the satisfaction of the given Bf-constraint. This latter reading shows that the bbm distribution has twofold purpose. The first is to quantify our belief about a given instantiation whereas the second is to induce a preorder, i.e., a priori preferences, among them.

The most appealing feature that makes the BFT an efficient tool is its faithfulness in recognition our knowledge as well as our ignorance. Obviously, the case of total knowledge matches the classical notion of perfect (certain and precise) relation where all tuples are known with certainty. This case is obtained through the certain belief function. Other kinds of relations may be modeled using the various functions offered by the BFT such as the vacuous belief function for uncertain and imprecise relation and the categorical belief function for certain but imprecise relation.

Likewise, some operations may be applied on imperfect relations, such as the vacuous extension and the marginalization. The vacuous extension of an imperfect relation  $R_i$  defined on  $S_i$ , to a larger set  $S'_i$ , such that  $S_i \subseteq S'_i$ , is an imperfect relation  $R_i^{(\uparrow S'_i)}$  defined on  $S'_i$  and obtained as follows:

$$m_i^{(\uparrow S'_i)}(\varphi) = \begin{cases} m_i(\theta) & \text{if } \varphi = \theta^{\uparrow S'_i} \text{ for all } \theta \subseteq \Theta_i \\ 0 & \text{otherwise} \end{cases}$$

Such that  $\theta^{\uparrow S'_i}$  denotes the cylindrical extension of the set  $\theta$  to  $S'_i$ . The vacuous extension is useful when we want to know to what extent a given instantiation may satisfy the Bf-constraint  $C_i$ . In fact, it corresponds to a refinement of knowledge.

The knowledge, initially, encapsulated in the bbm distribution can be refined as well as coarsened. The marginalization, which corresponds to a coarsening of knowledge, of an imperfect relation  $R_i$  defined on  $S_i$ , to a coarser set  $S'_i$ , i.e.,  $S_i \supseteq S'_i$ , is an imperfect relation  $R_i^{(\downarrow S'_i)}$  defined on  $S'_i$  and obtained as follows:

$$m_i^{(\downarrow S'_i)}(\varphi) = \sum_{\theta \subseteq \Theta_i: \theta^{\downarrow S'_i} = \varphi} m_i(\theta) \text{ for all } \varphi \subseteq \Theta_i$$

Such that  $\theta^{\downarrow S'_i}$  denotes the projection of the set  $\theta$  to  $S'_i$ . We can employ the marginalization when we want to know to what extent a given partial instantiation, if extended, may satisfy the Bf-constraint  $C_i$ .

### 2.2.2 Prioritized Constraints

In real world problems, not all the constraints are fully reliable. In fact, under uncertainty, the agent cannot entirely rely on some constraints that can be untrustworthy, misleading or that can be subject to change. For instance, in the agricultural production planning problem the reliability of some constraints depends on fluctuating prices and / or weather condition. In other words, some constraints may be relevant in a given circumstance but not in other circumstances. Regardless, this reliability affects the importance of each constraint to the set of solutions. For this reason, some constraints are prior to others.

To express priorities, we have recourse to the discounting principle provided by the BFT. First, we evaluate the reliability of each Bf-constraint  $C_i \in C$  using a discounting factor  $\alpha_i$ , so that, the smaller the reliability, the stronger the discounting. Second, we have to update the bbm distribution according to that factor. Let  $m_i$  be a bbm distribution related to the Bf-constraint  $C_i$  on the frame of discernment  $\Theta_i$  and let  $(1 - \alpha_i)$  be the confidence degree allocated to that Bf-constraint  $C_i$  that corresponds to its priority level.

The updated bbm, denoted by  $m_i^\alpha$  and induced from the old bbm  $m_i$  discounted by the coefficient  $\alpha_i$ , where every lost mass is reassigned to the universe of discourse, is obtained as follows:

$$m_i^\alpha(I) = \begin{cases} (1 - \alpha_i) m_i(I) & \text{if } I \neq \Theta_i \\ \alpha_i + (1 - \alpha_i) m_i(I) & \text{if } I = \Theta_i \end{cases}$$

such that  $\alpha_i \in [0, 1]$ , it is called the discounting factor which read as follows:

- $\alpha_i = 0$  means that the Bf-constraint  $C_i$  is fully reliable, so its priority is equal to 1 and the Bf-constraint is absolutely relevant. In this case, the bbm  $m_i$  is left unchanged.
- $\alpha_i = 1$  means that the reliability of the Bf-constraint  $C_i$  is totally doubtful, so its priority will be equal to 0, which means that it is possible to violate the Bf-constraint. In this case, all the information induced by the Bf-constraint  $C_i$  will be forthright discarded. The bbm  $m_i$  becomes a vacuous function that corresponds to the total ignorance state (i.e.,  $m_i^\alpha(\Theta_i) = 1$ ).

The priority determine the importance of the Bf-constraint. Let  $C_i$  and  $C_j$  be two Bf-constraints with priority levels, respectively,  $\alpha_i$  and  $\alpha_j$ , if  $\alpha_i < \alpha_j$  then the satisfaction of  $C_i$  is more relevant than the satisfaction of  $C_j$ . The notion of priority induces a preorder over the Bf-constraints.

### 2.2.3 Soft Constraints

After expressing our beliefs on the imperfect relations and updating those beliefs using priorities, we shall extract the satisfaction degree of the Bf-constraints by each given instantiation  $\theta$  in  $\Theta$  aside using the pignistic probabilities produced by the TBM pignistic transformation [8] of the bbm distribution.

Let  $C_i$  be a Bf-constraint,  $R_i$  its relation and let  $m_i$  be the associated bbm distribution over  $\Theta_i$ , the produced pignistic probability, denoted by  $BetP_i$ , is defined as follows:

$$BetP_i(\theta) = \sum_{\varphi \subseteq \Theta_i} \left( m_i(\varphi) \frac{|\theta \cap \varphi|}{|\varphi|} \right), \text{ for all } \theta \in \Theta_i$$

where,  $|\varphi|$  denotes the number of elements of  $\varphi$ . Hence, this notion of pignistic probability allows for expressing soft or flexible Bf-constraints starting from imperfect relations. It is of interest to discern the difference between hard constraint that should be certainly and fully satisfied and soft constraint whose satisfaction is not required to be neither certain nor total. Therefore, the satisfaction of a given constraint becomes, essentially, a matter of degree, such that:

- $BetP_i(\theta) = 1$  means that the instantiation  $\theta$  totally satisfies the Bf-constraint  $C_i$ ;

- $BetP_i(\theta) = 0$  means that the instantiation  $\theta$  totally violates the Bf-constraint  $C_i$ ;
- $0 < BetP_i(\theta) < 1$  means that the instantiation  $\theta$  partially satisfies the Bf-constraint  $C_i$ ;

A Bf-constraint  $C_i$ , whose scope is  $S_i$ , is said to be (totally or partially) satisfied by a given instantiation  $\theta \in \Theta_i$ , noted  $\theta \models C_i$  if and only if  $BetP_i(\theta) > 0$ . A Bf-constraint  $C_i$  is said to be unsatisfiable if there is no instantiation that satisfies it, i.e.,  $\forall \theta \in \Theta_i, BetP_i(\theta) = 0$ . Obviously, hard constraints are a particular case of Bf-constraints which are satisfied only to 1 or 0 degree.

#### 2.2.4 Preferences over the solutions

The  $BetP$  also induces a preorder among instantiations. Formally, let  $\theta$  and  $\theta'$  be two instantiations defined on the same set of variables,  $BetP_i(\theta) > BetP_i(\theta')$  means that is, a posteriori, preferred to  $\theta'$  for the satisfaction of the soft Bf-constraint  $C_i$ . Hence the  $BetP$  has twofold purpose as it allows expressing soft Bf-constraints and preferences among instantiations.

Classically, an instantiation  $\theta$  of a set of variables  $S \subseteq X$  is said to be consistent if and only if it satisfies all the constraints among that set. Within the BCSP view, the constraint satisfiability is not any more a yes/no query but a matter of degree and so the instantiation consistency is.

To get the consistency degree of an instantiation, we shall aggregate the satisfaction degrees of each Bf-constraint by the instantiation under consideration. In the literature, several aggregation functions have been proposed, especially, for decision models where uncertainty and imprecision are key issues. In order to select the appropriate function, we have considered three imperative criteria. First, the aggregation function should fulfill the most basic consistency principle, every instantiation that totally violates (i.e.,  $BetP_i(\theta) = 0$ ) at least one constraint is rejected. Hence, we need an aggregation operator that has an absorbent element  $a = 0$ . Second, we require that the unit interval  $[0, 1]$  be closed to the sought for aggregation function, so that, the resulted values may be easily interpretable and comparable. Finally, in order to avoid falling into the same weakness as the Fuzzy and the Possibilistic CSPs that suffer from the "drowning effect" because of the egalitarian min-max operators use which barely discriminate between instantiations that satisfy the CSP to the same degree, we propose using a utilitarian operator for aggregation.

Given these criteria, we find out that the most appropriate function may be the geometric mean. Formally, the consistency degree of an instantiation  $\theta$  of a set of variables

$S \subseteq X$  is obtained as follows:

$$\begin{aligned} \mathcal{C}(\theta) &= BetP_{\wedge} \{R_i | S_i \subseteq S\}(\theta) \\ &= \left( \prod_{R_i^{\uparrow S} | S_i \subseteq S} \{BetP_i\}(\theta) \right)^{1/k} \\ &= \left( \prod_{R_i | S_i \subseteq S} \{BetP_i\}(\theta^{\downarrow S}) \right)^{1/k} \end{aligned}$$

such that,  $k$  is the number of the Bf-constraints covering  $S$ .

- If  $\theta$  totally satisfies all the Bf-constraints covering  $S$ , it is said to be completely consistent, i.e.,  $\mathcal{C}(\theta) = 1$ .
- If  $\theta$  totally violates, at least, one Bf-constraint is said to be inconsistent, i.e.,  $\mathcal{C}(\theta) = 0$ .
- Otherwise, it is said to be partially consistent, i.e.,  $0 < \mathcal{C}(\theta) < 1$ .

As the BCSP is a generalization of the classical model, if the relations are perfect, a given instantiation is either consistent (1) or inconsistent (0).

A solution of BCSP ( $P$ ) is every consistent complete instantiation  $\theta$ , i.e., an instantiation of all the variables in  $X$  whose consistency degree is greater than 0, so that, all the Bf-constraints in  $C$  are satisfied. This consistency degree, evidently, corresponds to the satisfaction degree of the BCSP ( $P$ ) by that instantiation.

$$\mathcal{S}_P(\theta) = \mathcal{C}(\theta) = \left( \prod_{C_i \in C; R_i^{\uparrow X}} \{BetP_i\}(\theta) \right)^{1/m}$$

such that,  $m$  is the total number of the Bf-constraints covering  $X$ .

Accordingly, we can merely notice that the satisfaction degree of the BCSP, as defined above, accomplishes a sort of quantitative discrimination among the several instantiations inducing then a total preorder over them. Then, the higher is the satisfaction degree, the better is the instantiation.

The solution space of a BCSP ( $P$ ) of the set of all the feasible solutions, i.e.,

$$Sols(P) = \{\theta \in \{D_1 \times \dots \times D_n\} | \mathcal{S}_P(\theta) > 0\}$$

A BCSP ( $P$ ) is said to be:

- Totally consistent if and only if it has at least one solution that totally satisfies all the constraints of  $C$ , i.e.,  $\exists \theta \in Sols(P) | \mathcal{S}_P(\theta) = 1$ .
- Totally inconsistent if and only if all instantiations of  $X$  are inconsistent, i.e.,  $Sols(P) = \emptyset$  or also  $\forall \theta \in \{D_1 \times \dots \times D_n\} | \mathcal{S}_P(\theta) = 0$ .

- Partially consistent if and only if all solutions are somehow feasible, i.e.,  $Sols(P) \neq \emptyset | \forall \theta \in Sols(P), \mathcal{S}_P(\theta) < 1$ .

Toward the same view, the consistency degree of a BCSP is the satisfaction degree of its best solution, i.e.,

$$\begin{aligned} \mathbb{C}(P) &= \mathcal{S}_P(\theta^*) \\ &= \max_{\theta \in Sols(P)} (\mathcal{S}_P(\theta)) \\ &= \max_{\theta \in Sols(P)} \left( \prod_{C_i \in \mathcal{C}; R_i^{\uparrow X}} \{BetP_i\}(\theta) \right)^{1/m} \end{aligned}$$

### 3 A planning problem

In this section, we suggest a Belief CSP model for a simple vegetable crops production planning problem under uncertainty. The problem is to decide which crop to plant in which plot (a measured area of land). However, vegetable crops are, generally, cost expensive and of uncertain profitability due to the fluctuating prices and its dependence to weather condition that affects the harvest yields. The generic problem could be the following: a number of crops must be produced in a number of plots ( $a_i$ ). Each plot has a limited area and grows one single crop. Each crop has a profit ( $p_j$ ) and a labor-hour ( $h_i$ ) per unit area ( $1000m^2$ ) which are uncertain. The agriculturist's practical experience and his preferences are considered as pieces of evidence.

A farmer has to grow cucumber, pepper, potatoes, and peas in four plots  $a_1, a_2, a_3$ , and  $a_4$ . The total area of the land ( $L$ ) to be cultivated is  $100.000m^2(10ha)$  where  $a_1, a_2, a_3$ , and  $a_4$  represent, respectively, 40, 30, 20, and 10 percent of  $L$ . The farmer requires a minimum profit ( $R$ ) equal to 150.000 TND (Tunisian National Dinar) and a maximum labor-hour ( $H$ ) equal to 500 hours. As well, he prefers not to grow potatoes on the same plot for more than two years and after potatoes production, it is preferable to grow cleaning crops like cucumber to maintain the soil healthiness but he is doubtful about which crop he prefers more for  $a_1$ , pepper or peas. The evidence we have is that, last year, potatoes were grown in plot  $a_4$ .

A BCSP  $(X, D, C_\alpha, R)$  may be:

- $X = \{a_1, a_2, a_3, a_4\}$  is the set of four plots;
- $D = \{D_1, D_2, D_3, D_4\}$ , where  $D_1 = D_2 = D_3 = D_4 = \{cu_{(p_1, h_1)}, p_{(p_2, h_2)}, po_{(p_3, h_3)}, pe_{(p_4, h_4)}\}$  where  $cu, p, po$ , and  $pe$  stands respectively for cucumber, pepper, potatoes and peas;
- $C_\alpha = \{(C_1, \alpha_1), (C_2, \alpha_2), (C_3, \alpha_3)\}$  is a set of two

Bf-constraints.

$$\begin{aligned} C_1 &: \sum_{i,j=1}^n a_i.p_j \geq R; \alpha_1 = 0.4 \\ C_2 &: \sum_{i,j=1}^n a_i.h_j \leq H; \alpha_2 = 0.2 \\ C_3 &: a_i \neq a_k \forall i, k = 1..4 \text{ and } i \neq k; \alpha_3 = 0 \end{aligned}$$

- $R = \{R_1, R_2, R_3\}$  is a set of imperfect relations associated to  $C_\alpha$ .

Giving the uncertain values of  $p_j$  and  $h_j$ , the bbm distributions related to the imperfect relations  $R_1$  and  $R_2$  are, respectively, illustrated in Table 1.

The priority of the Bf-constraint  $C_3$  is equal to 1 (i.e.,  $1 - \alpha_3$ ), so it is fully reliable and absolutely relevant which means that it should be certainly and fully satisfied.  $C_3$  is a classical hard constraint. The associated relation  $R_3$  is a certain but imprecise relation where there is more than single instantiation may fully satisfy the Bf-constraint  $C_3$ .  $R_3$  is represented using the categorical bbm distribution  $m_3$  as follows:  $\forall \theta \in \Theta_3$ , if  $a_i \neq a_k \forall i, k = 1..4$  and  $i \neq k$  then  $\theta \in \varphi$  such that  $m_3(\varphi) = 1$ , otherwise,  $m_3(\theta) = 0$ . We can notice that  $|\varphi| = 4!$  complete instantiations that fully satisfy  $C_3$ .

**Table 1. The imperfect relations  $R_1$  and  $R_2$ .**

$R_i$	$m_i$
$R_1(\alpha_1 = 0.4)$	$m_1\{(a_1, p), (a_1, pe)\} = 0.5$ $m_1\{(a_4, po)\} = 0.3$ $m_1\{(a_2, cu), (a_2, p), (a_3, cu)\} = 0.1$ $m_1\{\Theta_1\} = 0.1$
$R_2(\alpha_2 = 0.2)$	$m_1\{(a_1, cu), (a_2, cu)\} = 0.3$ $m_1\{(a_4, cu)\} = 0.4$ $m_1\{(a_4, po), (a_3, po)\} = 0.1$ $m_1\{\Theta_2\} = 0.2$

Given the relative priorities, the updated bbm distributions  $m_1^\alpha$  and  $m_2^\alpha$  related, respectively, to the imperfect relations  $R_1$  and  $R_2$  are represented in Table 2.  $R_3$  is left unchanged.

At this level, as our problem model is updated, we can compute the satisfaction degrees of each Bf-constraint by any complete or partial instantiation  $\theta$ . For instance, let  $\theta_1 = \{(a_1, cu), (a_2, p)\}$  be a partial instantiation; the satisfaction degree of  $C_1$  by  $\theta_1$ , i.e.,  $BetP_1(\theta_1) = 0.08$  whereas the satisfaction degree of  $C_2$  by  $\theta_1$ , i.e.,  $BetP_2(\theta_1) = 0.17$ . Let  $\theta_2 = \{(a_1, cu), (a_2, p), (a_3, po), (a_4, pe)\}$  be a complete instantiation;  $BetP_1(\theta_2) = 0.32$  and  $BetP_2(\theta_2) = 0.25$ . If we get another complete instantiation  $\theta_3 = \{(a_1, pe), (a_2, p), (a_3, po), (a_4, cu)\}$ ;  $BetP_1(\theta_3) = 0.47$

**Table 2. The updated imperfect relations.**

$R_i$	Priority	$m_i^\alpha$
$R_1$ ( $\alpha_1$ = 0.4)	$1 - \alpha_1$ = 0.6	$m_1\{(a_1, p), (a_1, pe)\} = 0.3$ $m_1\{(a_4, po)\} = 0.18$ $m_1\{(a_2, cu), (a_2, p),$ $(a_3, cu)\} = 0.06$ $m_1\{\Theta_1\} = 0.46$
$R_2$ ( $\alpha_2$ = 0.2)	$1 - \alpha_2$ = 0.8	$m_1\{(a_1, cu), (a_2, cu)\} = 0.24$ $m_1\{(a_4, cu)\} = 0.32$ $m_1\{(a_4, po), (a_3, po)\} = 0.08$ $m_1\{\Theta_2\} = 0.36$

and  $BetP_2(\theta_3) = 0.45$ . We can notice that the instantiation  $\theta_3$  is preferred to  $\theta_2$  for the satisfaction of both of the soft Bf-constraints  $C_1$  and  $C_2$ . However, they are equally preferred for the satisfaction of  $C_3$  as  $BetP_3(\theta_2) = BetP_3(\theta_3) = 0.04$ .

Let us recall that every complete consistent instantiation is a possible solution for the BCSP. For example, one solution to the current BCSP may be  $\theta_2 = \{(a_1, cu), (a_2, p), (a_3, po), (a_4, pe)\}$  whose consistency degree  $\mathcal{C}(\theta_2) = 0.15$ , same as  $\theta_3 = \{(a_1, pe), (a_2, p), (a_3, po), (a_4, cu)\}$  whose consistency degree  $\mathcal{C}(\theta_3) = 0.2$ . However,  $\theta_3$  is preferred to  $\theta_2$ .

The best solution consistency degree is 0.2. Hence, the current BCSP is partially consistent as its consistency degree, i.e.,  $\mathbb{C}(P) = 0.2$ .

In order to make our framework valid, as a first step, we have adapted the very basic Backtrack algorithm to solve the BCSPs. However, given the sound basis of the BCSP framework, other interesting solving algorithms as well as consistency ones may be easily extended.

#### 4 Further work

The large-range expressivity brought by the BFT allows for covering both aspects of imperfection, uncertainty as well as imprecision, soft and prioritized constraints, and preferences over solutions. We may consider the BCSP as a unifying and general CSP framework where mapping to other frameworks is possible using the different kinds of relations. In addition, the BCSP is too close to the real world problems where an agent is not required to provide prior information and it takes into account his available knowledge and his preferences.

As a generalization of the standard CSP framework, the classical algorithms (e.g., the branch and bound algorithm, the consistency algorithms) can be adapted to our BCSP framework. Currently, we are working on implementing a specific algorithm for the BCSP employing some measures such as the belief and the plausibility offered by the BFT

and discussing its complexity.

#### References

- [1] Arthur P. Dempster. Upper and lower probabilities induced by a multivalued mapping. *The Annals of Mathematical Statistics*, 1967, 38(2):325–339.
- [2] Didier Dubois, Helene Fargier, and Henri Prade. Possibility theory in constraint satisfaction problems: Handling priority, preference and uncertainty. *Applied Intelligence*, 1996, 6:287–309.
- [3] Helene Fargier, Jerome Lang, and Thomas Schiex. Selecting preferred solutions in fuzzy constraint satisfaction problems. In *Proc. of the first European Congress on Fuzzy and Intelligent Technologies*, 1993, 3: 1128–1134.
- [4] Helene Fargier and Jerome Lang. Uncertainty in constraint satisfaction problems: a probabilistic approach. In *Proc. of the Second European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, 1993, 747:97–104.
- [5] Alan K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 1977, 8(1):99–118.
- [6] Ugo Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Information Sciences*, 1974,7(2):95–132.
- [7] Glenn Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, Princeton, NJ, 1976.
- [8] Philippe Smets. Belief functions. *Non-Standard Logics for Automated Reasoning*, 1988, 253–286.
- [9] Philippe Smets. Imperfect information: Imprecision Uncertainty. *Uncertainty Management in Information Systems. From Needs to Solutions*, 1997, 225–254.
- [10] Thomas Schiex. Possibilistic constraint satisfaction problems or How to handle soft constraints?. In *Proc. of the 8th International Conference on Uncertainty in Artificial Intelligence*, 1992, 268–275.

# DevOps Enhancement with Continuous Test Optimization

Dusica Marijan  
Simula, Norway  
dusica@simula.no

Sagar Sen  
Simula, Norway  
sagar@simula.no

**Abstract**—Growing evidence suggests the DevOps approach enables faster development and deployment, and easier maintenance of applications. Still, the efficiency of DevOps is constrained by long cycle times. This paper presents the approach for improving time-efficiency in DevOps, and in particular continuous integration testing, using continuous test optimization. The approach uses test redundancy analysis to discover test overlap with respect to feature interaction coverage, and based on detected redundancy to reduce the size of a test suite. Smaller-size test suites execute faster and enable shorter test cycles, which further enables shorter release cycles. The approach has been experimentally evaluated using an industrial case study, against three metrics: industry practice of test selection for continuous integration testing, retest-all approach, and random test selection. The results suggest that the proposed test redundancy detection and reduction efficiently reduces test cycles in CI compared to industry practice and retest-all approach, and improves fault-detection effectiveness compared to random test selection<sup>1</sup>.

**Index Terms**—DevOps, Continuous integration, Continuous integration testing, Test optimization, Test redundancy

## I. INTRODUCTION

DevOps is a growing development practice that promises to enable faster development and efficient deployment of applications without compromising on quality. The main principle underlying DevOps is increased communication and collaboration between development, testing, and operations, which makes it possible to minimize the time between making a change and deploying the change into production. Further, this makes it possible to respond faster to new and rapidly changing requirements, and thus remain agile. However, increased communication and collaboration do not suffice in reaching this goal on their own. A key aspect is enabling an *efficient and optimized continuous integration practice*. Continuous integration is a software development model of frequent integration and testing of source code to detect defects early in development. This practice has been associated with benefits such as improved code quality, more frequent releases, improved development productivity, less-costly development, and easier code maintenance [2], [3], [4]. The major bottleneck however in making continuous integration efficient is long-running testing of code changes after integration. In particular, after a change is made to the code, a set of automated tests are run to verify the change. As changes in DevOps are made frequently, testing of changes before integration runs

frequently, which amplifies the need for time-efficiency in testing. This implies that for each integration cycle test suites need to be optimized for short run-time and maximized fault-detection.

Existing techniques to improve the efficiency of software testing in continuous integration include test selection and prioritization [7], [8], [9], which aim to find an optimal combination and order of tests for achieving faster test runs. However, techniques that optimize testing for low run-time often compromise the ability of tests to detect faults [11], [12], [13], [14]. We argue that for DevOps, this aspect becomes especially important. It is essential that testing in DevOps is time-efficient, at the same time as being able to detect the most critical faults. Another issue that creates challenges for efficient testing in DevOps is the size of test suites. In particular, iterative feature requests and frequent changes of requirements constantly drive a test suite size larger and larger, while its quality is not maintained simultaneously. When test suites grow in size, the containing test cases start overlapping, covering the same features (parts of functionality) multiple times in different test instances. This effect is known as test redundancy, and it negatively affects time-efficiency of testing in DevOps. Besides directly increasing test effort (if all related test cases are post for running), test redundancy also increases test maintenance costs.

In this paper we propose an approach for improving time-efficiency of DevOps by continuously optimizing long running test cycles based on test redundancy detection and reduction. Specifically, for each test cycle we analyze test overlap with respect to changes and feature interaction coverage in a test suite, and then detect and remove tests that do not contribute to increased unique feature interaction coverage. We validated the approach in one industrial case study, comparing with industry practice and with the state of the art techniques. The approach demonstrated the improvement in time-effectiveness and fault detection effectiveness of CI testing in DevOps compared to industry practice.

The paper is organized as follows. In Section II we provide background and related work, as well as some challenges of efficient testing in DevOps in the presence of test redundancy. In Section III, we describe our approach for improving time-efficiency of DevOps with continuous test optimization based on redundancy detection. In Section IV we give results of the experimental study evaluating the approach against industry

<sup>1</sup>DOI reference number: 10.18293/SEKE2018-168

practice. We draw the conclusion in Section V.

## II. BACKGROUND AND RELATED WORK

In the following we revisit the key concepts underlying our work on making DevOps practices more cost-effective. We also summarize some of the critical challenges for effective testing in continuous integration and DevOps.

### A. DevOps

*DevOps* is defined as a set of software engineering practices that aim to build an agile relationship between development and operations. A key principle is constant communication and collaboration between development and operations, enabling benefits such as faster development and deployment of features into production, faster detection and correction of issues, and cost-effective running of dependable software with minimal risks. Existing approaches for improving testing in DevOps are still narrow. There is an approach reported for run-time monitoring and reporting to developers, referred to as the filling-the-gap tool, which enhances and automates the delivery of application performance information to the developer, with the goal of improving the quality of service or reducing maintenance cost [20]. However, more approaches addressing various other challenges in continuous integration testing for DevOps are missing, and such limited state of the art in this field gives even more motivation for our research.

### B. Continuous Integration

*Continuous integration (CI)* is a practice deemed a key enabler for DevOps. CI is a technique that continuously integrates code changes from all team members, merges them with the mainline, and verifies the changes against regressed aspects of the modified code (for unintended effects) with automated tests. The CI practice prevents working on isolated branches for too long, which over time start diverging from each other, leading to high effort of integrating such multiple branches into the mainline. An important aspect of CI is enabling rapid automated regression testing of code changes, which will give quick feedback to developers on the correctness of their changes. Since CI runs frequently, if it takes long time, it introduces time-inefficiency in DevOps. To support rapid regression testing in CI, we explore the concept of test optimization. Similarly, to improve CI testing, one approach was proposed combining test selection in the pre-commit stage with test prioritization in the post-commit stage [7]. However, this approach does not investigate the effect of test redundancy on the time-effectiveness of CI testing, which is one key focus of our work.

### C. Test Redundancy

Test redundancy can be defined with respect to coverage metrics, for example pairwise feature coverage, such that if two tests check interaction between the same pair of features, then one of these tests is redundant with respect to one another. Features represent smaller units of software-under-test functionality that are self-contained. Considering pairwise

feature coverage in the example provided below, two tests *TestA* and *TestB* cover the identical feature set  $\{b, c\}$ . As the pairwise feature set covered by *TestA* is a proper subset of the pairwise feature set covered by *TestB*, we say that *TestA* is redundant with respect to *TestB*.

$$\begin{aligned} \textit{TestA} &= [b, c] \rightarrow \{b, c\} \\ \textit{TestB} &= [a, b, c] \rightarrow \{a, b\}, \{b, c\}, \{a, c\} \end{aligned}$$

Test redundancy can be caused by multiple factors, such as test reuse in manual test specification, when existing tests are modified for testing new similar functionality, unintentionally leaving parts of already tested functionality. Other causes include incomplete requirements specification, redundancy of requirements, legacy, static test suites, parallel testing, or distributed testing [6]. In this work we are interested in analyzing test redundancy in integration tests, which test varying number of feature interactions to expose any faults in interaction between integrated individual code components.

### D. Regression Test Optimization

DevOps promotes iterative development, where smaller self-contained changes are made to software frequently. Every change is regression tested to check whether it introduces any faults caused by the interaction of integrated components. Since speed is one of the key requirements of efficient DevOps, regression testing requires only a set of relevant test cases. This is especially important in fast-evolving systems where test suites used for validating the correctness of systems grow quickly. Previous studies have shown that test suite size has a large impact on the overall test cost in the software development lifecycle [12], [15], [16]. Therefore, finding such a set of relevant test cases is the goal of regression test optimization. Specifically, regression test optimization aims to find an optimized set and order of regression tests that satisfy predefined optimization objectives. This includes selecting a relevant set  $S'$  based on  $S$ , known as test selection [17], and finding the execution order of tests in  $S'$ , known as test prioritization [19]. In this work we focus on test optimization guided by the analysis of test redundancy. i.e feature interaction overlap among different tests. Existing test optimization approaches have not been targeted towards DevOps and CI, and specifically have not been investigating the impact of test redundancy on the performance of CI testing in DevOps.

### E. Challenges of Testing in CI

Testing in continuous integration and DevOps is amenable to a number of challenges. First, it is highly **sensitive to long runtime**, since feedback on source code integration needs to be provided as rapidly as possible. Fast feedback enables faster test cycles, which further enable faster release cycles. Second, test effort needs to be steered towards achieving **just the quality required for deployment to staging or production**. If more effort is put into testing, this may negatively affect time- and cost-efficiency of testing. Third, since testing is time-limited in CI and DevOps, testing process (and in particular



test selection) should be *continuously optimized*, guided by risk analysis, based on the type of code changes made and their impact. Risk-analysis would enable defining a dynamic regression scope for each build and test iteration, with multiple layers of tests to enable iterative, and faster feedback. Fourth, as test suites grow overtime, to cover new functionality added to the codebase, tests start to overlap, building *test redundancy*. This creates the risk of increased test effort as many similar tests may seem relevant, and therefore selected for running. Therefore, the key challenge lies in identifying test redundancy and selecting test cases so that redundancy is minimized. This will help reduce long test runtime, and will enable reaching just the required level of quality. Furthermore, high levels of *test automation* are needed, in test selection and optimization (apart from test execution, where automation is considered a prerequisite for CI and DevOps).

### III. IMPROVING THE EFFICIENCY OF CI WITH CONTINUOUS TEST OPTIMIZATION

The approach for improving efficiency and effectiveness of CI that is proposed in this paper exploits the idea of *continuous test optimization based on test redundancy analysis*. As stated previously, one critical challenge of CI and DevOps is enabling short and effective test cycles given redundant test suites. A redundant test suite contains test cases that overlap, given a specific feature coverage criteria. In this context, cost-effective testing entails a trade-off between the size of a test suite and its comprehensiveness. Here, we refer to comprehensiveness as the ability of a test suite to detect faults caused by interactions between two features (pairwise coverage). For the sake of simplicity, in this paper we will restrict our approach to pairwise coverage only, while the proposed concept can be extended to any-wise feature coverage (consequently entailing some higher computational complexity). To better illustrate the complexity of CI testing in the presence of redundant test case with overlapped feature coverage, we present the following example.

#### A. Illustrating Example

A software system under test consists of a set of functionality modules, referred to as *features*  $FS = \{f_1, f_2, \dots, f_n\}$ . Features are used to build a set of solution configurations for video conferencing. Some features are  $f_1 = \text{video\_resolution}$ ,  $f_2 = \text{audio\_resolution}$ ,  $f_3 = \text{audio\_protocol}$ ,  $f_4 = \text{point\_to\_point\_calls}$ ,  $f_5 = \text{multi\_party\_calls}$ . A test suite  $TS = \{t_1, t_2, \dots, t_m\}$  is developed for testing these solution configurations, where test cases partially cover the total set of features  $FS$  in different combinations.  $Cov(t_i) = \{f_1, f_2, \dots, f_n\}$  denotes a set of features tested by a test case  $t_i$ . As the system under test evolves incrementally through continuous development and testing,  $TS$  evolves continuously and grows larger. New test cases are added covering new functionality, but also interactions between new and old functionality. The same features become part of multiple tests, but because of large size of a test suite, the same combinations of features often

become part of multiple tests. This in turn increases test effort, as the same interactions between system functionality are executed multiple times. In the example shown below, four test cases  $t_1$ ,  $t_2$ ,  $t_3$ , and  $t_4$  cover a set of features  $\{f_1, f_2, f_3, f_4, f_5\}$  in different combinations. Considering a pairwise interaction coverage as a criterion for test redundancy analysis, the covering set of features for tests  $t_1$  and  $t_4$  overlap with the covering set of features for tests  $t_2$  and  $t_3$  respectively. Since this overlap represents proper subsets i.e.  $Cov(t_1) \subset Cov(t_2)$ ,  $Cov(t_4) \subset Cov(t_3)$ , tests  $t_1$  and  $t_4$  are redundant with respect to tests  $t_2$  and  $t_3$ .

$$\begin{aligned} Cov(t_1) &= \{f_1, f_2\} \\ Cov(t_2) &= \{\{f_1, f_2\}, \{f_1, f_3\}, \{f_2, f_3\}\} \\ Cov(t_3) &= \{\{f_1, f_4\}, \{f_1, f_5\}, \{f_4, f_5\}\} \\ Cov(t_4) &= \{f_4, f_5\} \\ Cov(t_1) &\subset Cov(t_2), Cov(t_4) \subset Cov(t_3) \end{aligned}$$

#### B. The Approach

Our approach to reducing test cycles with redundancy detection and reduction explores the concept of *total test redundancy* and *partial test redundancy*. To explain these concepts, we introduce the following definitions.

*Def 1:* A test case  $t_1$  is totally redundant of a test case  $t_2$ , if  $Cov(t_1) \subseteq Cov(t_2)$ .

*Def 2:* A test case  $t_1$  is partially redundant of a test case  $t_2$ , if  $Cov(t_1) \neq Cov(t_2)$  and  $Cov(t_1) \cap Cov(t_2) \neq \emptyset$ .

In the first step of the approach, we address total redundancy, by detecting test cases whose covering set of features is fully covered by another test case. After such test cases have been identified, we remove them from a test suite. In the second step of the approach we address partial redundancy by combining interaction coverage metrics with historical fault detection effectiveness of tests obtained from test logs. The underlying idea is that partially redundant test cases which have historically exhibited good fault revealing performance can be classified as non-redundant, and otherwise as redundant. This idea is supported by studies showing that test execution history can help improve cost-effectiveness of testing [7], [8], [9], [21].

*Step 1:* Given the system under test and the changes to its source code, as well as an existing test suite, we first find a set of tests impacted by the changes. This is performed automatically, using association links between test cases and features (software functionality) covered by the test cases. Next, for the obtained test suite, we analyze test overlap between test cases to find those test cases whose covering feature interactions are completely covered by other test cases. We remove such test cases from a test suite, obtaining a test suite with only non-redundant and partially redundant test cases.

*Step 2:* Next, we look into partially redundant test cases contained within the test suite. These are the test cases whose covering feature interactions are partially covered by other test cases. We obtain execution history for these test cases

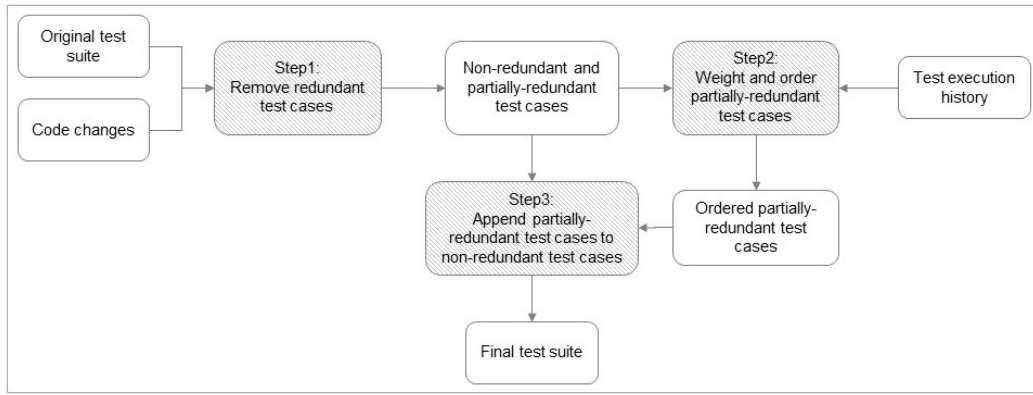


Fig. 1. Redundancy detection and reduction methodology.

(for those that have been executed before) and analyze their fault detection effectiveness in past executions. In particular, we assign different weights to test cases chronologically, based on how recently they detected faults. Those tests that detected faults more recently have higher weight than those that detected faults earlier. If  $n$  is the number of historical test execution iterations, then the weight  $\omega$  is calculated as follows:

$$\omega = \begin{cases} \frac{1}{n}, & \text{for test cases that have been executed before} \\ 1, & \text{for newly added test cases} \end{cases}$$

Next we sort the test cases based on  $\omega$  value, such that those test cases with higher fault detection effectiveness are ordered higher in the list of partially redundant test cases. The rationale for sorting is that CI is bound by tight time constraints. Therefore, we want to ensure that the most important test cases are run first, in case that available time budget for testing is smaller than the time required for running the whole test suite.

*Step 3:* Finally, the resulting test suite consists of test cases that have been identified as non-redundant in *Step 1*, which come first in order, followed by a set of partially redundant test cases, ordered based on their fault detection effectiveness. The approach is schematically presented in Figure 1.

#### IV. EXPERIMENTAL CASE STUDY

We conducted a set of experiments to evaluate the effectiveness of the proposed approach for test redundancy reduction in CI and DevOps. The approach was applied to an industrial case study of optimizing CI development and testing of video conferencing telepresence systems developed by Cisco Norway. These systems enable full "in-person" meeting experience with high-end video communication and collaboration between multiple parties using cameras with eye-tracking features and directional microphones which allow the transmission of high definition video and audio, as well as wireless sharing of presentations and other documents.

The objective of the experiments was to evaluate the *test execution time* and *fault-detection effectiveness* of CI testing when using our approach for test redundancy reduction. We

compared the approach to *existing industry practice of CI testing*, *retest-all* approach, and *random test selection*. Specifically, we are interested in answering the following research questions:

- RQ1:** What effect does the redundancy reduction approach (*RED*) have on the duration of test cycles in CI compared to industry testing practice (*IP*)?
- RQ2:** What effect does the redundancy reduction approach have on the duration of test cycles in CI compared to retest-all (*RA*) approach?
- RQ3:** What effect does the redundancy reduction approach have on fault detection effectiveness of test suites compared to random test selection (*RS*)?

##### A. Industrial System

The industrial system used in the experimental case study represents a video conferencing system that is developed following a DevOps practice. The system is highly complex, implementing around one hundred features, which are used in different combinations across different solution configurations for video conferencing. As the system is developed in CI, code updates are made frequently. Each update is followed by integration testing cycle, to verify the correctness of newly added functionality and to ensure that no regressions are introduced in existing functionality. The test suite is developed incrementally and is large in size, covering system features in various combinations of interactions. Extensive test suites for integration and configuration testing, together with the requirements for short test cycles make cost-effective testing of video conferencing systems in DevOps challenging. The test data used in the experiments consists of 400 test cases. Test execution history for these test cases is available for 6 last test execution runs, which gives 2400 test case executions. Historical data includes test execution result (pass/fail) and test execution duration.

##### B. Experiment Setup and Methodology

We address the posed three research questions RQ1-RQ3 in experiments E1, E2, and E3, respectively.

In experiment E1, we compare *RED* with *IP*, in terms of total test execution time. First, for the features affected by changes, we select a set of impacted test cases. These tests represent the initial test suite. Then we obtain execution history for these test cases, and apply our redundancy reduction approach to the initial test suite. We analyze test overlap in terms of pairwise feature coverage, and remove test cases whose covering set of pairwise feature interactions is completely contained within another test case(s). At this point, the test suite contains non-redundant test cases and partially redundant test cases. Next, for all partially redundant test cases, we analyze which of them have shown good fault-detection performance in the past. Based on this information, we eliminate test cases that have not contributed to increased fault detection effectiveness. Next, we calculate weights for the rest of partially redundant test cases, rewarding more recent fault detection higher. We sort these test cases according to their weight, and append them to the previously identified non-redundant test cases. For the resulting set of test cases, we measure the percentage reduction of the test suite size compared to the size of the original test suite (used by industry practitioners for testing the changes). We run experiment E1 5 times, for all available historical test execution data.

In experiment E2, we compare *RED* with *RA*, in terms of total test execution time. We chose *RA* as a comparison metric because this is a commonly used approach to regression testing in practice. Specifically, we modified the *RA* in a way to include only the tests affected by changes, versus retesting the entire test suite. Therefore, for the features affected by changes, we select a set of impacted test cases. These tests represent the initial test suite. Then we apply our redundancy reduction approach to the initial test suite, and examine test overlap in terms of pairwise feature coverage. We remove redundant test cases based on this criterion, which gives a test suite containing non-redundant test cases and partially redundant test cases. Then we analyze execution history for the partially redundant test cases, and eliminate those which historically have not showed to increase fault detection. Next, we order remaining partially redundant test cases based on their recent fault detection performance and append them to the previously identified non-redundant test cases. This gives us the final test suite. For this test suite, we measure the percentage reduction of the test suite size compared to the size of the initial test suite.

In experiment E3, we compare *RED* with *RS*, in terms of fault-detection effectiveness. We compare with *RS* because this is a commonly used alternative to automated guided test selection and reduction, primarily driven by low cost and low complexity. First, for the features affected by changes, we select a set of impacted test cases, which represent the initial test suite. Then we apply the same setup as in E1 to obtain the set of non-redundant test cases followed by an ordered set of partially redundant test cases, as the final reduced test suite. Next we measure the total test execution time for the final test suite (time limit), based on historical test execution time of each test case. Afterwards, we start randomly selecting test

cases from the initial test suite, accumulating execution time of each selected test case. We repeat this process until the time limit is reached. The resulting test suite obtained in this process is the randomly selected test suite. Now we measure the loss of fault detection of the randomly selected test suite compared to the fault-detection of the final test suite. The measured value is the percentage of faults that were detected by the final test suite and not by the randomly selected test suite. Because of the randomness in RS approach, we repeat the experiment 100 times.

### C. Results and Analysis

In this section we present the results of the experiments E1, E2, and E3, addressing research questions RQ1, RQ2, and RQ3, respectively. The results are graphically presented in Figure 2, Figure 3, and Figure 4, respectively.

1) *Time-effectiveness Compared with Industry Practice*: In the first experiment addressing RQ1, we compared *RED* and *IP* in terms of test suite execution time. The results show that *RED* was able to reduce test cycle by 30% on average compared to *IP*. The results are shown in Figure 2. Y axis corresponds to the percentage reduction of test execution time of the reduced test suite compared to the test suite used by practitioners.

2) *Time-effectiveness Compared with Retest-All*: In the second experiment aimed to answer RQ2, we compared *RED* and *RA* in terms of test suite execution time. In this experiment, *RA* showed to reduce total test suite execution time by 35% compared to *RA*. The results are shown in Figure 3. Y axis corresponds to the percentage reduction of test execution time of the reduced test suite compared to retest-all approach.

3) *Fault-detection Effectiveness Compared with Random Selection*: In the third experiment addressing RQ3, we compared *RED* with *RS* in terms of fault detection effectiveness, for the same (given) test budget. The results demonstrate that *RED* can achieve up to 70% better fault detection compared to randomly selected test cases, and 40% on average better fault detection compared to randomly selected test cases, for the test suites used in the experiment. The results are shown in Figure 4. Y axis shows the distribution of the percentage of fault detection effectiveness gain of the reduced test suite compared to randomly selected test suite.

In summary, the results of the experiments E1, E2, and E3 demonstrate that the proposed approach can effectively reduce test cycles in CI compared to industry practice of CI testing by 35% on average, and compared to retest-all approach by 30% on average. The results further demonstrate that the proposed approach can improve fault detection effectiveness of a test suite compared to random test selection up to 70%, for the same test time budget.

### D. Threats to Validity

A threat to external validity of the results is the choice of the industrial case study of continuous integration testing and the test dataset. While the used industrial context is an example of good industry practice, we cannot say that it is representative,

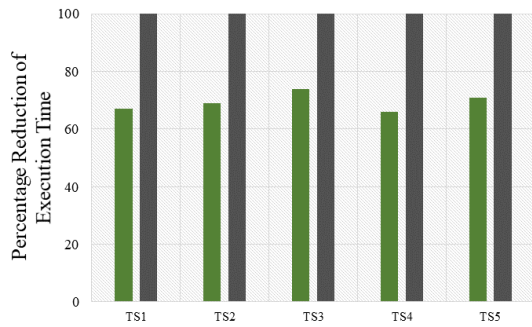


Fig. 2. Comparison of the proposed approach with *industry\_practice* approach in terms test suite execution time.

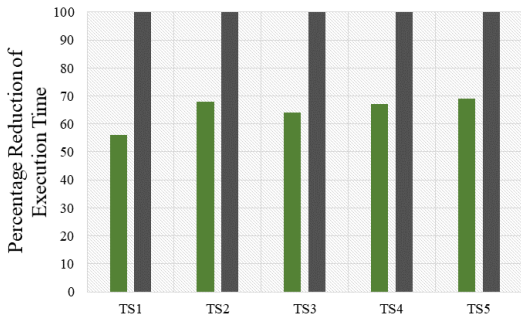


Fig. 3. Comparison of the proposed approach with *retest\_all* approach in terms of test suite execution time.

and continuous integration testing can be applied differently in different companies. More studies are needed to verify whether our results generalize to other practices of CI and DevOps. This is part of our future work. A threat to internal validity could be potential faults in our implementations of the optimization algorithms. We have thoroughly tested the code to ensure that these threats are minimized.

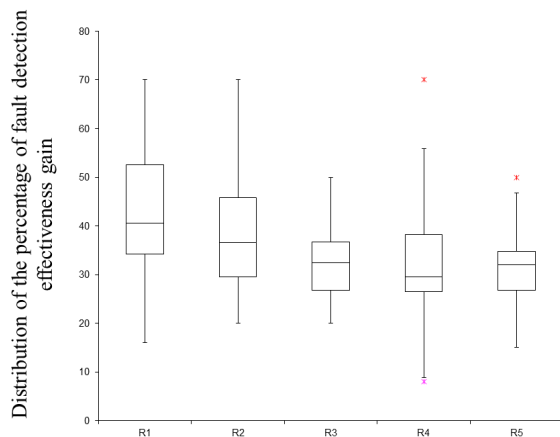


Fig. 4. Comparison of the proposed approach with *random\_test\_selection* approach in terms fault detection effectiveness.

## V. CONCLUSION

In this paper we proposed the approach for improving time-efficiency of DevOps using continuous test optimization. The approach is based on test redundancy analysis in terms of feature interaction coverage. By reducing test redundancy, it is possible to reduce CI test cycles and further release cycles in DevOps. The approach has been evaluated and has demonstrated improvement in time-efficiency compared to industry practice, retest-all approach, and random test selection.

## ACKNOWLEDGMENT

This work is supported by The Research Council of Norway, through the Certus SFI project. We thank to Cisco Systems Norway for using their test data set and for fruitful discussions that contributed to this work.

## REFERENCES

- [1] Economic Benefits of HP Future Smart Agile Transformation, Evidence and case studies (continuousdelivery.com).
- [2] B. Vasilescu, Y. Yu, H. Wang, P. Devanbu, and V. Filkov, Quality and productivity outcomes relating to continuous integration in GitHub, International Symposium on the Foundations of Software Eng., 2015.
- [3] A. Miller, A hundred days of continuous integration, AGILE, 2008.
- [4] M. Leppanen, S. Makinen, M. Pagels, V. P. Eloranta, J. Itkonen, M. V. Mantyla, and T. Mannista, The highways and country roads to continuous deployment, IEEE Software, 2015.
- [5] M. Hilton, N. Nelson, T. Tunnell, D. Marinov, D. Dig, Trade-offs in continuous integration: assurance, security, and flexibility, 11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE), 2017.
- [6] L. Bass, I. Weber, L. Zhu, DevOps: A Software Architect's Perspective, Addison-Wesley Professional, 2015.
- [7] S. Elbaum, G. Rothermel, J. Penix, Techniques for Improving Regression Testing in Continuous Integration Development Environments, International Symposium on the Foundations of Software Engineering, 2014.
- [8] D. Marijan, A. Gotlieb, S. Sen, Test Case Prioritization for Continuous Regression Testing: An Industrial Case Study, IEEE International Conference on Software Maintenance (ICSM), 2013.
- [9] D. Marijan, M. Liaaen, Effect of Time Window on the Performance of Continuous Regression Testing, ICSME, 2016.
- [10] D. Marijan, M. Liaaen, Test Prioritization with Optimally Balanced Configuration Coverage, HASE, 2017.
- [11] G. Rothermel, M. Harrold, J. Ostrin, and C. Hong, An empirical study of the effects of minimization on the fault detection capabilities of test suites, Int. Conference on Software Maintenance (ICSM), 1998.
- [12] G. Rothermel, M. J. Harrold, J. Ronne, and C. Hong, Empirical studies of test-suite reduction, Software Testing Verification and Rel., 2002.
- [13] W. Wong, J. Horgan, A. Mathur, and A. Pasquini, Test set size minimization and fault detection effectiveness: a case study in a space application, 21st Computer Software and App. Conference (COMPSAC), 1997.
- [14] W. E. Wong, J. R. Horgan, S. London, and A. P. Mathur, Effect of test set minimization on fault detection effectiveness, ICSE, 1995.
- [15] H.-Y. Hsu and A. Orso, Mints: A general framework and tool for supporting test-suite minimization, ICSE, 2009.
- [16] Y. Yu, J. A. Jones, and M. J. Harrold, An empirical study of the effects of test-suite reduction on fault localization, ICSE, 2008.
- [17] M. Grindal, B. Lindstrm, J. Offutt and S. F. Andler, An evaluation of combination strategies for test case selection, Empirical Software Engineering, vol. 11, 2006.
- [18] A. Gotlieb, D. Marijan, FLOWER: optimal test suite reduction as a network maximum flow, Int. Symp. on Soft. Testing and Analysis, 2014
- [19] Y.-C. Huang, K.-L. Peng and C.-Y. Huang, A history-based cost-cognizant test case prioritization technique in regression testing, Journal of Systems and Software, vol. 85, 2012
- [20] J.F. Perez, W. Wang and G. Casale, Towards a DevOps Approach for Software Quality Engineering, Workshop on Challenges in Performance Methods for Software Development, 2015.
- [21] J.M. Kim and A. Porter, Technique for Regression Testing in Resource Constrained Environments, International Conference on Software Engineering, 2001.

# Reducing the Cost of Android Mutation Testing

Lin Deng

Department of Computer and Information Sciences  
Towson University, Towson, Maryland  
ldeng@towson.edu

Jeff Offutt

Department of Computer Science  
George Mason University, Fairfax, Virginia  
offutt@gmu.edu

**Abstract**—Due to the high market share of Android mobile devices, Android apps dominate the global market in terms of users, developers, and app releases. However, the quality of Android apps is a significant problem. Previously, we developed a mutation analysis-based approach to testing Android apps and showed it to be very effective. However, the computational cost of Android mutation testing is very high, possibly limiting its practical use. This paper presents a cost-reduction approach based on identifying redundancy among mutation operators used in Android mutation analysis. Excluding them can reduce cost without affecting the test quality. We consider a mutation operator to be *redundant* if tests designed to kill other types of mutants can also kill all or most of the mutants of this operator. We conducted an empirical study with selected open source Android apps. The results of our study show that three operators are redundant and can be excluded from Android mutation analysis. We also suggest updating one operator’s implementation to stop generating trivial mutants. Additionally, we identify subsumption relationships among operators so that the operators subsumed by others can be skipped in Android mutation analysis.

## I. INTRODUCTION

Mobile applications (mobile apps) are software programs specifically developed for mobile devices. Due to the convenience of mobile devices, people use mobile apps more often than applications on other platforms [1]. Approximately 85% of mobile devices use the Android operating system [2]. In March 2018, more than 3.6 million Android apps are available for download on the Google Play Store [3]. However, many Android apps contain software faults, and users often experience problems. An Android analysis organization [3] found that 14% of Android apps are “low-quality.”

Our prior work applied mutation testing to testing Android apps [4], explored the feasibility of Android mutation testing [5], and empirically evaluated its fault detection effectiveness using naturally occurring faults and crowdsourced faults [6]. The results show that Android mutation testing is very effective at detecting both types of software faults.

However, Android mutation testing can be expensive in several ways. Due to the constraints on size, weight, and power consumption, most Android devices are equipped with hardware that is slower than desktops and laptops. Executing and testing Android apps take more execution time than traditional software programs. Moreover, while Android mutation testing has been found to be effective at designing high-quality test cases and assessing test cases generated by other testing techniques, the number of mutants that need to be executed increases the cost of Android mutation testing. For example,

executing 20 tests, one minute for each test, on 1,000 Android mutants may require up to 13.8 days.

This paper presents our experimental evaluation that tries to speed up mutation testing by finding redundant mutation operators that can be excluded from Android mutation testing, while still maintaining fault-detection effectiveness. Specifically, this experimental study analyzed redundancy among the 19 Java traditional mutation operators [7] and the 17 Android mutation operators [5].

The results of our study show that three mutation operators are redundant and can be excluded without reducing the effectiveness of Android mutation testing: (1) Unary Arithmetic Operator Deletion (AODU), (2) Unary Arithmetic Operator Insertion (AOIU), and (3) Logical Operator Insertion (LOI). Also, the design of the Activity Lifecycle Method Deletion (MDL) operator requires further improvement. Furthermore, the Button Widget Switch (BWS) operator subsumes Button Widget Deletion (BWD), and Operator Deletion (ODL) subsumes Constant Deletion (CDL), Conditional Operator Deletion (COD), and Variable Deletion (VDL). In addition, mutants created by the Fail on Back (FOB) operator, the TextView Deletion (TVD) operator, and the Orientation Lock (ORL) operator are very hard to kill.

This paper is organized as follows. Section II introduces background on Android mutation testing. Section III describes the experiment used to identify the redundancy among Android mutation operators, then, presents and analyzes the experiment results. Section IV gives an overview of related research. Section V discusses threats to validity, and the paper concludes and suggests future work in Section VI.

## II. BACKGROUND

In 1978, DeMillo et al. invented mutation testing [8], a syntax-based software testing technique that is very effective at designing high-quality tests and evaluating pre-existing tests. Mutation testing modifies a software artifact such as source code, to create new versions, called *mutants*. *Mutation operators* define the rules that specify the changes that are made to a software artifact. Testers design test cases to cause mutants to behave differently from the original, then the mutants are called *killed*. Well designed mutation operators can lead to very powerful test cases. The more mutants a test set can kill, the more effective the test set is at finding faults.

Mutation operators have been created for many different languages, including C and Java [7], [9]. Our prior work [4]–[6] used the novel programming features, unique characteristics, and testing challenges of Android apps to design and

TABLE I: Android Mutation Operators

Category	Android Mutation Operator
Event-based	Intent Payload Replacement (IPR)
	Intent Target Replacement (ITR)
	OnClick Event Replacement (ECR)
	OnTouch Event Replacement (ETR)
Component Lifecycle	Activity Lifecycle Method Deletion (MDL)
	Service Lifecycle Method Deletion (SMDL)
	Button Widget Deletion (BWD)
XML-related	EditText Widget Deletion (TWD)
	Activity Permission Deletion (APD)
	Button Widget Switch (BWS)
	TextView Deletion (TVD)
Common Faults	Fail on Null (FON)
	Orientation Lock (ORL)
	Fail on Back (FOB)
Context-aware	Location Modification (LCM)
Energy-related	WakeLock Release Deletion (WRD)
Network-related	Wi-Fi Connection Disabling (WCD)

evaluate 17 Android mutation operators, as listed in Table I. We also used 15 Java traditional method-level mutation operators [7] and four deletion mutation operators [10], [11].

### III. EMPIRICAL EVALUATION

Normally, all mutation operators are applied to generate mutants. This creates lots of mutants that must be executed many times at significant cost. Recent research [12] has found that between 90% and 99% of mutants are *redundant* in the sense that any test that kills another mutant is guaranteed to kill the redundant mutant. If redundant mutation operators can be identified and excluded, the cost of mutation will be significantly reduced. All mutants generated from the same mutation operator are of the same *type*. Thus, we use this study to determine whether mutants of one type are killed by the tests designed to kill mutants of other types.

In particular, this empirical evaluation tries to evaluate the redundancy in Android mutation testing by addressing the following research questions:

**RQ1:** How many mutants of one particular type can be killed by tests created to kill another type of mutants?

**RQ2:** Which types of mutants are less likely to be killed by tests created to kill other types of mutants?

**RQ3:** Can any mutation operator be excluded or improved without significantly reducing effectiveness?

#### A. Experimental Subjects

This experimental evaluation used 12 Android classes and their XML layout and configuration files from four open source Android apps: *JustSit* [13], *MunchLife* [14], *TippyTipper* [15], and *Tipster* [16]. Table II provides an overview of the projects. The 19 Java traditional mutation operators [7] generated 1,947 muJava mutants and the 17 Android operators generated 1,018 mutants. The number of muJava mutants ranged from four for *About.java* in *TippyTipper* to 534 for *MunchLifeActivity.java* in *MunchLife*, and the number of Android mutants ranged from one for *AndroidManifest.xml* in *MunchLife* to 258 for *JustSit.java* in *JustSit*.

#### B. Redundancy Scores

The mutation-adequate test set  $T_i$  includes tests that are specifically designed to kill all the mutation of type  $i$ . To quantify the redundancy among Java traditional mutation operators

TABLE II: Details of Experimental Subjects

Apps	Components	LOC	XML Nodes	muJava Mutants	Android Mutants
JustSit	JustSit.java	444		394	258
	main.xml		13		
	About.java	48		9	13
	about.xml		6		
	RunTimer.java	99		131	25
	run_timer.xml		3		
	JsSettings.java	61		28	31
	settings.xml		6		
	AndroidManifest.xml		14	0	4
	MunchLife	MunchLifeActivity.java	384		534
main.xml			12		
Settings.java		68		47	8
preferences.xml			5		
TippyTipper	TippyTipper.java	239		105	198
	main.xml		20		
	SplitBill.java	134		124	49
	SplitBill.xml		31		
	Total.java	279		231	115
	Total.xml		44		
	About.java	30		4	14
	About.xml		10		
	Settings.java	61		13	15
	Tipster	TipsterActivity.java	297		327
	main.xml		30		
<b>Total</b>		<b>2144</b>	<b>204</b>	<b>1947</b>	<b>1018</b>

and Android mutation operators, Praphamontipong and Offutt [17] defined the redundancy score  $r_{i,j}$  to be:

$$\text{Redundancy Score: } r_{i,j} = \frac{m_{i,j}}{M_j} \times 100\% \quad (1)$$

where,  $m_{i,j}$  is the number of mutants of type  $j$  killed by the mutation-adequate test set  $T_i$ , and  $M_j$  is the total number of non-equivalent mutants of type  $j$ .

In other words, the redundancy score  $r_{i,j}$  is the percentage of mutants of type  $j$  killed by a test set that is adequate for type  $i$ . For example, a program has 100 non-equivalent Relational Operator Replacement (ROR) mutants and 200 non-equivalent Arithmetic Operator Insertion (AOIS) mutants. A tester designs a test set that kills all the non-equivalent AOIS mutants, getting an AOIS mutation-adequate test set. If this AOIS mutation-adequate test set also kills 60 ROR mutants, the redundancy score  $r_{AOIS,ROR}$  in this program is  $60 \div 100 = 60\%$ .

Note that for a given subject app, according to the definition above, every possible pair of mutation operators has a redundancy score. Then, across all the subject apps in an experimental evaluation, there are multiple redundancy scores for the same pair of mutation operators with different values. For example,  $r_{AOIS,ROR}$  may be 60% in subject  $s_1$ , 50% in  $s_2$ , and 40% in  $s_3$ . Consequently, a score that can represent the overall redundancy relationship is required. Praphamontipong and Offutt [17] defined the average redundancy score ( $r_{average,i,j}$ ) to be the average value of all the  $r_{i,j}$  of the operator in all experimental subjects. The average redundancy score is not weighted, i.e., we compute the redundancy score for each subject app, then calculate the average of the scores.

Redundancy score indicates quantitatively whether a mutation operator is redundant or not. For example, if a mutation operator has a redundancy score of 0%, it means no tests that were designed to kill other types of mutants killed any mutants of this type. That is, the mutation operator is not redundant.

However, if a mutation operator has a redundancy score of 100% for the tests that are specifically designed to kill mutants of another type, it means this operator is totally redundant and does not contribute anything to the quality of tests. Excluding it from the mutation analysis can reduce cost without reducing effectiveness. If a mutation operator has a redundancy score of 50%, half of the mutants generated by this operator are killed by the tests designed for other types of mutants. Some programs do not use all language features, thus the relevant mutation operators cannot be used to generate tests. Then, no tests will be designed for this mutation type.

### C. Experimental Procedure

This study includes four steps to obtain the redundancy scores among the mutation operators:

- 1) **Generate mutants:** Given a subject, apply the 19 Java traditional mutation operators and the 17 Android mutation operators to generate mutants.  $m_n$  represents the mutants created by operator  $n$ .
- 2) **Eliminate equivalent mutants and design tests:** For each set of mutants  $m_n$ , eliminate all equivalent mutants. Then, design a set of test cases to kill all the non-equivalent mutants, denoted by  $t_n$ , that is, tests designed to kill the mutants of type  $n$ . We design tests independently for each type of mutants. No redundant tests are introduced once all the mutants are killed.
- 3) **Execute tests:** For each set of test cases  $t_n$ , execute all tests on all mutants.
- 4) **Compute the redundancy scores:** For each pair of mutation operators and for each subject app, compute the redundancy score  $r_{i,j}$ . Then, to get an overview across all the subjects in the experiment, compute an average redundancy score for each mutant type.

Our tool implements a multithreading controller to parallelize the execution with multiple emulators and real devices. The tool executes on a MacBook Pro with a 2.6 GHz Intel i7 processor and 16 GB memory to control 8 emulators and 12 Motorola MOTO G Android smartphones in the experiment. All devices run on the Android KitKat operating system.

### D. Experimental Results

This section presents experimental results and key findings.

#### **RQ1: How many mutants of one particular type can be killed by tests created to kill another type of mutants?**

Table III shows the average redundancy scores across all the subject apps. The columns represent mutation operators, and rows represent tests designed to kill all mutants of that type. So, for example, the tests designed to kill all AODU mutants ( $test\_AODU$ ) killed 18.2% of the AOIS mutants. Some pairs mutant types never showed up in the same program, so their tests could not kill mutants of the other type. For example, LOR and CDL mutants never appeared together, so  $test\_LOR$  is marked “n/a” for CDL, and vice versa.

Four Java traditional mutation operators, ASRS, LOD, SOR, and AODS, did not generate any mutants, and four Android mutation operators, ETR, LCM, SMDL, and WCD, did not generate any mutants. Thus, they are not listed in Table III. WRD mutants are also excluded because testers need to use the

*dumpsys* tool to check system information, thus they cannot be redundant with other types of mutants. APD mutants are excluded because the principle of APD is to try all possible tests to identify those *un-killed* APD mutants, instead of designing tests to kill mutants.

#### **RQ2: Which types of mutants are less likely to be killed by tests created to kill other types of mutants?**

According to the results in Table III, three Android mutation operators were found to be very hard to kill. On average, only 6.4% of Fail on Back (FOB) mutants were killed by the mutation adequate test sets of other mutation operators, with the highest redundancy score of 33.3%. FOB injects a “Fail on Back” event handler into every Activity class. Since Android apps are event-based programs, their execution flows rely heavily on events initiated by user actions. The Back button lets users move backward to the previous Activity, interrupting the usual execution flow. It is usually not on the “happy path” from the perspective of software design, and results in a common fault of Android apps, that is, the software fails when the Back button is clicked. To kill FOB mutants, testers need to design tests that press the *Back* button at least once at every Activity. However, in this experiment, very few tests designed for other mutation operators included the user action of clicking the *Back* button.

Very few TextView Deletion (TVD) mutants were killed. On average, less than 1% of TVD mutants were killed by the mutation adequate test sets of other mutation operators, and its highest redundancy score was 8.3%. Since TextView widgets cannot be edited by users, they usually do not associate with any user events, nor require event handlers from the implementation of the app. However, TextView widgets are widely used by developers to present essential information. TVD deletes TextView widgets from screens one at a time. Killing a TVD mutant needs a test to ensure that the TextView widget displays correct information. Very few tests checked TextView widgets’ contents, unless the TextView widget was used to display some variable results, such as a tip amount.

Very few Orientation Lock (ORL) mutants were killed. On average, only 2.5% of Orientation Lock (ORL) mutants were killed by the mutation adequate test sets of other mutation operators, and its highest redundancy score was 12.5%. Most mobile devices have the unique feature of being able to change the screen orientation. To use to this feature, many apps change their layout of the GUI when the orientation changes. However, different screen sizes and resolutions on different devices make switching the orientation difficult for the developers, leading to faults. ORL mutants freeze the orientation of an Activity by inserting a special *locking* statement into the source code, so that no switching actions can be accepted by the app. To kill ORL mutants, testers need to design tests that explicitly change the orientation, then check whether the GUI structure is displayed as expected after switching the orientation. In this experiment, no other mutation operators consider switching the screen orientation, so there was no redundancy.

#### **RQ3: Are any Android mutation operators redundant enough to be excluded, or can any be improved? In particular, can the mutants of one type always be killed by tests created to kill another type?**





on the screen. *Subsumption* is used to theoretically compare test criteria: a criterion C1 *subsumes* another criterion C2, if every test that satisfies C1 is guaranteed to satisfy C2 [18]. In mutation testing, an operator MO1 *subsumes* another operator MO2 if a test set that kills all mutants of MO1 is guaranteed to kill all mutants of MO2. Thus, BWS subsumes BWD, that is, every test set designed to kill all the BWS mutants can kill all the BWD mutants. As a result, when users include BWS in the Android mutation analysis, excluding BWD mutants will **not** affect test effectiveness. Note that if an Activity only has one button widget, BWS cannot generate any mutants. This is because to achieve *switching*, the Activity must display at least two buttons. Thus, it is recommended to disable BWD when there are BSW mutants, and enable it otherwise.

The Conditional Operator Deletion (COD) mutation operator also has six “1.000” values (second highest), and the Constant Deletion (CDL) mutation operator has five “1.000” values (third highest). Also, the ODL test sets killed all the mutants of CDL, COD, and the Variable Deletion mutation operator (VDL). The Operator Deletion mutation operator (ODL) was originally designed by Delamaro et al. [11]. It deletes each arithmetic, relational, logical, bitwise, and shift operator from all expressions. CDL deletes each constant in an expression, and VDL deletes each variable in an expression. Figure 3 shows example ODL, CDL, and VDL mutants. According to the definitions, it is guaranteed that ODL subsumes CDL and VDL. COD deletes unary conditional operators. Figure 3 also shows that ODL and COD generate the same mutants. Therefore, ODL theoretically subsumes COD. Not surprisingly, test cases designed to kill ODL mutants also kill CDL, COD, and VDL mutants, which means when using ODL, we can exclude CDL, COD, and VDL.

Original: <code>int x = y + 2 ;</code>	ODL Mutant_1: <code>int x = y ;</code>
	ODL Mutant_2: <code>int x = 2 ;</code>
	CDL Mutant: <code>int x = y ;</code>
	VDL Mutant: <code>int x = 2 ;</code>
Original: <code>int x = - y ;</code>	ODL Mutant: <code>int x = y ;</code>
Original: <code>if (! isError) { x = y ; }</code>	ODL Mutant: <code>if (isError) { x = y ; }</code>
	COD Mutant: <code>if (isError) { x = y ; }</code>

Fig. 3: Example ODL, CDL, VDL, and COD Mutants

The Unary Arithmetic Operator Insertion (AOIU) inserts a minus sign in front of integer variables. The Logical Operator Insertion (LOI) inserts a bitwise complement operator in front of integer variables. 50.3% of AOIU mutants and 44.9% of LOI mutants were killed by test\_FOB tests, which are simple tests that only launch an Activity and click the Back button.

<code>int level = 1; current_level.setText ( Integer.toString ( level ) );</code> // Original
<code>int level = 1; current_level.setText ( Integer.toString ( -level ) );</code> // AOIU Mutant
<code>int level = 1; current_level.setText ( Integer.toString ( ~level ) );</code> // LOI Mutant

Fig. 4: AOIU and LOI Examples

Figure 4 gives example AOIU and LOI mutants. In Android apps, each GUI widget is assigned a resource ID that is recorded as an integer number. These resource IDs are stored and managed in XML files. Both AOIU and LOI generate many mutants by mutating the resource IDs in Android apps. Figure 5 shows an example where AOIU changes the resource ID of *upbutton*. However, once a resource ID is changed and not mapped to its original GUI widget, the Android app will immediately crash after launched, making the mutant trivial and redundant. That is, any test case that launches the app can kill this mutant. Similarly, LOI also generates trivial mutants. Therefore, when using mutation testing for Android apps, we recommend to exclude AOIU and LOI.

<code>Button upbutton = (Button) findViewById ( R.id.upbutton );</code> // Original
<code>Button upbutton = (Button) findViewById ( - R.id.upbutton );</code> // AOIU

Fig. 5: AOIU Changes Android Resource ID

In summary, we recommend the following:

- 1) Exclude AODU, because of its highest average redundancy scores
- 2) Improve the design of MDL, because MDL generates trivial mutants
- 3) Exclude BWD when using BWS, because BWS subsumes BWD
- 4) Exclude AOIU and LOI, because around 50% of AOIU and LOI mutants are trivial
- 5) Exclude CDL, COD, and VDL when using ODL, because ODL subsumes them

### E. Re-evaluating the Effectiveness

Based on the evaluation results, we provide recommendations to eliminate the redundancy among Android mutation operators. However, it is not clear whether the effectiveness of Android mutation testing still holds after removing and modifying redundant mutation operators. Due to the high computational cost of Android mutation testing, re-conducting the whole effectiveness evaluation in Section III would take several months. Thus, we elected to check the results on one subject app.

According to the recommendations, we updated the implementation of our Android mutation testing tool. We took Tipster as the subject app for the re-evaluation. Originally, Tipster generated 327 muJava mutants and 130 Android mutants. After removing and modifying redundant mutation operators, Tipster generated 259 muJava mutants and 125 Android mutants, with an overall 16% reduction in terms of the total number of the mutants. After that, a new set of mutation adequate tests was designed. Originally, Tipster had 64 crowdsourced faults, in which 51 were detected by the old mutation adequate test set. After re-conducting the evaluation, the newly designed mutation adequate test set using fewer and less redundant mutants found the same 51 crowdsourced faults in Tipster. Therefore, it is concluded that removing and modifying redundant mutation operators in this research did not impact the effectiveness of Android mutation testing.

#### IV. RELATED WORK

Traditional mutation testing uses three types of approaches to reduce cost: *do-fewer*, *do-smarter*, and *do-faster* [19]. As a *do-fewer* approach, *selective mutation* was introduced by Wong and Mathur to choose a subset of mutation operators [20]. The muJava tool selects 15 operators to preserve almost the same test coverage as non-selective mutation [7]. Empirical studies in both Java and C show that the Deletion mutation operators are able to result in very effective tests with much lower cost [10], [11]. This study, as a *do-fewer* approach, also discussed them in Android mutation testing.

#### V. THREATS TO VALIDITY

Similar to most experiments in software engineering, this empirical evaluation has several threats to validity.

**Internal validity:** In this experiment, we designed only one set of Android mutation-adequate tests for each type of mutant. The results of redundancy scores may differ for different Android mutation-adequate tests. Also, in this experimental study, we identified all the equivalent mutants by hand. Manual work could introduce human errors.

**External validity:** We cannot guarantee that the selected subjects are representative. The results and redundancy scores may differ from the results in this study if we used different subject apps. To improve the ability to compare results, we chose Android apps that have previously been used in other Android testing studies.

**Construct validity:** The implementation of our Android mutation testing tool and the associated mutation operators may include software faults. In this study, we constantly tested the experimental environment to ensure reliability.

#### VI. CONCLUSIONS AND FUTURE WORK

Android mutation testing is an effective approach to design and evaluate tests for Android apps. However, due to the unique conditions of Android devices and apps, the cost of Android mutation testing can be very expensive, in terms of computational time and effort. We conducted an empirical study to identify redundancy among mutation operators, with the goal of finding mutation operators that are redundant and do not contribute to the quality of tests.

The results of our study show that three Java traditional mutation operators (AODU, AOIU, and LOI) are redundant in Android mutation analysis. Excluding them can save costs without reducing test quality. As BWS subsumes BWD, we recommend skipping BWD mutants when BWS is used. As ODL subsumes CDL, COD, and VDL, these three can be excluded if ODL is used. Our study indicates that three Android mutation operators (FOB, TVD, and ORL) have very low average redundancy scores (6.4%, 0.7%, and 2.5%). They are very hard to kill by other types of tests. Also, we provide a recommendation for improving the design of MDL to stop generating trivial mutants.

Kurtz et al. [21] found that traditional mutation scores are inflated during mutation analysis, so are flawed as a test quality measurement device. Since a very strong and rich test set is needed to perform minimal mutation analysis and compute dominator mutation scores, we did not include them into this

study, due to the expensive cost. For future work, we hope to use minimal mutation analysis and dominator mutation scores to verify the conclusions in this study.

#### ACKNOWLEDGMENT

This work was partly funded by The Knowledge Foundation (KKS) through the project 20130085: Testing of Critical System Characteristics (TOCSYC).

#### REFERENCES

- [1] Kleiner Perkins Caufield & Byers, "Internet trends 2015," Online, May 2015, <http://www.kpcb.com/internet-trends>, last access September 2015.
- [2] International Data Corporation, "Smartphone OS market share, 2017 Q1," Online, May 2017, <https://www.idc.com/promo/smartphone-market-share/os>, last access March 2018.
- [3] "Android apps on Google Play," 2018, <http://www.appbrain.com/stats/number-of-android-apps>, last access March 2018.
- [4] L. Deng, N. Mirzaei, P. Ammann, and J. Offutt, "Towards mutation analysis of Android apps," in *Tenth Workshop on Mutation Analysis (Mutation 2015)*, April 2015, pp. 1–10.
- [5] L. Deng, J. Offutt, P. Ammann, and N. Mirzaei, "Mutation operators for testing android apps," *Information and Software Technology*, vol. 81, pp. 154 – 168, 2017.
- [6] L. Deng, J. Offutt, and D. Samudio, "Is mutation analysis effective at testing android apps?" in *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, July 2017, pp. 86–93.
- [7] Y.-S. Ma, J. Offutt, and Y.-R. Kwon, "MuJava : An automated class mutation system," *Software Testing, Verification, and Reliability*, Wiley, vol. 15, no. 2, pp. 97–133, June 2005.
- [8] R. A. DeMillo, R. J. Lipton, and F. G. Sayward, "Hints on test data selection: Help for the practicing programmer," *IEEE Computer*, vol. 11, no. 4, pp. 34–41, April 1978.
- [9] H. Agrawal, R. DeMillo, R. Hathaway, W. Hsu, W. Hsu, E. Krauser, R. J. Martin, A. Mathur, and G. Spafford, "Design of mutant operators for the C programming language," Software Engineering Research Center, Purdue University, West Lafayette, IN, Technical Report SERC-TR-41-P, March 1989.
- [10] L. Deng, J. Offutt, and N. Li, "Empirical evaluation of the statement deletion mutation operator," in *6th IEEE International Conference on Software Testing, Verification and Validation (ICST 2013)*, Luxembourg, March 2013.
- [11] M. E. Delamaro, J. Offutt, and P. Ammann, "Designing deletion mutation operators," in *7th IEEE International Conference on Software Testing, Verification and Validation (ICST 2014)*, Cleveland, Ohio, March 2014.
- [12] P. Ammann, M. E. Delamaro, and J. Offutt, "Establishing theoretical minimal sets of mutants," in *7th IEEE International Conference on Software Testing, Verification and Validation (ICST 2014)*, Cleveland, OH, March 2014, pp. 21–30.
- [13] (2010) JustSit. <https://play.google.com/store/apps/details?id=com.brocktice.JustSit>, last access September 2016.
- [14] (2014) MunchLife. <https://play.google.com/store/apps/details?id=info.bpace.munchlife>, last access September 2016.
- [15] (2013) TippyTipper. <https://code.google.com/p/tippytipper>, last access September 2016.
- [16] I. Darwin, "Tipster," 2016, <https://github.com/IanDarwin/Android-Cookbook-Examples/tree/master/Tipster>, last access September 2016.
- [17] U. Praphamontripong and J. Offutt, "Finding redundancy in web mutation operators," in *Twelfth Workshop on Mutation Analysis (Mutation 2017)*, March 2017, pp. 134–142.
- [18] P. Ammann and J. Offutt, *Introduction to software testing*, 2nd ed. Cambridge University Press, 2017, ISBN 978-1107172012.
- [19] J. Offutt and R. Untch, "Mutation 2000: Uniting the orthogonal," in *Proceedings of Mutation 2000: Mutation Testing in the Twentieth and the Twenty First Centuries*, San Jose, CA, October 2000, pp. 45–55.
- [20] W. E. Wong, M. E. Delamaro, J. C. Maldonado, and A. P. Mathur, "Constrained mutation in C programs," in *Proceedings of the 8th Brazilian Symposium on Software Engineering*, Curitiba, Brazil, October 1994, pp. 439–452.
- [21] B. Kurtz, P. Ammann, J. Offutt, M. E. Delamaro, M. Kurtz, and N. Gökçe, "Analyzing the validity of selective mutation with dominator mutants," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ser. FSE 2016. New York, NY, USA: ACM, 2016, pp. 571–582.

# A Test Case Generation Method Based on State Importance of EFSM for Web Application

Junxia Guo, WeiWei Wang, Linjie Sun, Zheng Li and Ruilian Zhao  
College of Information Science and Technology  
Beijing University of Chemical Technology  
Beijing, China  
Email: gjxia; lizheng; rlzhao@mail.buct.edu.cn

<sup>1</sup> **Abstract**—Test cases generation is a principal process in web application testing. Most existing methods generate test cases for improving test efficiency mainly from the aspects like minimizing the test case suite, increasing the code coverage, and so on. However, similar with traditional software having important functions, classes or modules, some web states are more vital than others in web applications. It can be thought that those vital web states relatively have higher influence on the performance of web application. So, they should be given more attention in test case generation. In more detail, the importance of web states can be measured from its page contents or topological structures. Meanwhile, as we known Model-based Testing is a kind of widely used approach in automatic test case generation. Therefore in this paper we propose an EFSM based test case generation method considering the importance of web states for web applications. The experimental results show that our methods can deterministically enhance the testing efficiency of web application.

**Index Terms**—Web application testing; test case generation; Web state importance; EFSM model;

## I. INTRODUCTION

With the high speed development of Internet, web applications have become widespread and still increase rapidly. Web applications testing becomes more and more difficult. As a main part of testing, test cases generation is crucial to improve the test efficiency. Unquestionable, good test cases can detect bugs faster. So, test case generation approaches for web applications aim to generate better test cases automatically. There are several kinds of methods with respect to test case generation for web applications, for instance, Capture/Replay-based methods, source code analysis-based methods and model-based methods. Most of those methods generate test cases for improving test efficiency mainly from the aspects like minimizing the test case suite, increasing the code coverage, and so on. However, similar with traditional software having important functions, classes or models[1], some web states are more vital than others in web applications. These web states relatively have higher influence on the performance of web application. They should be tested preferentially. So, the test case generation for web applications should give more attention on these web states.

In order to generate test cases that can cover the important web states, we need to, firstly, find those web states in a web

application. As we known, web states of a web application can be described with a graph, such as a state-flow graph, a state-transition graph and so on. Although the importance of nodes can be calculated according to the algorithms of graphic theory, the importance is only reflected from the topology of graph, without referring to the importance of a web page itself. Meanwhile, nowadays, more and more new dynamic page techniques are adopted in web applications in order to get better user experience with the help of client-side scripting or server-side scripting or both of them. Asynchronous JavaScript and XMLHttpRequest (*Ajax*) is a set of those techniques which can change content dynamically without reloading the entire page, hence can decrease the information transfer between client side and server side. This makes the traditional methods that use a web page as a web state for web application test become not suitable. So, we need to pay more attention to the change of Document Object Model (*DOM*) structures.

Model-based testing (*MBT*) is a promising paradigm for test case generation. Therefore, in this paper, we firstly propose a method of building Extended Finite State Machine(*EFSM*) model based on session data for web applications, in which a state node represents a web state in the format of combining a URL and DOM structure. Then, we measure the importance of states from that of page contents or topological structures, and give an algorithm to evaluate the importance of web states. Finally we present a test case method based on the state importance of EFSM model.

The primary contributions of this paper are as follows:

- 1) Propose a method for building EFSM model based on session data, which can illustrate a web application more explicitly.
- 2) Present a test case generation method based on the importance of web states, which considers the aspects of page contents and topological structures.
- 3) Implement a prototype tool and empirically evaluate our EFSM model building method and test case generation method. The results show that our methods are usable and effective.

The rest of this paper is organized as follows. Section II presents an overview of the related work. Section III describes how to construct an EFSM model for a web application based on its session data in detail. Section IV explains the algorithm

<sup>1</sup>DOI reference number: 10.18293/SEKE2018-177

for evaluating the importance of web states from the aspects of page contents and topological structures. Section V reports the experimental results and analysis. Finally, conclusion and future work are given in Section VI.

## II. RELATED WORK

In this paper, we broadly categorise test case generation techniques for web applications into three groups, including Capture/Replay-based techniques, source code analysis-based techniques, model-based techniques. Capture/Replay-based techniques are one of notable trends for the automated test case generation and execution in the area of web application testing in recent years. Currently the most popular capture-replay tools for AJAX testing are Selenium[2], Sahi[3], and Watir[4], which can test DOM-based web applications by capturing events information from user interaction. Such tools need to access the DOM, can assert expected UI behaviour defined by the tester and replay the events[5]. However, a substantial amount of manual efforts are required for testing.

Source code analysis-based techniques generate test cases based on the information which are obtained from the page contents of client-side pages or the code structure of server-side code of web applications. Such as, Wang et al.[6] proposes a static analysis approach for automatic generating test cases for web applications. In this approach, source code is analyzed to extract interfaces which are composed of input parameters with domain information and user navigation map which is composed of all the possible URLs from web application source code. Then through the navigation graph, a set of paths is selected and test cases are generated for each path. Guodong et al.[7] present SymJS, a symbolic framework for both pure JavaScript and client-side web programs. The tool contains a symbolic execution engine for JavaScript, and an automatic event explorer for web pages. Mark Harman et al.[8] introduce three related algorithms and a tool for automated web application testing using Search Based Software Testing (*SBST*). Their approach starts with a static analysis phase that collects static information to aid the subsequent search based phase and produces a test suite that maximizes branch coverage of the server-side application under test.

Model-based testing(MBT) is an increasingly wide-used approach which has gained much interest in recent years, from academic as well as industrial domain. The idea of MBT is to create and maintain a model that contains the information about the structure and possibly about the desired behaviours of a web application. Test cases are then derived either manually or automatically from the model with algorithms that systematically cover the model using so called selection criteria. Ricca and Tonella[9] proposed a UML model for web application which incorporates static and dynamic aspects of web application and re-interpreted it as a graph. They developed a tool ReWeb, which is used to create the model, and another tool TestWeb, which is used to generate and execute a set of test cases based on the model built by ReWeb. Alessandro Marchetto et al.[10], [11] proposed a state-based testing approach, specially designed to exercise

Ajax web applications. The DOM of the page manipulated by the Ajax code is abstracted into a state model. Callback executions triggered by asynchronous messages received from the web server are associated with state transitions. Test cases are derived from the state model based on the notion of semantically interacting events. QI et al.[12] present a combinatorial strategy for full form test. This approach aims to exploring more states and building a complete automated test model for a web application. Miguel[13] proposed a test generation and filtering technique for model-based testing for web applications. A model contains the information of all UI Test Patterns linked with connectors. Each UI Test Pattern contains its specific configurations with the data needed for test execution. Priti Bansal et al.[14] proposed a Model Based Test Case Generation technique, in which the model represents the navigation behavior of a web application. The related information is derived from requirements and low level design. Traversing the model can generate test sequences which can be incorporated with input data to generate test cases later.

## III. METHOD FOR BUILDING EFSM MODEL BASED-ON SESSION DATA

To describe a web application explicitly, we use the combination of URL and DOM structure as a state. In addition, in order to ensure the executability of generated test cases, the detailed information about how to trigger the transitions is also necessary, including the trigger event, preconditions and follow-up actions. EFSM is a widely used model which consists of states and transitions. It is an enhanced model which adds the preconditions of transitions and actions based on Finite State Machine (*FSM*). The information about preconditions of transitions and follow-up actions can be depicted on EFSM model. Thus we assume that EFSM is suitable for describing web applications.

An EFSM model is formally represented as a 6-tuple  $(S, S_0, I, V, O, T)$ , where  $S$  is a finite set of states,  $S_0 \in S$  is an initial state named START,  $I$  is a set of input declarations,  $V$  is a finite set of internal/context variables,  $O$  is a set of output declarations,  $T$  is a finite set of transitions. Each member of  $I$  is expressed as event(input parameters) meaning event occurs with a list of input parameters. Each member of  $O$  is described as action. Each transition  $t \in T$  is represented by a 5-tuple  $(source(t), target(t), event(t), condition(t), action(t))$ , where  $source(t) \in S$  is the start state of transition  $t$ ,  $target(t) \in S$  is the target state,  $event(t) \in I$  is an incentive event or empty,  $condition(t)$  is the preconditions performing transition  $t$ , and  $action(t)$  represents a sequence of actions[15], [16].

The framework of EFSM model building method is shown in Figure1. It is mainly consisted of four processes. 1) Get state and transition-related information through crawling client-side. 2) Get transition-related information through session data. 3) Combine the information of client-side with the session data. 4) Build EFSM model. We present the details in following subsections.

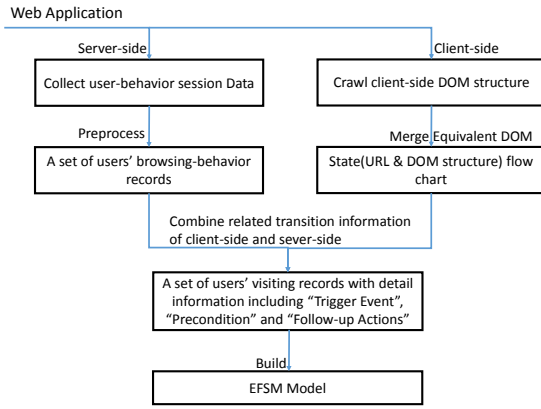


Fig. 1. Method Framework of Building EFSM Model

### A. DOM Structures and Transition-related Information Collection from Client-side

In this paper, we use a tool named Crawljax[17] to get the DOM structures of a web application. Crawljax works in the mode of depth-first crawl rule. The result includes the states information and the transition-related information which could be got from client-side.

There may exist duplicate states that are triggered by different events and handled by the same script function. Meanwhile, there are states which have the same structure but have different contents, for example different text content. Therefore, in order to avoid state explosion, the above two kinds of states need to be reduced and combined. Firstly, we abstract the DOM structure of all states by filtering the structure-unrelated elements, such as text content, time stamp and so on. Then we judge the similarity of two states by using the edit distance of two DOM structures, which is calculated through Levenshtein[18] method. Finally, we reduce all the same states and combine all the similar states which have high similarity.

### B. Session Data Preprocessor

Session data, a kind of log data which can record the original information when a user visits a web application and sends requests. Based on those data, we can get the whole visiting trails of every user. In addition, we can get the transition information through them. In this paper, we collect the session data from server-side. The detailed information of session data mainly includes *User IP*, *Session ID*, *Date & Time*, *Request Event*, *Request URL*, *Referrer URL* and *Responded HTTP Code*.

The preprocessing of session data mainly has three procedures, which are clearing up unusable records, user identification and dividing users' visiting trails. We use session ID to identify different users. There are mainly two kinds of methods for separating user's visiting sequence. One is based on the referrer information. Another is according to the visiting time threshold. In this paper, we united the two kinds of methods as the rule, which can be described as: Record a user's visiting trail based on the page URL referrer. If the interval of the user visiting is larger than 30 minutes, we confirm the user start a

new visiting. The time threshold is determined according to the related work[19], [20].

### C. Transition-related Information Integration

User-operations of web application may interact with the server-side or not. For example, when deleting a mail in the email system, the selection operation do not interact with server-side while the deletion operation do. In order to get integrated transition-related information, we need to combine the related information got from client-side and server-side.

For example a email deletion operation usually has two steps. Select the check-box to mark the email. Then click the "delete" button to finish the operation. The first step just has client-side information recorded about this transition. The second step has interaction with server-side. The related information will be recorded in session data. The information of those two steps will be combined and recorded to the related users' visiting sequence. The relationship diagram of records in session data could be illustrated as Figure2.

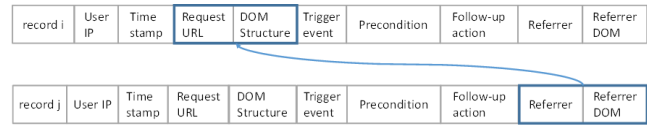


Fig. 2. Relationship Diagram of Records in Session Data

### D. Build EFSM Model

As mentioned above, the item of  $\langle \text{URL, DOM-structure} \rangle$  is a state in our method. The information of *Trigger Event*, *Precondition*, *Follow-up Action* is integrated transition information. According to the relationships between item  $\langle \text{Referrer URL, Referrer DOM} \rangle$  and  $\langle \text{Request URL, DOM Structure} \rangle$  shown in Figure2, which describes the transition sequence of states, the EFSM model could be built. The pseudo-code of building EFSM model algorithm is shown in Algorithm 1.

#### Algorithm 1 EFSM Model Construction

---

**Require:** integrated user session  
**Ensure:** an EFSM model description file for the web application

```

1: EFSM.init(); //store the model information
2: while session ≠ NULL do
3:   while record ≠ NULL do
4:     if record.state NotIn EFSM.state then
5:       EFSM.state.add(record.state);
6:     end if
7:     t.init(record); //initialize the transition t
8:     if t NotIn EFSM.trans then
9:       EFSM.trans.add(t);
10:    end if
11:  end while
12: end while
  
```

---

## IV. TEST CASE GENERATION METHOD CONSIDERING WEB STATE IMPORTANCE

We consider the importance of web states mainly from two aspects. One aspect is from the topological structure. Another is from the importance of contents on related web page. Based on the important value of each state, we use traditional Genetic Algorithm(GA) to generate test cases from the EFSM which is built using the method described in SectionIII. We present our method's detail in following subsections.

### A. Evaluate Web State Importance

The content of a web page is mainly the text content. The purpose of evaluating page content importance is to derive the importance of the textual content displayed in the current page over other web pages of the web application. For a web application, we can assume that the pages with important content are visited frequently. Thus the web pages containing critical contents are more important for the web application.

The topological structure of web states is the inter-dependency between the related web pages which can be divided into data dependency and link dependency. Data dependency refers to the operation of requesting data from the server during the transition from the current state to a new state. Link dependency refers to a simple jump from the current state to a new state without data interaction.

Therefore, the importance of a web state can be evaluated by combining the importance of the state in topological structure and the related content. In this paper, we firstly calculate content importance of the web page for each state. Then we calculate the state importance directly by introducing the content importance into topological importance calculation.

1) *Content importance of a web page*: By traversing the state of EFSM model, we can get all web pages for all states. For each web page, we can get a vector model of page content. A keyword vector model reflecting the entire web application is inferred with all pages' vector models. Comparing the vector model of a page and the keyword vector model of the web application, the content importance of the web page can be calculated.

When trigger a transition according to the information of precondition, the textual content of the web page for destination state can be extracted from the HTML code using tag reduction method. The word segmentation operation is performed on the textual content and the number of occurrences of each word is counted. Word segmentation and word frequency constitute a vector model of the page content, which can be formalized as:  $State(v) = \{w1 : m1, w2 : m2, , wi : mi\}$ . Where  $w$  is the word appearing in the page and  $m$  is the occurrence frequency of the word. After obtaining the vector models of all pages, a keyword vector model of the web application is inferred with all pages' vector models  $Key = \{w1 : n1, w2 : n2, , wi : ni\}$ .

The content importance ( $Con\_Importance$ ) of a web page can be calculated through comparing the vector model of the page ( $State(v)$ ) and the keyword vector model of the web application ( $Key$ ). The calculation formula is as follows.  $Con\_Importance(v) = \sin(State(v), Key) = \cos\theta =$

$$\frac{State(v) \cdot Key}{\|State(v)\| \cdot \|Key\|}$$

2) *State importance evaluation*: The state importance evaluating algorithm presented here draws from the PageRank algorithm's logic, whose core idea is that the page with high importance relies on the important pages, and the page relying on high importance pages is of high importance[21].

Including the content importance, we consider 5 factors totally, which are the importance of the states which current

state has data or link dependency and those depend on current state. We divide those 5 factors in two attributes: Authority and Hub. Authority refers to the summed importance of the states on which the current state depends. There are mainly three parts to measure authority: data dependency, link dependency and content importance. The calculation formula is as follows, where  $Authority(v)$  is the summed importance of the states (*including*  $u, w$ ) on which the current state ( $v$ ) depends.  $(w, v) \in E, DTR$  means that state  $v$  has a data dependency on state  $w$ , and  $(u, v) \in E, LTR$  means state  $v$  has a link dependency on state  $u$ .  $Importance(w)$  means the state importance of state  $w$ .  $\alpha$  is a constant more than 1 since we think the data dependency is more important than the link dependency.

$$Authority(v) = \sum_{(w,v) \in E, DTR} \alpha Importance(w) + \sum_{(u,v) \in E, LTR} Importance(u) + Con\_Importance(v)$$

Hub refers to the summed importance of states that depend on the state. It mainly consists of two parts: data dependency and link dependency. The calculation formula is as follows.

$$Hub(v) = \sum_{(w,v) \in E, DTR} \alpha Importance(w) + \sum_{(u,v) \in E, LTR} Importance(u)$$

So a state's importance can be calculated with the formula  $Importance(v) = Authority(v) + Hub(v)$ .

The algorithm for calculating state importance is described in Algorithm 2. The web application is abstracted into the form of an EFSM model. The initial value of state importance equals the Authority value, which equals to the content importance. The initial Hub value equals 0. In the iterative process, the value of current importance is compared with that of the previous generation. And when the difference is less than threshold  $\epsilon$ , the iteration ends and the page importance result is obtained. Note that  $Importance_t$  refers to an importance value vector of all the web pages in the  $t_{th}$  iteration.

---

#### Algorithm 2 Web State Importance Calculation

---

**Require:**  $EFSM = \langle state, trans \rangle$ , the threshold  $\epsilon$

**Ensure:** Pages' content importance value

```

1: for  $v$  in state do
2:    $Importance(v) = Authority(v), H(v) = 0$ 
3: end for
4: while  $\|Importance_t - Importance_{t-1}\| \leq \epsilon$  do
5:   for  $v$  in state do
6:      $Authority_t(v) = \sum_{(w,v) \in E, DTR} \alpha Importance_{t-1}(w) +$ 
7:        $\sum_{(u,v) \in E, LTR} Importance_{t-1}(u) + Con\_Importance_{t-1}(v)$ 
8:      $Hub_t(v) = \sum_{(w,v) \in E, DTR} \alpha Importance_{t-1}(w) +$ 
9:        $\sum_{(u,v) \in E, LTR} Importance_{t-1}(u)$ 
10:     $Importance_t(v) = Authority_t(v) + Hub_t(v)$ 
11:   end for
12:    $Importance_t = \frac{Importance_t}{\|Importance_t\|}$ 
13:    $t=t+1$ 
14: end while

```

---

### B. Test Case Generation

There are many test generation approaches for EFSMs. The search-based algorithm is the most commonly used. In this paper, we use GA to generate test cases based on state importance of EFSM model for web applications.

A test case is one transition path on EFSM which can be expressed as a sequence  $Ind = \langle t_1, t_2, \dots, t_i, t_j, \dots, t_n \rangle$ . A test case is an individual, and the initial population is randomly generated from the EFSM. Fitness function is to guide the evolution towards optimal solutions. It determines whether an individual can be selected into the next evolution. We design the fitness function from two aspects. One is maximizing the summed importance of the web states in an individual. The other is minimizing the repeated ratio of transitions in an individual. The fitness function can be formalized as follows.

$$fitness = \sum_{i=0}^n Importance(s_i) \cdot \frac{||UniTrans||}{||Ind||}$$

In this formula,  $Importance(s_i)$  refers to the importance of the web state  $s_i$ ,  $||UniTrans||$  is the number of not-repeated transitions and  $||Ind||$  is the number of transitions in this individual  $Ind$ .

Genetic operators include selection, crossover and mutation. Selection is based on the fitness of the individual, which means the individual of greater fitness has larger probability to be selected as the parent. Crossover operator in our method is single-point crossover. Mutation is applied to alter gene values in an individual.

## V. EXPERIMENT

In this section, we firstly validate the feasibility of our modelling method. Then we generate test cases from the EFSM model guided by the state importance to demonstrate the benefit of our method. To assess the effectiveness of our method, we conduct case studies on two web applications. The following research questions motivate our experiments.

RQ1: Is our session data based modelling method effective and feasible?

RQ2: Do the test cases which are generated guided by state importance from the EFSM, cover the important web states earlier?

### A. Experimental Subjects

In the experimental studies, we use an open source online Book Store[22] and the laboratory management system (DBLab) developed by our group as the subjects to evaluate the validity and effectiveness of our method. The lines of code(LOC) of DBLab and Book store is 10162 and 6304 respectively. Book store allows users to add and remove books from a cart, login and register new users and so on. DBLab is a laboratory management system that includes user registration and login, group meeting management and viewing, user account management, library management, file management, student forums, data sharing and other functional modules.

### B. EFSM Model Evaluation

The EFSM model is abstracted based on the Session data recorded on the server-side when the users access the web application. It may not be sufficient for modelling the web application. The integrity of the EFSM model is directly related to the Session data. In order to analyze the dependency

between the model and the Session data, and judge whether the EFSM model is integral, we perform the following experiment.

Taking the DBLab as an example, according to the record time of session data, the EFSM model is established with session data from one month, two months, three months, and four months respectively. Finally, the EFSM model of the web application is manually analyzed and checked to verify if there are missing states or transitions. The experimental results are as follows.

TABLE I  
STATISTICS OF DBLAB GENERATES EFSM BY MONTH

Item	records N. in Session	state N. on EFSM	Transition N. on EFSM
June	954	20	40
June to July	1583	24	54
June to August	1942	24	54
June to September	2814	26	61
Manually checked	-	28	65

It can be seen from the Table I, after modelling the web application manually, the EFSM model increased two states and four transitions. Through analyzing the extra states and transitions, we find that these functions of DBLab are not daily use type. Based on the above analysis, we can see that the proposed EFSM model construction method for web applications in this paper has a great dependence on Session data. However, if the users' session records for web applications reach certain level of saturation, the EFSM model established based on the session data can basically reflect all the functions of the web application and realize the complete description for the web application. So our session data based modelling method is effective and feasible(RQ1).

The detail information of EFSM models which are built through our method for two test subjects is, 1) the number of states and transitions of the DBLab's model is 28 and 65 respectively; 2) the number of states and transitions of the BookStore's model is 9 and 36 respectively.

The EFSM model of two test subjects can be found at <http://research.cs.buct.edu.cn/guo/files/EFSM-BS.pdf> and <http://research.cs.buct.edu.cn/guo/files/EFSM-DBLab.pdf>.

### C. Test Case Generation Method Evaluation

The purpose of the test case generation method proposed in this paper is to prioritize those nodes with high importance to be tested earlier. Because that such nodes have high error propagation capability and are more likely to bring negative affect in the web applications. We select the three most important nodes from the BookStore and DBLab as the high importance nodes according to the node importance value. The three high importance nodes in the BookStore are State1, State2, and State8. The three high importance nodes in the DBLab are State16, State18, and State22.

We use our method and the random method to generate test cases from the EFSMs of the two applications. The test suite size(population size) are set to 10 (BookStore) and 20 (DBLab). The goal of the test suite is to cover all the transitions. Our test case generation method and random method were executed 10 times repeatedly. The results is

that, in the BookStore, the number of test cases generated by our method is 4.6 to cover the three high importance nodes on average, and the number of test cases generated by random method is 8.2 on average. In DBLab, the number of test cases generated by our method is 5.2 to cover the three high importance nodes on average, and the number of test cases generated by random method is 15.8 on average. The importance of test suites generated by our method and random method for ten times is shown in Figure 3 ( $H$ : experimental number,  $V$ : number of generated test cases).

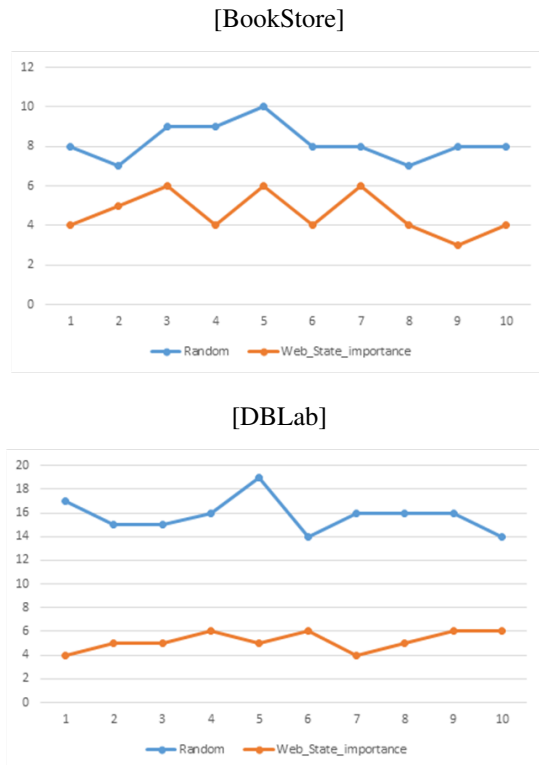


Fig. 3. The importance of test suites comparison

As the Figure 3 shows, under the premise of full transitions coverage, the test suite generated by our method has a higher importance value than the randomly generated test suite. At the same time, our test suite can cover important nodes when executing a small number of test cases. It can be seen that the method proposed in this paper can ensure that nodes with high importance to be tested firstly ( $RQ2$ ).

## VI. CONCLUSION AN FUTURE WORK

In this paper, we proposed an EFSM model constructing method for web applications based on session data, which can address the accurate description problem of web applications, and an algorithm to evaluate state importance of the EFSM model. In addition, we give a test case generation method based on the state importance of EFSM model, which can ensure the important web states can be tested first and then improve the efficiency of web application test.

The experimental results show that our methods can modeling web application well and the generated test case an cover the important web state earlier.

As the future work, firstly we will try to modify the EFSM model constructing method. Because that the method proposed in this paper need a sufficient amount of session data to detect the states and transitions. Secondly, we would like to design other algorithms for evaluating state importance by introducing other items, for example the number of variables.

## ACKNOWLEDGMENT

The work described in this paper is supported by the National Natural Science Foundation of China under Grant No.61702029, No.61672085 and No.61472025.

## REFERENCES

- [1] M. Hammad, M. L. Collard, and J. I. Maletic, "Measuring class importance in the context of design evolution," in *IEEE International Conference on Program Comprehension*, 2010, pp. 148–151.
- [2] "Selenium," <https://www.seleniumhq.org/>.
- [3] "Sahi," <http://sahipro.com/>.
- [4] "Watir," <http://watir.com/>.
- [5] Y. F. Li, P. K. Das, and D. L. Dowe, "Two decades of web application testing: a survey of recent advances," *Information Systems*, vol. 43, no. C, pp. 20–54, 2014.
- [6] M. Wang, J. Yuan, H. Miao, and G. Tan, "A static analysis approach for automatic generating test cases for web applications," in *International Conference on Computer Science and Software Engineering*, 2008, pp. 751–754.
- [7] G. Li, E. Andreasen, and I. Ghosh, "Symjs: automatic symbolic testing of javascript web applications," in *The ACM Sigsoft International Symposium*, 2014, pp. 449–459.
- [8] N. Alshahwan and M. Harman, "Automated web application testing using search based software engineering," in *Ieee/acm International Conference on Automated Software Engineering*, 2011, pp. 3–12.
- [9] F. Ricca and P. Tonella, "Analysis and testing of web applications," in *International Conference on Software Engineering*, 2001, pp. 25–34.
- [10] A. Marchetto, P. Tonella, and F. Ricca, "State-based testing of ajax web applications," in *International Symposium on Search Based Software Engineering*, 2009, pp. 3–12.
- [11] A. Marchetto and P. Tonella, "Using search-based algorithms for ajax event sequence generation during testing," *Empirical Software Engineering*, vol. 16, no. 1, pp. 103–140, 2011.
- [12] X. F. Qi, Z. Y. Wang, J. Q. Mao, and P. Wang, "Automated testing of web applications using combinatorial strategies," *Journal of Computer Science and Technology*, vol. 32, no. 1, pp. 199–210, 2017.
- [13] A. M. Torsel, "Automated test case generation for web applications from a domain specific model," in *Computer Software and Applications Conference Workshops*, 2011, pp. 137–142.
- [14] P. Bansal and S. Sabharwal, "A model based approach to test case generation for testing the navigation behavior of dynamic web applications," in *Sixth International Conference on Contemporary Computing*, 2013, pp. 213–218.
- [15] A. S. Kalaji, R. M. Hierons, and S. Swift, *An integrated search-based approach for automatic testing from extended finite state machine (EFSM) models*. Butterworth-Heinemann, 2011.
- [16] A. S. Kalaji and Hierons, "Generating feasible transition paths for testing from an extended finite state machine (efsm)," in *International Conference on Software Testing, Verification, and Validation Workshops*, 2010, pp. 230–239.
- [17] "Crawljax," <http://crawljax.com>.
- [18] A. Mesbah and A. V. Deursen, "Migrating multi-page web applications to single-page ajax interfaces," in *European Conference on Software Maintenance and Reengineering*, 2007, pp. 181–190.
- [19] L. D. Catledge and J. E. Pitkow, "Characterizing browsing strategies in the world-wide web," in *International World Wide Web Conference*, 1995, p. 10651073.
- [20] J. Guo, C. Gao, N. Xu, G. Lu, and H. Han, "Analyzing query trails and satisfaction based on browsing behaviors," in *Web Information System and Application Conference*, 2014, pp. 107–112.
- [21] L. L. D. Chen, X. L. Ren, Q. M. Zhang, Y. C. Zhang, and T. Zhou, "Vital nodes identification in complex networks," *Physics Reports*, vol. 650, pp. 1–63, 2016.
- [22] "Book store," <http://gotocode.com/>.



# Parallel Property Checking with Symbolic Execution

Junye Wen, Guowei Yang  
Department of Computer Science  
Texas State University, San Marcos, TX  
{j\_w236, gyang}@txstate.edu

**Abstract**—Systematically checking code against functional correctness properties is costly, especially for complex code annotated with rich behavioral properties. This paper introduces a novel approach to checking properties in parallel using symbolic execution. Our approach partitions a check for the whole set of properties into multiple simpler sub-checks—each sub-check focusing on a single property, so that different properties are checked in parallel among multiple workers. Furthermore, each sub-check is *guided* by the checked property to avoid exploring irrelevant paths and is *prioritized* based on distances towards the checked property to provide early feedback. We implement our approach in Symbolic PathFinder, and experiments on systematically checking assertions in Java programs show the effectiveness of our approach.

## I. INTRODUCTION

Researchers have long recognized the value of annotating functional correctness properties of code using assertions [7] or executable contracts, such as those supported by the Java Modeling Language [15] or Eiffel [17]. However, developers are often reluctant to use them largely due to the high computational cost of running automated analyses to check them.

Symbolic execution [13], [14] is a powerful program analysis technique that has a number of useful applications and has been widely used as a systematic technique for bug finding [10], [20], [22], [28], but symbolic execution is computationally expensive due to the large number of paths to explore as well as the high cost of underlying constraint solving. Scaling symbolic execution remains challenging for complex programs in practice. When programs are annotated with functional correctness properties, symbolic execution can be naturally applied to automatically check program behaviors against the annotated properties to check their validity. However, the scalability issue is even exacerbated as the annotated properties often introduce extra paths and extra constraints. This paper is focused on reducing the computational cost of symbolic execution in checking properties.

A lot of advances in symbolic execution have been made during the last decade. Specifically, parallel analysis [5], [23]–[25] allows multiple workers to explore largely disjoint sets of program behaviors in parallel, and has shown particular promise in addressing the scalability issue of symbolic execution. However, to the best of our knowledge, none of

the approaches consider the characteristics of the annotated properties in their parallelization strategies.

This paper introduces a novel approach to parallel property checking using symbolic execution. Our key insight is that properties are normally written without side effects, and thus checking of each property is independent of checking of other properties. Our approach partitions a check for the whole set of properties into multiple simpler sub-checks—each focusing on one single property, so that different properties are checked in parallel among multiple workers. Furthermore, each sub-check is *guided* by the checked property to avoid exploring irrelevant paths and is *prioritized* based on distances towards the checked property to provide earlier feedback, allowing users to fix bugs in code or refine properties earlier. Specifically, during state space exploration we statically check whether the checked property is reachable or not along the current path, and prune the search when the checked property cannot be reached. Moreover, we prioritize the state space exploration so that the state whose corresponding location has the shortest distance towards the checked property is explored first, i.e., the shortest path to the checked property gets explored first. Therefore, the prioritized state space exploration can provide earlier feedback on the checked property. Note that the chance of pruning irrelevant state space is much higher in each sub-check than in the original check, since in a sub-check the program under analysis has only one property at a particular location in the program, while the program under analysis in the original check has multiple properties scattered in different locations in the program.

We implement our approach in Symbolic PathFinder [18]. To evaluate the efficacy of our approach we apply it in the context of symbolic execution for checking Java programs annotated with assertions. We conduct experiments based on five subjects: three Java programs with manually written assertions and two Java programs with synthesized assertions. Experimental results show that our approach for parallel property checking detects more assertion violations and reduces the overall analysis time compared with regular non-parallel property checking. For one subject, while regular property checking timed out after executing for two hours, our parallel property checking technique completed within four seconds. In addition, for most sub-checks, our guided check prunes state space and reduces the time cost, and our prioritized check provides earlier feedback compared to regular check.

## II. MOTIVATING EXAMPLE

We use an example to illustrate how our approach leverages the annotated properties to improve the scalability of symbolic execution for property checking. Consider the source code of median shown in Figure 1. It computes the middle value of its three integer inputs; this method is adapted from previous work [12], and five assertions are manually added to check the correctness of the program. For example, the user asserts  $x \leq y \ \&\& \ y \leq z$  at line 4, indicating that  $y$  should be the middle value of the three inputs; otherwise, an assertion violation is captured.

```

1 int median(int x, int y, int z) {
2   if (y < z) {
3     if (x < y){
4       assert x <= y && y <= z; //#1
5       return y;}
6     else if (x < z){
7       assert y <= x && x <= z; //#2
8       return x;}
9   }
10  else {
11    if (x > y){
12      assert z <= y && y <= x; //#3
13      return y;}
14    else if (x > z){
15      assert z <= x && x <= y; //#4
16      return x;}
17  }
18
19  assert (x<=z && z<=y) || (y<=z && z<=x); //#5
20  return z;
21}

```

Fig. 1. Method to compute the middle value of three input numbers and its annotated assertions.

The workload of checking five assertions in this program is conducted by five workers running in parallel, such that each worker checks one single assertion. For example, the worker responsible for checking assertion #1 analyzes a program version, where the code together with the target assertion #1 remain unchanged, while all the other four assertions are removed.

In addition, each sub-check is further optimized using *guided* and *prioritized* state space exploration based on the checked assertion. For checking assertion #1, the sub-check is guided by assertion #1, avoiding exploring the irrelevant parts of the program. Therefore, instead of exploring all the six possible paths in the program, the guided check only explores one path, that satisfies path condition  $y < z$  and  $x < y$  and reaches the checked assertion. It results into up to 5/6 reduction in terms of the number of paths to be explored. If multiple paths can reach the checked assertion, we use shortest distance based heuristics to prioritize the search so that the assertion can be checked as early as possible and a feedback, i.e., whether the assertion is violated or not, can be returned to the user as early as possible.

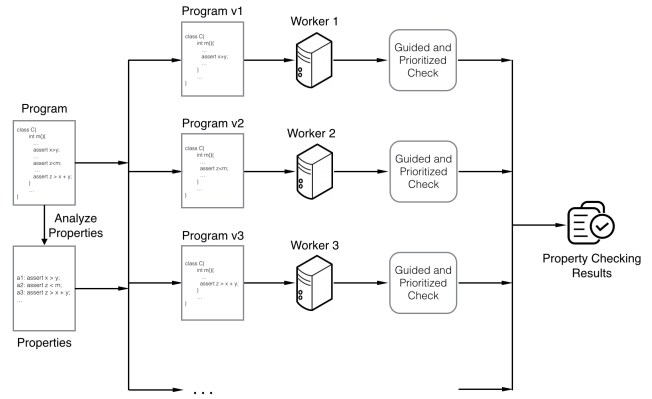


Fig. 2. An overview of the approach

## III. APPROACH

Our approach is focused on how to optimally utilize the computing resources available to check properties, specifically in a parallel setting where the checking can be conducted among several workers. Our key insight is that properties are normally written without side effects, and thus checking of each property is independent of checking of other properties. The result from checking all properties in one run should be the same as that from checking properties in multiple multiple runs in parallel. This enables us to partition a check for the whole set of properties into multiple simpler checks—each focusing on one single property, so that different properties are checked in parallel among multiple workers. Therefore, the original check is converted into multiple simpler sub-checks in parallel for better scalability.

Figure 2 shows an overview of the approach. Consider a program  $P$  with multiple properties  $PT = \{PT_1, PT_2, \dots, PT_m\}$  to check. Our approach first statically analyzes the program to find all the  $m$  properties to check, and accordingly prepare  $m$  program versions  $V = \{v_1, v_2, \dots, v_m\}$  where each version contains only one property that does not appear in other versions. These versions are then checked by  $m$  workers, each worker focusing on one version and altogether checking all the properties in parallel. Each worker works on its own program version with one single property using property guided and prioritized check. Finally, the property checking results from these workers are delivered to the user.

The partition of properties not only simplifies the program to be checked due to the removal of other properties, but also allows further optimization of each sub-check. Since each sub-check focuses on one single property, it is more likely to have paths that do not reach the checked property compared with the original check that focuses on multiple properties. Leveraging this observation, each sub-check, i.e., a symbolic execution run for checking one single property, is *guided* by the checked property such that it only explores the program state space that is relevant to the checked property. If the current path cannot reach the checked property, symbolic execution does not continue along the path and backtracks. By effectively

---

**Algorithm 1** Procedure *check* for checking a property

---

**Input:** Program  $P$ , property  $PT$ , search depth bound  $DepthBound$ **Output:** A set of property violations detected during symbolic execution  $VS$ 

```
1: Queue  $tq \leftarrow$  enabled transitions at current state  $s$ 
2: while  $\neg tq.isEmpty()$  do
3:    $t \leftarrow tq.remove()$ 
4:    $nt \leftarrow GetCFGNode(P, t)$ 
5:    $na \leftarrow GetCFGNode(P, PT)$ 
6:   if  $\neg IsCFGPath(nt, na)$  then
7:     continue
8:   else
9:      $s' \leftarrow execute(s, t)$ 
10:     $pc \leftarrow$  current path condition
11:     $depth \leftarrow depth + 1$ 
12:    if  $pc$  is not satisfiable then
13:      continue
14:    end if
15:    if  $isPropertyViolated(s')$  then
16:       $VS.add(violation(s'))$ 
17:    end if
18:    if  $depth == DepthBound$  then
19:      continue
20:    else
21:       $check(s')$ 
22:    end if
23:  end if
24: end while
```

---

*pruning* paths that cannot reach the checked assertions, our approach avoids the cost of exploring irrelevant paths.

Algorithm 1 shows the procedure *check* for performing property checking for a program with one single property. Given as input a program, a property to check, and a bound on the search depth, the procedure check the conformance of the program behaviors with the checked property, and return all property violations in the program. It starts with the initial state for  $s$ , 0 for  $depth$ , and an empty set for  $VS$ . It finds all enabled transitions at the current state (Line 1) to systematically search the state space. Lines 4 – 5 locate the Control Flow Graph (CFG) nodes for the enabled transition, and the checked property, respectively. Both the enabled transition and the checked property could correspond to multiple CFG nodes, we simplify it here assuming that each corresponds to one CFG node. It checks whether the current transition reaches the checked property, and if not prune the search (Lines 6 – 7); otherwise, it executes the transition to get to the next state, and update the  $pc$  and  $depth$  (Lines 9 – 11). If  $pc$  is unsatisfiable (i.e., the corresponding path is infeasible), the checked property is violated, or search depth reaches the bound, it backtracks to explore other un-explored enabled transitions (Lines 12 – 20); otherwise, it recursively explores the states rooted at the new state  $s'$  (Line 22).

In addition, each sub-check is *prioritized* to provide early feedback to the user. In the context of property checking, usually one property violation is enough for investigating the violation, and there is no need to find all property violations. Our insight is that the earlier a property is checked, the earlier the user could start the investigation and fix the potential problem either by modifying the code or by refining the checked property. As there is no precise way to predict the feasibility of paths and how long each path would take. We use a heuristics to prioritize the check. Specifically, we calculate

the distances from the current point towards the checked property along all potential paths, and choose the shortest path to explore first [16].

To prioritize the search, at each branching point, we sort the list of enabled transitions based on an estimated distance to the checked property in a CFG. For each enabled transition  $t_i$ , we compute an estimated distance to the checked property. The enabled transitions queue ( $tq$  in Algorithm 1) is sorted in ascending order based on the estimated distances of the transitions before the queue is explored. The enabled transition with the shortest distance is explored first. The distance is a lower bound on the number of CFG branches from a node  $n_i$  (corresponding to  $t_i$ ) to node  $n_j$ , that is corresponding to the checked property:

$$\forall n_i . n_j : d_i := \min ( \text{branches} ( n_i, n_j ) )$$

In our approach, we use the all-pairs shortest path algorithm to compute the lower bound on the number of CFG branches. The complexity is cubic in the number of branches in the CFG. We note that metrics other than number of branches can also be used as a distance estimate, for example, the number of bytecodes.

#### IV. EVALUATION

We empirically evaluate the effectiveness of our approach for parallel property checking. Our evaluation addresses the following research questions:

- **RQ1:** How does the efficiency of our parallel property checking compare with regular property checking?
- **RQ2:** How does the cost of our guided check compare with regular check?
- **RQ3:** How does our prioritized check compare with regular check in terms of providing feedback to the user?

##### A. Artifacts

In our evaluation, we use five subjects including `median`, `testLoop`, `trityp`, `WBS`, and `TCAS`. All of them have been used before for evaluating symbolic execution techniques [19], [25], [27], [28].

The first subject `median` is shown in Figure 1. The second subject `testLoop` is used to investigate how our approach can help deal with loops, as they pose particular challenges to symbolic execution and handling them efficiently is an active area of research. The third subject is a Java version of the classic triangle classification program by Ammann and Offutt. The classification logic of the `trityp` program seems deceptively simple, but are non-trivial to reason about. We consider the correct version of assertions developed for `trityp` in previous work [27].

For the two subject programs `WBS` and `TCAS`, we use mechanically synthesized assertions. To synthesize assertions for our experiments, we use the *Daikon* tool for invariant discovery [9]. Specifically, we apply *Daikon* on each subject to

discover invariants and transform them to assertions. Daikon requires a test suite to execute the program under analysis and detect its likely invariants. TCAS had a test suite available in the Software Infrastructure Repository [1], so we used this test suite which contains 1608 tests. For WBS, we wrote a random test generator to create a test suite with 1000 tests. We selected all the eight Daikon invariants for TCAS and randomly selected 25 out of 35 invariants for synthesizing assertions.

### B. Experiment Setup

In this work, we use Symbolic PathFinder (SPF) [18], an open-source tool for symbolic execution of Java programs built on top of the Java PathFinder (JPF) model checker [26] to perform symbolic execution. We implemented guided and prioritized check in SPF as a customized listeners, and we built customized control flow graphs to compute estimated distances and reachability information to guide and prioritize property checking. We also conduct experiments using regular symbolic execution as implemented in SPF for comparison. Choco constraint solver [2] is used for solving path conditions involved in symbolic execution.

To evaluate RQ1 and RQ2, symbolic execution is configured to detect all assertion violations; while to evaluate RQ3, symbolic execution is configured to stop when it detects the first assertion violation, to check whether our prioritized check could provide earlier feedback than regular check.

We assume that there are enough workers available for performing the tasks in parallel. In practice, resources could be limited, and we need design strategies for statically grouping work before dispatching or for dynamically stealing work among workers, which is left for future work.

We perform the experiments on the Lonestar cluster at the Texas Advanced Computing Center (TACC) [3]. TACC provides powerful computation nodes with reliable and fast connectivity. The programs for each worker node are executed on independent processors without memory sharing.

### C. Results and Analysis

In this section, we present the results of our experiments, and analyze the results with respect to our three research questions.

RQ1: How does the efficiency of our parallel property checking compare with regular property checking?

Table I shows the experimental results for checking all assertions in the subject programs using our parallel property checking approach and using regular non-parallel property checking approach. It shows the number of detected assertion violations, and three types of checking cost, i.e., time, number of states explored, and the maximum memory cost, for each approach. Since in the parallel property checking sub-checks are analyzed in parallel among multiple workers, the table shows cost ranges of values across all sub-checks, and it also shows the overall time cost for the parallel property checking; while for regular symbolic execution the cost is collected by running regular symbolic execution on the original program annotated with all assertions. We note that 0 in time cost means

less than 1 second. *TO* indicates that the corresponding check timed out.

We find that there are no assertion violations for median and `trityp`, while for the other three subjects, the parallel approach detects more assertion violations than regular approach. This is because some expensive assertion checking happens only in the parallel property checking. Since Symbolic PathFinder backtracks as soon as it detects an assertion violation, the inputs reaching deep assertions may be reduced due to violations of the shallow assertions along the same path, and thus may not detect the possible violations of the deep assertions in regular property checking approach.

Moreover, for all subjects except for WBS, the parallel approach is more efficient than regular approach in property checking. Specifically, it achieves almost 3X speedup for TCAS. For `testLoop`, while regular symbolic execution timed out after executing for *two* hours, the parallel property checking completed within 31 seconds. Without surprise, most sub-checks explored only part of the state space. We also note however for WBS our approach took more time, and explored more states, which is because of the cost for detecting the 130 more violations.

In addition, we find that although the parallel approach takes almost the same memory cost as regular symbolic execution for most runs, it takes more memory for some sub-checks for WBS and TCAS; we note however that the maximum memory reported by SPF may vary a lot due to the underlying garbage collection, and thus this comparison is not very meaningful.

RQ2: How does the cost of our guided check compare with regular check?

Table II reports the experimental results for each sub-check using guided check and prioritized check compared to using regular check, i.e., regular symbolic execution. As we explained before, the comparison in memory cost is not very meaningful, thus here we only report the cost in terms of time and explored states.

To evaluate RQ2, symbolic execution is configured to check for all assertion violations. We observe that for 44 out of 50 versions, our guided check explored fewer states than regular check, since guided check prunes state space exploration when the checked property is not reachable. For example, for `v1` of `testLoop`, guided check explored 103 states while regular check explored 154 states, which is about 1/3 reduction. Accordingly, the guided check took less time than regular check for most of these cases. For example, for `v1` of `trityp`, guided check took 18 seconds while regular check took 22 seconds. However, we note that for some cases, although there was a reduction in states, the time cost of guided check was even higher than regular check due to the overhead of static analysis involved in guided check.

RQ3: How does our prioritized check compare with regular check in terms of providing feedback to the user?

To evaluate RQ3, run symbolic execution is configured to stop when it finds the first assertion violation. From Table II, we observe that for 40 out of 50 versions, prioritized check

TABLE I  
RESULTS OF PARALLEL AND REGULAR PROPERTY CHECKING.

Subject	Parallel Property Checking					Regular Property Checking			
	Detected Violations	Total Time (s)	Time (s)	# of States	Memory (MB)	Detected Violations	Time (s)	# of States	Memory (MB)
median (5 assertions)	0	2	0-2	5-13	965-965	0	2	13	965
testLoop (2 assertions)	2	31	0-30	103-180	965-965	-	TO	-	-
trityp (10 assertions)	0	49	18-48	33-49	965-965	0	103	81	965
WBS (8 assertions)	222	7	0-7	359-671	965-1178	92	2	533	965
TCAS (25 assertions)	251	680	27-679	679-935	965-1685	195	2025	2047	965

TABLE II  
PROPERTY CHECKING USING GUIDED AND PRIORITIZED CHECK AND REGULAR CHECK.

Subject	Ver	Check all violations				Check first violation			
		Guided Check		Regular Check		Prioritized Check		Regular Check	
		Time	States	Time	States	Time	States	Time	States
median	v1	0	5	0	11	0	5	0	11
	v2	0	7	1	11	0	7	1	11
	v3	0	5	1	11	0	5	1	11
	v4	1	5	0	11	0	5	1	11
	v5	2	13	2	13	1	13	1	13
testLoop	v1	0	103	0	154	0	103	0	154
	v2	30	727	TO	TO	0	180	TO	TO
trityp	v1	18	33	22	40	18	33	22	40
	v2	18	35	18	36	18	35	21	36
	v3	33	49	40	57	34	49	35	57
	v4	29	39	29	39	29	39	32	39
	v5	48	35	53	36	45	35	55	36
	v6	28	37	30	42	28	37	29	42
	v7	26	37	27	40	26	37	29	40
	v8	19	35	20	36	21	35	21	36
	v9	22	39	19	42	22	39	20	39
	v10	20	39	23	39	23	39	21	39
WBS	v1	0	451	0	455	0	163	0	255
	v2	0	359	0	359	0	222	0	341
	v3	0	527	1	530	0	527	0	530
	v4	0	623	1	623	0	9	0	9
	v5	0	535	1	535	0	535	0	561
	v6	7	671	11	680	0	9	0	9
	v7	0	487	1	500	0	48	0	117
	v8	0	527	0	530	0	368	0	421
TCAS	v1	219	727	250	760	289	727	265	702
	v2	27	727	27	760	37	727	43	702
	v3	34	687	33	702	37	687	33	702
	v4	149	687	156	702	125	687	137	702
	v5	30	679	41	754	27	679	34	754
	v6	35	679	40	754	33	679	37	754
	v7	31	679	35	679	33	679	34	679
	v8	28	679	33	679	33	679	34	679
	v9	241	695	275	722	240	695	257	722
	v10	251	695	270	722	222	695	318	722
	v11	32	695	36	722	1	33	1	38
	v12	31	695	33	722	1	33	1	38
	v13	201	695	226	727	238	695	241	727
	v14	130	695	132	727	134	695	146	727
	v15	28	695	34	727	11	229	13	270
	v16	28	695	35	727	9	229	12	270
	v17	679	743	644	745	557	743	568	745
	v18	31	743	32	745	31	743	32	745
	v19	36	935	34	950	15	370	26	439
	v20	33	935	36	950	8	247	14	323
	v21	30	719	30	874	29	678	29	678
	v22	34	719	35	874	11	167	18	214
	v23	33	815	35	827	0	20	0	33
	v24	28	815	39	827	10	191	20	331
	v25	34	815	35	827	9	211	19	231

explored fewer states than regular check, and for 8 versions, both techniques explored the same number of states. For instance, for v24 of TCAS, prioritized check explored 191 states, while regular check explored 331 states. However, for the other 2 versions (i.e., v1 and v2 of TCAS, prioritized check explored slightly more states than regular check. This is not surprising as the shortest path selected by our heuristics is based on number of branches in CFG, and may result in more states to explore in symbolic execution. Similar to previous experiments, prioritized check usually took less time when it explored fewer states, as the time cost is correlated with states exploration. For example, for v10 of TCAS, prioritized

check took 222 seconds, while regular check took 318 seconds, which is about 1.5X speedup. Moreover, for v2 of testLoop, prioritized check completes in less than one second; in contrast, regular check timed out after running for two hours. Only for few versions, prioritized check took slightly more time than regular check.

## V. RELATED WORK

Several research projects have proposed techniques for parallel symbolic execution [5], [23], [25]. Static partitioning [25] leverages an initial shallow symbolic execution run to minimize the communication overhead during parallel symbolic execution. It creates pre-conditions using conjunctions of clauses on path conditions encountered during the shallow run, and restricts symbolic execution by each worker to explore only paths that satisfy the pre-condition. ParSym [23] parallelizes symbolic execution dynamically by taking each path exploration as one unit of work and using a central server to distribute work between parallel workers. Cloud9 [5] utilizes load balancing that initially assigns the whole program analysis to a worker, and whenever an idle worker becomes available, the load balancer instructs the busy worker to suspend exploration and breaks off some of its unexplored subtree to send to the idle worker to balance the work load. While these techniques use parallelization to speed up symbolic execution in general and check the whole bounded state space, our work is focused on checking side-effect-free properties and ignores path exploration that is irrelevant to the checked properties.

Much work has been done for guiding symbolic execution [16], [19], [21]. Directed symbolic execution [19] uses a def-use analysis to compute change affected locations and then uses this information to guide symbolic execution to explore only program paths that are affected by the changes. Santelices and Harrold [21] use control and data dependencies to symbolically execute groups of paths, rather than individual paths. Ma et al. [16] propose a call chain backward search heuristic to find a feasible path to the target location. Our work leverages reachability of properties to guide symbolic execution to only explore paths relevant to the checked properties.

Some recent projects [11], [27], [29] have explored more efficient checking of properties. Guo et al. [11] introduce assertion guided symbolic execution for eliminating redundant executions in multi-threaded programs to reduce the overall computational cost. An execution is considered redundant when it shares the same reason why it cannot reach the bad state with previous executions, and thus can be eliminated

for the purpose of checking assertions. While it focuses on eliminating redundant executions for multi-threaded programs, our guided check focuses on eliminating irrelevant executions for single-threaded programs. iProperty [27] computes differences between assertions of related programs in a manner that facilitates more efficient incremental checking of conformance of programs to properties. Our approach is orthogonal and can use iProperty to compute differences between assertion versions when the checked assertion is changed, thus speeding up the assertion checking carried out by each worker. iDiscovery [29] uses assertion separation to focus symbolic execution on checking one assertion at a time, and violation restriction to generate at most one violation of each assertion. While our work shares some insight with assertion separation on checking assertions separately, the guided and prioritized check in our work has potential to more efficiently check each assertion.

This work is different from property-based slicing and property-aware testing and verification [4], [6], [8], since here we simply check the reachability of properties and apply this for guiding symbolic execution rather than other testing or verification techniques.

We have presented the high-level ideas of this work in Java PathFinder workshop 2015 to get early feedback, with no formal proceedings for the paper. In this paper we have developed the ideas further, and we have also provided more evaluation of the work.

## VI. CONCLUSIONS AND FUTURE WORK

This paper introduced a novel approach for partitioning the problem of property checking using symbolic execution into simpler sub-checks where each check is focused on checking one single property. All sub-checks are performed by multiple workers in parallel for better scalability. The parallelized property checking enabled us to further optimize each sub-check by pruning irrelevant paths regarding the checked property. Moreover, check is prioritized to explore shorter paths towards properties so that earlier feedback on the checked property can be provided to the user. Experiments using five subject programs with assertions that are manually written as well as automatically synthesized, showed that our approach for parallel property checking reduced the overall analysis time compared with regular non-parallel property checking; and in sub-checks which focus on checking one single assertion, our guided check pruned state space exploration and thus reduced the time cost, and our prioritized check provided earlier feedback compared to regular check.

As for future work, we plan to conduct more extensive evaluation of our approach using more complex subjects, such as open source programs. We would also like to investigate how to parallelize property checking when not enough resources are available, for example, the number of available workers is fewer than the number of checked properties in the program.

## ACKNOWLEDGMENTS

This work is partially supported by the National Science Foundation under Grant No. CCF-1464123.

## REFERENCES

- [1] SIR Repository. <http://sir.unl.edu>.
- [2] Choco solver. <http://www.emn.fr/z-info/choco-solver>.
- [3] Lonestar cluster. <https://www.tacc.utexas.edu/systems/lonestar>.
- [4] R. H. Bordini, M. Fisher, M. Wooldridge, and W. Visser. Property-based slicing for agent verification. *J. Log. and Comput.*, 19(6):1385–1425, Dec. 2009.
- [5] S. Bucur, V. Ureche, C. Zamfir, and G. Candea. Parallel symbolic execution for automated real-world software testing. In *EuroSys*, pages 183–198, 2011.
- [6] G. Canfora, A. Cimitile, and A. D. Lucia. Conditioned program slicing. *Information & Software Technology*, pages 595–607, 1998.
- [7] L. A. Clarke and D. S. Rosenblum. A historical perspective on runtime assertion checking in software development. *SIGSOFT Software Engineering Notes*, 2006.
- [8] J. C. Corbett, M. B. Dwyer, J. Hatcliff, and Robby. Bandera: a source-level interface for model checking java programs. In *ICSE*, pages 762–765, 2000.
- [9] M. D. Ernst. *Dynamically Discovering Likely Program Invariants*. PhD thesis, University of Washington Department of Computer Science and Engineering, Seattle, Washington, Aug. 2000.
- [10] P. Godefroid, S. K. Lahiri, and C. Rubio-González. Statically validating must summaries for incremental compositional dynamic test generation. In *SAS*, pages 112–128, 2011.
- [11] S. Guo, M. Kusano, C. Wang, Z. Yang, and A. Gupta. Assertion guided symbolic execution of multithreaded programs. In *ESEC/FSE*, pages 854–865, 2015.
- [12] J. A. Jones. *Semi-Automatic Fault Localization*. PhD thesis, Georgia Institute of Technology, Atlanta, GA, 2008.
- [13] S. Khurshid, C. S. Păsăreanu, and W. Visser. Generalized symbolic execution for model checking and testing. In *TACAS*, pages 553–568, 2003.
- [14] J. C. King. Symbolic execution and program testing. *Communications of the ACM*, pages 385–394, 1976.
- [15] G. T. Leavens, Y. Cheon, C. Clifton, C. Ruby, and D. R. Cok. How the design of jml accommodates both runtime assertion checking and formal verification. *Sci. Comput. Program.*, pages 185–208, 2005.
- [16] K.-K. Ma, K. Y. Phang, J. S. Foster, and M. Hicks. Directed symbolic execution. In *SAS*, pages 95–111, 2011.
- [17] B. Meyer, J.-M. Nerson, and M. Matsuo. Eiffel: Object-oriented design for software engineering. In *ESEC*, pages 221–229, 1987.
- [18] C. S. Păsăreanu, W. Visser, D. Bushnell, J. Geldenhuys, P. Mehltitz, and N. Rungta. Symbolic Pathfinder: integrating symbolic execution with model checking for Java bytecode analysis. *Automated Software Engineering*, pages 391–425, 2013.
- [19] S. Person, G. Yang, N. Rungta, and S. Khurshid. Directed incremental symbolic execution. In *PLDI*, pages 504–515, 2011.
- [20] C. S. Păsăreanu, P. C. Mehltitz, D. H. Bushnell, K. Gundy-Burlet, M. Lowry, S. Person, and M. Pape. Combining unit-level symbolic execution and system-level concrete execution for testing NASA software. In *ISSTA*, pages 15–26, 2008.
- [21] R. Santelices and M. J. Harrold. Exploiting program dependencies for scalable multiple-path symbolic execution. In *ISSTA*, pages 195–206, 2010.
- [22] K. Sen and G. Agha. Cute and jcute: Concolic unit testing and explicit path model-checking tools. In *CAV*, pages 419–423, 2006.
- [23] J. H. Siddiqui and S. Khurshid. ParSym: Parallel symbolic execution. In *ICSE*, pages V1–405 – V1–409, 2010.
- [24] J. H. Siddiqui and S. Khurshid. Scaling symbolic execution using ranged analysis. In *OOPSLA*, pages 523–536, 2012.
- [25] M. Staats and C. Păsăreanu. Parallel symbolic execution for structural test generation. In *ISSTA*, pages 183–194, 2010.
- [26] W. Visser, K. Havelund, G. Brat, S. Park, and F. Lerda. Model checking programs. *Automated Software Engg.*, pages 203–232, 2003.
- [27] G. Yang, S. Khurshid, S. Person, and N. Rungta. Property differencing for incremental checking. In *ICSE*, pages 1059–1070, 2014.
- [28] G. Yang, C. S. Păsăreanu, and S. Khurshid. Memoized symbolic execution. In *ISSTA*, pages 144–154, 2012.
- [29] L. Zhang, G. Yang, N. Rungta, S. Person, and S. Khurshid. Feedback-driven dynamic invariant discovery. In *ISSTA*, pages 362–372, 2014.

# Software Process Improvement Programs: What happens after official appraisal?

Regina Albuquerque

Graduate Program in Informatics –  
Pontifícia Universidade Católica do  
Paraná (PUCPR), Curitiba – Paraná  
– Brazil

[Regina.fabia@pucpr.br](mailto:Regina.fabia@pucpr.br)

Andreia Malucelli

Graduate Program in Informatics –  
Pontifícia Universidade Católica do  
Paraná (PUCPR), Curitiba – Paraná  
– Brazil

[malu@ppgia.pucpr.br](mailto:malu@ppgia.pucpr.br)

Sheila Reinehr

Graduate in Informatics – Pontifícia  
Universidade Católica do Paraná  
(PUCPR), Curitiba – Paraná – Brazil

[sheila.reinehr@pucpr.br](mailto:sheila.reinehr@pucpr.br)

**Abstract—** *Studies of critical success factors in the implementation of programs of software process improvement have been conducted on a large scale in recent decades. However, these studies only focus on the implementation and do not consider the factors that influence the continuity or abandonment of these programs after official appraisal. Therefore, the objective of this study is to investigate the critical factors involved in the maintenance or abandonment of these programs. For this purpose, an exploratory survey was conducted, with the participation of consultants and appraisers of software process improvement reference models. The result was the proposal of a theoretical framework of critical success factors to maintain the use of processes of reference models, composed of the following categories: human factors, project factors, consultancy, organizational factors and technical factors involved in the process.*

**Keywords:** *Software Process Improvement, continuity of SPI, abandonment of SPI.*

## I. INTRODUCTION

Since NATO meeting that was held in 1968, where the term *software engineering* was born, the field has evolved significantly in terms of methods, tools and programming languages [1]. Several paths have been taken by both practitioners and researchers during this evolution. One of them concerns the belief that the quality of the software development process directly influences the quality of the resulting software product [2]. Based on this assumption several models to support software process improvement (SPI) were developed [3], especially those based on the concepts of maturity and capability, such as CMMI-DEV (*Capability Maturity Model Integration for Development*) [4]. Nowadays, there are also reference models that are highly recognized in their home country such as MR-MPS-SW (*Brazilian Software Process Improvement Model*) [5] and MoProSoft (the *Mexican Model for Software Process*) [6].

In 2003 the MPS.BR (*Brazilian Software Process Improvement Program*) was created by SOFTEX (*Association for Promoting the Excellence of Brazilian Software*) to support small and medium software organizations in the journey of SPI. The program is based on five core components [5]: Reference Model for Software (MR-MPS-SW), Reference Model for Services (MR-MPS-

SV), Reference Model for Human Resources Management (MR-MPS-RH), Assessment Method (MA-MPS) e Business Model (MN-MPS).

The reference model called MR-MPS-SW is focused on software processes and is equivalent to CMMI-DEV [15]. Both define maturity levels that establishes process evolution patterns, characterizing stages of improvement in the implementation of processes in an organization.

To facilitate the adoption of the model by the Brazilian software companies, MR-MPS-SW has seven maturity levels (starting at level G, the lowest one, and progressing up to level A): A (Optimizing) corresponds to CMMI-DEV Level 5; B (Quantitatively Managed) corresponds to CMMI Level 4; C (Defined), D (broadly defined) and E (partially defined) correspond to CMMI level 3; F (Managed) and G (Partially Managed) are equivalent to CMMI-DEV level 2.

The cooperative business model is a hiring modality that stands out because it has the financial support of the Brazilian federal government (it has already reached the mark of 10 million), allowing companies to be organized in groups, sharing costs of training and implementation.

Organizations adopting such reference models are submitted to an official assessment to attest their process maturity level. In the case of CMMI-DEV, the appraisal model is called SCAMPI [4]. In the case of MR-MPS-SW, the appraisal method is called MA-MPS. Based on the results of the appraisal, organizations can plan changes to their process aiming at reaching better business results [7]. In both cases, an appraisal expires after 3 years.

In seeking maturity in their development processes, organizations often modify their management style. This change can impact positively on the results of software projects. For example, according to [8][9] some benefits can be observed such as the improvement on the predictability of cost and schedule estimates.

Although some very promising results of implementing such models have been described, there are some challenges on the way. As stated by [10], “*understanding how to implement software process improvement (SPI) successfully is arguably the most challenging issue facing the SPI field*”. The finding of this difficulty was the motivation to carry out a significant number of researches on critical success factors for SPI initiatives, as seen in the systematic literature reviews (SLR) [11] [12].

Although there are some studies [13] [14] aiming to understand what influences the continuity of SPI programs after the official appraisal, there are several open questions such as: why do some organizations start at the lower maturity levels and do not evolve to a higher one? Why several organizations do not renew their appraisals when they expire? What really happen to the assessed processes after the official assessment: are they still up and running? Are they abandoned? Are they still been improved? There is, therefore, a need for more research that explores different cultural and organizational contexts, as well as the publication of experience reports from the software industry on the continuous improvement of software processes.

In this context, the objective of this study was to understand what makes it difficult and easier to maintain these SPI programs in software companies evaluated. To answer this question, a survey was conducted with SPI specialists. The results indicate that the continued use of SPI programs is related to human factors, the SPI project itself, organizational factors, consultancy and processes.

The study is organized into five sections. Section II presents the theoretical basis; section III the research method; section IV the results and the final considerations in section V.

## II. BACKGROUND

As discussed in the introduction there is a dearth of studies on continuity of SPI programs. Therefore, it was decided to carry out a systematic literature review (SLR) on the implementation of SPI programs. The study followed the steps defined in Kitchenham [16]: (i) planning, (ii) identification, and (iii) the selection of primary studies.

The research protocol was defined in the planning phase (purpose of the review, research questions, inclusion and

exclusion criteria, and evaluation of the selected studies). The research questions were defined: i) Which factors positively influence the implementation of software process improvements? and; ii) Which factors negatively influence the implementation of software process improvements?

In the identification phase, the search for relevant studies was carried out in the electronic bases of the ACM Digital Library, IEEE Explore, Science Direct and Springer Link, from December/2012 to January/2013, with supplementation in May/2014. The search period comprised the years 2002 to 2013. Publications were considered in journals and conferences, written in English. These procedures identified 2,474 articles.

The articles were selected considering three main activities i) reading of the titles of the articles; ii) reading of the abstracts of the articles and; iii) reading of whole article. Exclusion criteria were secondary articles and primary articles not related to implementation of SPI programs. The exclusion of these items was carried out in pairs, resulted in 51 papers which were analyzed with the open and axial coding procedures from the Grounded Theory [17]. The validation was done by specialists in SP, who performs implementations and assessments of maturity models.

The analysis of the studies from the systematic review resulted in 03 conceptual categories on success factors in the implementation phase of SPI: 35 Properties of Critical Success Factors (PCSF); 13 Critical Success Factors and; 04 macro categories of Critical Success Factors. Figure 1 shows the Critical Success Factors (CSF) grouped by macro categories of CSF, as show Table 1.

**Table 1. Classification of Critical Success Factors.**

Categories	Critical Success Factor	CSF Properties
Human	Motivation and acceptance of change	Acceptance of change; Motivation for change.
	Support, commitment and involvement	Support, commitment and involvement (senior management and employees).
	Technical and personal skills	Technical and methodological skills; Personal skills (behavioral skills, attitudes and behaviors).
Improvement Project	Implementation strategies	Training; SPI project management; Adaptation of the SPI to the needs of the company; Gradual implementation of improvement project; Definition of implementation strategy; Consideration of culture (regional and organizational); Pilot projects; Selection of suitable professional for improvement.
	Resources	Availability of human resources; Availability of infrastructure resources; Availability of financial resources; Availability of external resources (consultancy).
	Adequate consultancy	Relationship of trust between consultancy and company; Adequate consultancy work; Easy access of the company team to the consultancy.
Organizational	Communication	Adequate communication; Awareness of benefits.
	Goals	SPI goals aligned to the business; Clear and relevant SPI goals.
	Leadership	Existence of SPI leadership
	Internal and external policies	Internal policies that support SPI; External policies that support SPI.
	Organizational structure	Formalization of functions and responsibilities; Stable business environment.
	Return on investment	Visibility of return on investment.



Processes	Processes	Standardization of new processes; Monitoring of new processes; Institutionalization of new processes, and metrics.
-----------	-----------	--

### III. SURVEY

This is an exploratory study and, as such, it aims to provide a better understanding of the theme [18]. The research question that directed this work was: *"What are the factors that influence the maintenance or abandonment of SPI programs?"*. For this purpose, a survey was conducted involving consultants and appraisers of software processes in order to assess the factors found in the literature and identify new factors in the field. The study followed the script proposed by Kitchenham et al. [19], establishing the activities described in the following sections.

#### A. Identifying the research goals

The aim of this study was to identify which factors of implementation identified in the systematic literature review, influence the maintenance or abandonment of software process improvement programs.

#### B. Identifying and Projecting the Sample

The population of this study was composed of Brazilian consultants and appraisers of the MPS.BR program, accredited by SOFTEX, and consultants and appraisers of the CMMI models. To identify the sample, a search was made from the SOFTEX website (<http://www.softex.br>), which identified 473 consultants and 124 appraisers of the MPS.BR program, affiliated with SOFTEX at the time of the research. However, an attempt was made to select a sample of professionals who were effectively exercising process improvement activities. A selection of professionals involved in the MR-MPS-SW model was conducted. This was based on the document entitled Result of Software Process Appraisal, which contains all the information of the appraisal conducted at the company, including the name of the lead appraiser and the assistant appraiser(s). This document is published on the SOFTEX website in the section on current appraisals of the model, for each company successfully appraised for the MR-MPS-SW and with an up-to-date appraisal, i.e., conducted in the last three years. From this search, there are currently thirty professionals involved in appraising the MPS.BR program in Brazil.

#### C. Preparing the Questionnaire

The questionnaire was developed based on the implementation of the SPI identified in the SLR. It was divided into two sections. The first contained six objectives and subjective questions to characterize the professional by his experience. The second was divided into two parts: factors with a positive influence and factors with a negative influence. The questionnaire in Portuguese is available at [https://pucpr.co1.qualtrics.com/jfe/form/SV\\_ehaV0rvIW5B4wrB](https://pucpr.co1.qualtrics.com/jfe/form/SV_ehaV0rvIW5B4wrB).

For the factors with a positive influence, the following question was asked: *"In your opinion, to what extent does each factor contribute in practice when appraised companies maintain the use of the defined process?"* For each factor, the respondents were offered three evaluations levels of the Servqual method [20]: minimum acceptable level, maximum desirable level and perceived level. These factors were assessed using the degree of importance according to a scale that varied from 1 (least relevant) to 9 (most relevant).

In this method, the interval between the mean of the acceptable minimum level and the mean of the desired maximum level is called the tolerance range, that is, it comprises values not perceived by the evaluator as being acceptable for the performance of that indicator. In this study, we consider that the factors with averages situated within the tolerance range, are factors perceived by the respondents as performed in the organizations practice. Factors with averages below the tolerance range indicate that these factors are not performed in the organizations' daily lives.

For the factors with a negative influence, the following question was asked: *"In your opinion, to what extent does each factor contribute in practice when companies abandon their processes?"* For this evaluation, a 9-point Likert scale was shown, starting with minimum value 01 (Totally disagree) to maximum value 9 (Totally agree). The respondents could also add other factors that they considered relevant for both maintenance and abandonment of SPI programs, using as open answer box.

The last section of the questionnaire included a confidentiality agreement, ensuring that the individual information would not be made public.

#### D. Application of the pilot test

Before forwarding the questionnaire to the target public, a pilot test was conducted with one consultant of the CMMI-DEV model and one of the MR-MPS-SW model to evaluate the tool and the content of the questionnaire. The questionnaire was then distributed using a link sent by e-mail.

#### E. Validity threats

Regarding content validity in the design of the instrument [18], a pilot survey was conducted with the aim of evaluating whether the questions were easily understood and whether the degree of information provided by the questionnaire was appropriate. The participants reviewed the questionnaire and suggested some small changes to wording, aiming to facilitate understanding.

External validity refers to the conditions of generalization, i.e. the representativeness of the sample and the correspondence between respondents and the unit of analysis. Thus, to avoid inconsistent responses on the problem,

respondents that were practitioners and experienced in SPI programs were identified.

#### IV. IDENTIFICATION OF PERCEPTION OF SPI EXPERTS

##### A. Professional Profile

Twenty-one SPI specialists (representing 70% of the sample) participated in the study. The professionals were characterized by their experience in implementations and assessments. We consider as experienced the specialists who performed more than five implementations and/or assessments. We identified 18 experienced consultants and 14 lead appraisers in the MR-MPS-SW model and 09 consultants and 04 experienced appraisers in the CMMI-DEV model.

We identified that the total number of implementations performed by these professionals up to the time of the survey totaled 485 implementations in the MR-MPS-SW model (84% of the total assessments performed in the country) and 100 implementations in the CMMI model (45% of the country's total assessments). This shows that the sample of the participants is very representative in relation to the total evaluations of the maturity models implemented in the country.

##### B. Critical Maintenance Factors (CMF)

This section will present the analysis of the data found, considering the three levels of evaluation for each factor of the SERVQUAL method.

Table 2 shows the experts' perception for the maintenance factors (positive influence) of the category of **“human factors”** for the post-evaluation period. Regarding the minimum acceptable, human factors are in a range of importance with a value higher than 05, the average of the method, and under 07, which reinforces the importance attributed to these factors. For the Perceived Level, only factors of support senior management and employees are slightly above the minimum acceptable. The other factors are below the minimum acceptable, indicating that, according to the perception of the respondents, in practice the organizations are not paying due attention to these issues.

TABLE 2. Critical Maintenance Factors: Human.

Maintenance Factor	SERVQUAL		
	NMA	NID	NP
Support of senior management	6,67	8,62	6,71
Support of employees	6,19	8,24	5,67
Technical and methodological skills	5,86	7,71	5,57
Personal skills (behavioral skills, attitudes and behaviors)	6,05	7,95	5,71
Acceptance of change	6,61	8,44	6,22
Motivation for change	6,10	8,15	5,70

Table 3 shows the experts' perception for the maintenance factors (positive influence) of the category of **“process”**. It is worth noting that the standardization of new processes was

the only factor within the zone of tolerance with the score for the Maximum Desired Level (6.43) and the Minimum Acceptable Level (6.29). The easy processes obtained a high score for the Maximum Desired Level (8.52) and low for Perceived Level (5.88), indicating that this factor is considered important but in practice, this does not occur in the organizations.

TABLE 3. Critical Maintenance Factors: Process.

Maintenance Factor	SERVQUAL		
	NMA	NID	NP
Standardization of new processes	6,29	8,05	6,43
Monitoring of new processes	6,90	8,62	6,19
Institutionalization of new processes	7,05	8,81	6,38
Adequate processes (Easy)	6,94	8,52	5,88
Adequate metrics	6,67	8,71	5,38

Table 4 shows the experts' perception for the maintenance factors (positive influence) of the category of **“SPI project”**. Only the Definition of SPI project implementation strategy had a score slightly higher than the minimum acceptable (6.05) for the Perceived Level (6.14). Nevertheless, it is worth emphasizing that the score for the Perceived Level is very close to the minimum acceptable. This indicates that despite being present in the organizations, it is not sufficient to ensure the use of the processes. The factors with the lowest Perceived Level values are Consideration of regional culture (5.33) and Conducting a pilot project for new processes (5.90). The values of the remaining factors were higher than 06 for Perceived Level.

TABLE 4. Critical Maintenance Factors: SPI project.

Maintenance Factor	SERVQUAL		
	NMA	NID	NP
Definition of implementation strategy	6,05	8,29	6,14
SPI project management	6,67	8,52	6,05
Consideration of culture organizational	6,48	8,14	6,05
Consideration of culture regional	5,71	7,35	5,33
Training	6,57	8,29	6,05
Adaptation of the SPI to the needs of the company	7,00	8,60	6,60
Gradual implementation of SPI	6,50	8,27	6,27
Availability of human resources	7,10	8,76	6,14
Availability of financial resources	6,95	8,57	6,29
Availability of external resources	6,35	8,20	6,15
Selection of suitable professional for improvement	6,38	8,48	6,00
Pilot projects	6,48	8,33	5,90

Table 5 shows the experts' perception for the maintenance factors (positive influence) of the category **“Organizational factors”**. SPI goals aligned to the business and Existence of leadership scored higher for the ideal level (8.71). For the perceived level, the factors with the lowest scores were: Consciousness on benefits (5.81), ROI visibility (5.19),

communication (5.62) and internal support policies (5.52), which indicates that in practice they may be neglected. External support policies had the lowest score for the acceptable minimum level (5.35), which indicates that in the opinion of experts, this factor is not as important for continuity of RLS programs.

TABLE 5. Critical Maintenance Factors: Organizational.

Maintenance Factor	SERVQUAL		
	NMA	NID	NP
Awareness of benefits	7,00	8,67	5,81
Adequate communication	6,71	8,48	5,62
Formalization of functions and responsibilities	7,10	8,76	6,38
Ambiente empresarial estável	6,40	8,40	6,10
Internal policies that support SPI	6,43	8,10	5,52
External policies that support SPI	5,35	7,63	6,05
Existência de liderança	7,00	8,71	6,48
SPI goals aligned to the business	6,86	8,71	6,00
Clear and relevant SPI goals	6,81	8,57	6,05
Visibility of return on investment	6,52	8,43	5,19

### C. Critical Abandonment Factors (CAF)

This section shows in ascending order for the average of the factors that the specialists identify the possible causes for the abandonment of SPI programs. The results are presented below by category.

In the “**Human Factors**” category (Table 6), the factors had average scores higher than 06. Therefore, all the factors in this category are considered to influence the abandonment of SPI. It is worth highlighting that the lack of support from the upper management was considered the most critical factor in the abandonment of these initiatives (8.19). This was followed by the time/commercial pressures factor (7.75) and Work load (7.52). This shows that the specialists perceive that a lack of adequate support from the upper management and pressures at work on the executors of the process are factors that hinder the continuation of process improvements.

TABLE 6. Critical Abandonment Factors: Human.

Critical Abandonment Factor	Avg.	Standard Deviation
Lack of technical and methodological competencies	6.05	1.50
Lack of personal competencies	6.10	1.45
Resistance from employees	6.43	1.75
Low employee motivation	6.71	1.68
Lack of employee involvement	6.95	1.24
Bad/negative experiences	7.15	1.67
Imposition	7.25	1.69
Work load	7.52	1.50
Time/commercial pressures	7.75	1.49
Lack of support from upper management	8.19	1.25

Processes category shown in Table 07, all the factors were considered critical to the abandonment of SPI programs. The factor with the lowest average (5.67) was Reduced creativity, and the factor with the highest average (8.05) was Lack of monitoring. Four factors had averages higher than 07:

Inadequate metrics, Extensive documentation, Bureaucracy and Complex processes.

TABLE 7. Critical Abandonment Factors: Processes.

Critical Abandonment Factor	Avg.	Standard Deviation
Reduced creativity	5.67	2.76
Lack of standardization	6.19	1.69
Complicated framework	6.67	2.37
Lack of flexibility	7.00	1.82
Inadequate metrics	7.19	1.75
Extensive documentation	7.29	1.65
Bureaucracy	7.33	2.01
Complex processes	7.38	1.20
Lack of monitoring	8.05	1.12

The SPI project category is shown in Table 08. To the respondents, the most critical abandonment factors were: Lack of consideration for organizational culture (7.60) and Lack of human resources (7.52), with average values very close to 08 on the evaluation scale. The least influential factor for the abandonment of SPI programs was Implementation on a large scale (large scope of the improvement project causing coordination problems) (5.75). The considering regional culture factor (4.64) was not considered critical to the abandonment of SPI. The other factors scored over 06 and 07 on the evaluation scale.

TABLE 8. Critical Abandonment Factors: SPI Project.

Critical Abandonment Factor	Avg.	Standard Deviation
Not considering regional culture	4.64	2.87
Implementation of SPI project on a large scale	5.75	2.39
Lack of training	6.19	1.91
Lack of implementation strategy	6.40	2.11
Lack of infrastructure resources	6.86	1.59
Lack of financial resources	6.90	2.10
Lack of SPI project management	6.95	2.13
Lack of human resources	7.52	1.66
Not considering organizational culture	7.60	1.38

Table 9 shows the Organizational Factors category. Five factors were more frequently identified by the respondents as being critical to the abandonment of SPI: High turnover (7.0); Lack of awareness of the benefits of the improvement project (7.14); Lack of alignment between business and the goals of the improvement project (7.43); Lack of clarity of the goals of the project and (7.38) and Lack of understanding of the return on the investment (7.52), with averages up to 07. The other factors had averages higher than 06. Therefore, they are also factors that should be taken into consideration regarding the continuity of improvement programs.

TABLE 9. Critical Abandonment Factors: Organizational.

Critical Abandonment Factor	Avg.	Standard Deviation
Lack of formalism of functions and responsibilities	6.48	1.97
Inadequate communication	6.76	1.79
High cost of SPI	6.81	1.60
High turnover	7.00	1.89
Lack of awareness of benefits	7.14	1.71
Lack of clarity in SPI goals	7.38	1.28
Lack of alignment between business and SPI	7.43	1.60

Lack of understanding of return on investment	7.52	1.83
---	------	------

## V. CONCLUSION

Studies on the successful implementation of software process improvement programs have been the focus of researchers in the past two decades. However, the reasons why these companies end up maintaining or abandoning these initiatives is a field that has not seen a great deal of research. Thus, this study sought to bridge this gap. The research concludes that the continuation or abandonment of software process improvement programs depends critically on 13 factors, namely: i) motivation and acceptance of change; ii) support, involvement and commitment; iii) technical and personal competencies; iv) strategies for SPI; v) resources and communication; vi) goals; vii) organizational structure; viii) policies; ix) return on investment; x) leadership; xii) adequate external consultancy services and; xiii) processes.

The study also found that the perception of SPI specialists is that in practice organizations do not pay due attention to these issues. This is probably one of the reasons why software development organizations abandon these programs.

The main limitation of this study is that it does not delve more deeply into the issues identified in the survey. There are some unanswered questions as: What does it happen to a company after the official evaluation using a maturity model? Why is not possible to identify a consistent evolution in maturity levels? Do companies abandon only the official evaluation process or abandon the overall improvement program? Do they abandon completely the implanted process or just parts of it?

For this purpose, it would be necessary to conduct a deeper study of the appraised software organizations to understand why these companies maintain and why they abandon software process improvement programs. Therefore, the next stages of this work will involve conducting a case study of the appraised software companies to assess the findings of this exploratory study in practice. Our preliminary case studies with the organizations showed some similar factors.

In future studies, it would be interesting to investigate how the maintenance of these SPI programs occurs in practice, i.e., looking at companies that continue their active appraisals and companies whose appraisals are overdue.

## REFERENCES

- [1] Sommerville, I., *Software Engineering*. 9th ed. Pearson Prentice Hall, 2011.
- [2] A. Fuggetta. *Software Process: a roadmap*. In *Proceedings of the conference on the future of software engineering – international conference on engineering*, Limerick, Irlanda, 2000, p. 25-34.
- [3] G.M. Kituyi and C. A. Amulen. *Software capability maturity adoption model for small and medium enterprises in developing countries*, In *The Electronic Journal on Information Systems in Developing Countries EJISDC*, v.55, n.1, p.1-19, 2012.
- [4] CMMI PRODCUT TEAM. *CMMI for Development*. (CMU/SEI-2010-TR-033). Versão 1.3. Pittsburg: Software Engineering Institute – Carnegie Mellon University, 2010.
- [5] Sociedade para Promoção da Excelência do Software Brasileiro (SOFTEX). *Guia Geral MPS de Software - 2016*. Disponível em: <[http://www.softex.br/mpsbr/\\_guias/default.asp](http://www.softex.br/mpsbr/_guias/default.asp)>.
- [6] Secretaria de Economia do México - SEM. *Modelo de Processos para la Industria de Software – MoProSoft versión 1.3*, Agosto de 2005. Disponível em: <http://www.comunidademoprosoft.com>.
- [7] Lepments, M.; McBrid, T.; Ras, E. *Goal alignment in process improvement*, In the *Journal of Systems and Software*, v.85, p.1440–1452, 2012.
- [8] Elm, J.; Goldenson, D. *The Business Case for Systems Engineering Study: Detailed Response Data*. (CMU/SEI-2012-SR-011). Software Engineering Institute, Carnegie Mellon University, 2013. <http://www.sei.cmu.edu/library/abstracts/reports/12sr011.cfm>.
- [9] Kalinowski, M.; Weber, K.; Franco, N.; Zanetti, D.; Santos, G. *Results of 10 Years of Software Process Improvement in Brazil Based on the MPS-SW Model*. In *Quality of Information and Communications Technology (QUATIC)*, 2014 9th International Conference, p. 28-37, Sept. 2014.
- [10] Dyba, T. *An Empirical Investigation of the Key Factors for Success in Software Process Improvement*. *IEEE transactions on software engineering*, vl. 31, n5, p. 410-424, May 2005.
- [11] Khan, A. A. ; Keung, J. *Systematic review of success factors and barriers for software process improvement in global software development*. *IET Software* (Volume: 10, Issue: 5, 10 2016). DOI: 10.1049/iet-sen.2015.0038.
- [12] Bayona, S.; Calvo-Manzano, J.A.; Feliu, T.S. *Critical Success Factors in Software Process Improvement: A Systematic Review*. *International Conference on Software Process Improvement and Capability Determination - SPICE 2012: Software Process Improvement and Capability Determination* pp 1-12.
- [13] Almeida, C. D. A., Albuquerque, A. B., Macedo, T. C. “Analysis of the continuity of software processes execution in software organizations assessed in MPS.BR using Grounded Theory”, *XXIII Software Engineering and Knowledge Engineering*, Miami 2011.
- [14] Uskarci, A., Demirörs, O. *Do staged maturity models result in organization-wide continuous process improvement? Insight from employees*. In *Computer Standards & Interfaces* 52 (2017) 25–40.
- [15] Sociedade para Promoção da Excelência do Software Brasileiro (SOFTEX). *Guia de Implementação – Parte 11: Implementação e Avaliação do MR-MPS-SW:2012 em Conjunto com o CMMI-DEV v1.3*. Agosto, 2012. Disponível em: <[http://www.softex.br/mpsbr/\\_guias/default.asp](http://www.softex.br/mpsbr/_guias/default.asp)>.
- [16] Kitchenham, B. *Procedures for Performing Systematic Reviews*. Software Engineering Group, Keele University, Keele, UK, July, 2004.
- [17] Strauss, A.; Corbin, J. *Basics of Qualitative Research*, 2<sup>o</sup> ed.: Sage Publications, Thousand Oaks, London New Delhi, 1998, 312p.
- [18] Forza, C. *Survey research in operations management: A process-based perspective*. *International Journal of Operations & Production Management*, v.22, n.2, p.152-194, 2002.
- [19] Kitchenham, B.; Pflieger, S.L.P. *Principles of Survey Research: Parts1-6*. *ACM SIGSOFT - Software Engineering Notes*, 27, n. 1-6, Setembro 2002.
- [20] Parasuraman, A.; Berry, L.L.E.; Zeithaml, V.A. *SERVQUAL: A multiple-item scale for measuring consumer perception of service quality*. *Journal of Retailing*, York University, 1998.

# Improving Code Summarization by Combining Deep Learning and Empirical Knowledge

Lingbin Zeng, Xunhui Zhang, Tao Wang, Xiao li, Jie Yu, Huaim Wang  
College of Computer Science  
National University of Defense Technology  
Changsha, Hunan, China  
{zenglingbin16, zhangxunhui, taowang2005, xiaoli, yujie16, hmwang}@nudt.edu.cn

**Abstract**—Code summaries are human-readable text that describes the functionality of code blocks. Software developers use code summaries to understand the specification of API while code retrieve system relies on code summaries for effective code search. However, code summaries are often written by software developers. Writing good code summaries usually requires great effort. It could be helpful if developers use automatic code summarization system to generate code summaries. Recently, some works have applied deep learning methods to generate code summaries for code snippets. However, those deep learning methods treat code snippets as streams of text tokens while ignoring the inherent code structure information. In this paper, we propose a novel code summarization method named the CDE-Model (Code summarization by Deep learning and Empirical knowledge) that combines inherent code structure information with deep learning models. The CDE-Model proposes several empirical strategies to transform code snippets to refined code representation and feeds them into an encoder-decoder neural network for text generation. We conduct large-scale experiments on 1500 popular Java projects on GitHub<sup>1</sup> with 396,184 pairs of code snippets and summaries. Experimental results show that the quality of code summaries generated by our CDE-Model is better than other two methods. To the best of our knowledge, this paper is the first to combine code structure information with deep learning.

**Keywords:**Code summarization; GitHub; Recurrent neural network; Java language.

## I. INTRODUCTION

The rapid development of open source software (OSS) provides a massive reusable resource for software development [1] [2] [3] [4]. In OSS, code summaries are very important as they describe the functionality of code blocks in the form of human-readable text [5]. On one hand, software developers use code summaries to understand the specification of API. On the other hand, code retrieve system relies on code summaries for effective natural language code search [6]. Thus, it is crucial to maintain high-quality and adequate code summaries in OSS projects. However, code summaries are often manually written by software developers and writing good code summaries usually requires great human effort. In our pilot study, we found that the annotation rate for even famous projects on GitHub are very low (see Table I). This could significantly hinder software innovation. It could be helpful if developers

use automatic code summarization system to generate code summaries.

TABLE I: Annotation rate of several famous GitHub projects

Project name	Lines of codes	Code annotation rate
Redis	83,233	23.7%
cocos2d-x	461,685	8.3%
Hadoop	1,217,655	13.1%
blueprints	33,356	8.6%
Tensorflow	300,864	9.2%
hebel	10,040	8.1%
jQuery	42,300	12.0%

To address this problem, automatic code summarization systems have been proposed for generating code summaries. Previous code summarization methods usually rely on information retrieval and text mining methods. For example, Vassallo C et al. [7] propose the CODES method which extracts candidate summaries from StackOverflow<sup>2</sup> discussions and creates Javadoc descriptions based on social connection theory. Wang et al. [8] introduce an approach to analyze constructs of code snippets and extract keywords to produce code summaries.

Recently, deep learning methods have become a popular research topic in code summarization research. Iyer et al. [1] use long short-term memory (LSTM) network to generate code summaries according to code context. The model they trained is especially useful for short code snippets. Although the deep learning based methods have shown effectiveness in code summarization, they usually treat code snippets as stream of text tokens while ignoring the inherent code structure information. Explicitly utilizing code structure information, such as loop, condition and equation expression, into deep learning models may improve the performance of code summarization.

In this paper, we propose a novel code summarization approach named the CDE-Model to automatically generate code summaries. Different from the previous methods, our approach first utilizes code syntax specification to embed explicit code structure information into raw source code. Then, we feed the transformed code snippets into an encoder-decoder neural network for summary generation. The key contributions of our study are as follows:

<sup>1</sup><https://github.com>

DOI reference number: 10.18293/SEKE2018-191

<sup>2</sup><https://stackoverflow.com>

- A new code summarization framework that combines deep learning and code structure information.
- A large-scale dataset of Java code summarization task, which contains 396,184 pairs of source code and text summaries.
- A high-performance neural network model that leads to a significant improvement of BLEU readability score.

The rest of this paper is organized as follows. Section II reviews previous works. Section III describes our methods. Section IV shows the experimental result and the last section concludes this paper.

## II. RELATED WORK

Many works have been proposed for code summarization. These works can be divided into three categories including information retrieval methods, text mining methods and deep learning methods.

### A. Information retrieval

Wong et al. [9] propose an approach which uses information retrieval methods to associate comments in the Q&A community with code snippets. Vassallo et al. [7] propose to use social connection theory to connect code snippets and comments in the StackOverflow. Bahihi et al. [10] present a method named CrowdSummarizer which exploits crowdsourcing, gamification and natural-language processing to automatically generate high-level summaries of Java program methods. Moreno et al. [11] present a technique to automatically generate human readable summaries for Java classes with heuristics rules.

### B. Text mining

Wang et al. [8] present an approach to automatically generate natural language descriptions of Java methods. They identify the statements in the code snippets and extract the keywords to generate sentences as code summaries. McBurney et al. [12] propose a source code summarization technique that generate English descriptions of Java methods by analyzing how those methods are invoked. Hill et al. [13] propose an approach that automatically extracts natural language phrases from source code and categorize the phrases in a hierarchy. These methods could generate the logic description for the code snippets. However, they could not generate code summaries in a high-level abstraction.

### C. Deep learning

Iyer et al. [1] present a deep learning model to generate code summaries automatically. They use the LSTM network, a recurrent neural network (RNN), to encode code snippets into a fixed vector and decode vector into code summaries. The model has a good performance on short code snippets and could generate the new words that are not appeared in the code snippets. Paulus R et al. [14] introduce a neural network model with intra-attention, and propose to combine supervised word prediction and reinforcement learning to generate summaries. The result shows that summaries created by reinforcement learning model are more readable. Loyola et

al. [15] propose a model to automatically describe changes introduced in the source code of a program with encoder-decoder architecture. The result showed that it can generate feasible and semantically sound descriptions.

## III. OVERVIEW OF CDE-MODEL

In this section, we describe the framework of our approach. Similar to the works of Iyer et al. [1] and Paulus R et al. [14], we also model the code summarization problem as a sequence-to-sequence learning task which maps a sequence of code tokens into a sequence of natural language tokens. Different to their approaches, we consider the inherent syntactic structure information in the code tokens. We propose to analyze code syntax and weave the syntactic structure information into deep learning networks.

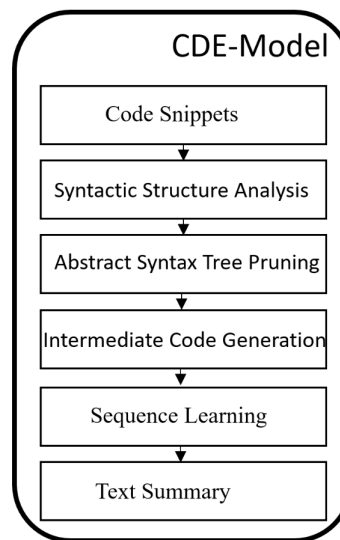


Fig. 1: Framework of the CDE-Model

Figure 1 shows the framework of our approach which consists of four steps for code summarization task. Specifically, given a code snippet, we first conduct syntactic structure analysis to generate the abstract syntax tree (AST) which is very useful for code analysis and mining. Second, for code summarization task, what we want is concise human-readable description rather than programming logic description. Thus, not all of the AST nodes can be useful. We propose several heuristic strategies to prune the AST to simplify the code snippet. Third, based on the pruned AST, we generate an intermediate code which is simpler than raw code snippets while preserving the code structure information. Finally, we feed the intermediate code into a sequence-learning neural network model to generate text summary.

The key step in our approach is to prune the AST tree. Programming languages have control statements such as loop and condition to determine the execution trace of software. In this paper, we mainly discuss the pruning strategies that deal with the loop nodes in the AST (see figure 2). Because loop is one of the most common control statements in the code snippets [16].

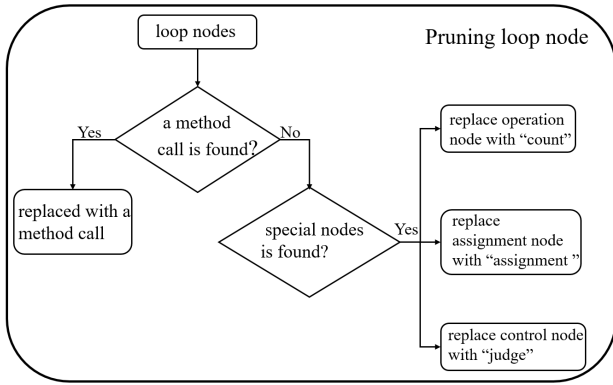


Fig. 2: The key steps in pruning loop nodes

### A. Pruning loop node

After we generate the AST from a code snippet, we propose to traverse the AST to find loop nodes. If we find loop nodes such as for and while, we will continue to recursively traverse its child nodes with three different strategies discussed below.

1) *If encountering a method call in the loop subtree:* If we find a method call in the subtree, we will replace the entire loop subtree with a statement of method call. We present an example in figure 3. We can see that `buf.append(b)` is an object method call inside the loop body. Thus, we use text “`buf.append`” to replace the loop node.

```

public String toString() {
    StringBuilder buf = new StringBuilder(length * 2);
    for (int i = 0; i < length; i++) {
        int b = data[i];
        buf.append(b);
    }
    return buf.toString();
}

```

Fig. 3: An example of code snippets

One may say that doing so could remove lots of useful information in the loop subtree. However, we believe that in Java programming language good class method name usually reveals the semantic meaning. In addition, if there are more than one method call, we will choose the last method call in the loop body to replace the entire loop subtree. Because several empirical study show that the later the statement, the more likely it representing the true meaning [14].

2) *If encountering special nodes in the loop subtree:* If there are no method calls in the loop subtree, we traverse the loop subtree from the last child nodes to the first to find three different type of special nodes including operation nodes, assignment nodes and control nodes. We tabulate the processing methods in table II.

If the first encountered node is mathematic notation, such as “+” “-”, “\*”, or “/”, the entire loop subtree will be replaced with a text node of “count”. If the first encountered symbol is “=”, we will give an “assignment” text node as the summary of the loop body. The summary will be taken as “judge” if

TABLE II: Special nodes

Types of nodes	Related symbols	Replacement as text nodes
Operation node	“+”, “-”, “*”, or “/”	“count”
Assignment node	“=”	“assignment”
Control node	“break” “return true” “return false”	“judge”

control statements such as “break”, “return true” or “return false” is first encountered.

3) *Other situations:* If there are no special nodes in the loop body, we select the last node in the loop subtree to replace the loop body.

After pruning the AST, we expand the AST as normal text stream of code snippet. It should be noted that the code structure information has been explicitly embed in the code snippets. To help clarify our algorithm, we give an example in table III to show the difference of the three strategies.

TABLE III: Code snippets after pruning ASTs

	Before pruning	After pruning
Method Call	<pre> void queueIsEmpty() {     for (Node p = head; p != null;         p = p.next)         {Itr it = p.get();         if (it != null) {             p.clear();             it.shutdown();} }     head = null;     itrs = null;} </pre>	<pre> void queueIsEmpty() {     it shutdown ;     head = null;     itrs = null; } </pre>
Special Nodes	<pre> public ContactIrc getContact(final String id) {     if (id == null    id.isEmpty()) {         return null;     }     for (ContactIrc contact : this.contacts) {         if (id.equals( contact.getAddress())) {             return contact;} }     return null;} </pre>	<pre> public ContactIrc getContact (final String id) {     if (id == null    id.isEmpty()) {         return null;     }     Judge     return null; } </pre>
Others	<pre> public void removeAt (int index, int size) {     final int end = Math. min(mSize, index + size);     for (int i = index; i &lt; end; i++) {         removeAt(i);     } } </pre>	<pre> public void removeAt (int index, int size) {     final int end = Math. min(mSize, index + size)     removeAt(i); } </pre>

### B. Sequence learning

We build a sequence-to-sequence generation system for code summarization task. Our approach use RNN with attention-based mechanism and encoder-decoder architecture to produce code summaries. The network architecture is shown in figure 4

The RNN Encoder-Decoder with attention-based mechanism consists of two RNN that act as an encoder layer and a decoder layer. The encoder layer maps a variable-length source sequence to a fixed-length vector. The decoder layer maps the vector representation back to a variable-length target sequence [17]. The biggest difference in the attention model is that it does not require the encoder to encode all input information into a fixed-length vector [18].

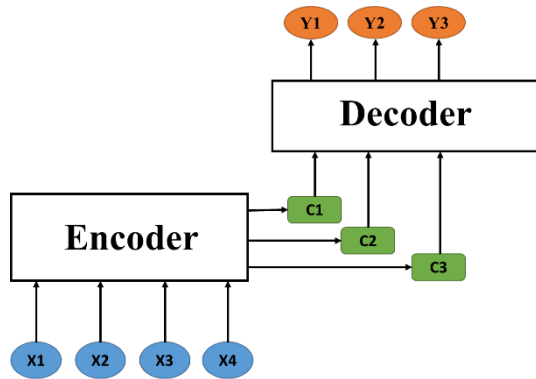


Fig. 4: Basic framework of encoder-decoder neural network

In this paper, we use the LSTM cell for encoder and decoder layers. LSTM has been shown to have good performance in text translation model with attention mechanism to generate one word at a time. A detail introduction of the LSTM and encoder-decoder related neural network can be seen in [19] [20].

#### IV. EXPERIMENTS

##### A. Dataset

In this section, we describe the dataset used in this paper. We collect data from GitHub which is one of the most popular open source community in the world. All the data can be downloaded in the Trustie<sup>3</sup>, a famous open source community in China [21].

We briefly describe the dataset generation process. First, we search in the GHTorrent [22] and use Kraken [23] to crawl the top 1500 popular Java projects ranked by star numbers. Second, we use java-parser<sup>4</sup> tools to analyze source code and extract the code snippets and comments of java API respectively [24]. Those pairs of code snippets and text comments will be treated as training data for our neural network models. In addition, to improve data quality, we remove noisy comments in the dataset. Specifically, we remove the comments shorter than two words. Special symbols except dash “-” and underline “\_” are also removed. Eventually, we generate 396,184 pairs of code snippets and summaries. As shown in table IV, the average length of code snippets is of 108.7 words. The average length of text summaries is of 8.8 words.

TABLE IV: Basic information of datasets

Total pairs	Average code length	Average summary length
396,184	108.7	8.8

##### B. Experiment settings

We implement the CDE-Model based on Tensorflow<sup>5</sup>. We build an encoder-decoder network with 6 layers each with 128

<sup>3</sup>www.trustie.net

<sup>4</sup>https://www.eclipse.org/jdt/

<sup>5</sup>https://www.tensorflow.org/

units. We restrict the input vocabulary size to the top 40,000 most frequent code tokens, and the output text vocabulary to the top 40,000 most frequent tokens in the training set. We train the models with a batch size of 128 and a learning rate  $\alpha$  of 0.5. We do not stop training the model until perplexity score becomes stable. We use 85% of the dataset for training, 10% for validation and 5% for testing. We have published all of our data and codes in Trustie<sup>6</sup>.

To compare our models with the other methods, we also implement two other models based on the CDE-Model. The Del-Model just removes the loop structure in the code snippets while the Gen-Model keeps the code unchanged. We list the three models in table V.

TABLE V: Methods to be compared in experiments

Methods	Description
CDE-Model	Replace the loop structure with heuristic strategies
Del-Model	Delete the loop structure in the code snippets
Gen-Model	Keep the loop structure unchanged in the code snippets

##### C. Evaluation metrics

We use Bilingual evaluation understudy (BLEU) score as the evaluation metric in this paper. BLEU is an algorithm for evaluating the quality of machine translated text from one natural language to another [25]. Recently, it has become a popular evaluation metric in deep learning based code summarization. In this paper, we report the average BLEU-4 score in experiments, which is often used to measure the quality of text sentences.

##### D. Experimental results

We tabulate the experiments results in table VI. We find that the CDE-Model outperforms the other methods by a large margin. Specifically, the BLEU-4 values (the second column in table VI) of the CDE-Model is 10.4% higher than the Del-Model and 32.5% higher than the Gen-Model. In addition, we conduct a detailed performance analysis on the code snippets containing loop structure. In our test dataset, there are 2296 code snippets which contain the loop nodes. We measure the BLEU-4 score (the third column in table VI) on this data and find that the CDE-Model is much better than the other methods. Specifically, the BLEU-4 score of the CDE-Model is 6.7% higher than the Del-Model and 85.38% higher than the Gen-Model. This means that utilizing code structure information with empirical data processing strategies into deep learning models can improve the code summarization task significantly.

TABLE VI: BLEU-4 score of different methods

Models	BLEU-4 values	BLEU-4 values containing loop node
CDE-Model	0.52801	0.47548
Del-Model	0.47817	0.42813
Gen-Model	0.37254	0.23058

<sup>6</sup>https://www.trustie.net/projects/1738



Second, we randomly sample 100 code snippets containing loop nodes to evaluate the three methods. As shown in figure 5, the performance curve of the CDE-Model wins the Del-Model and the Gen-Model in most cases. This further demonstrates that the loop structure has significant impact on the model performance. Replacing the loop structure (by the CDE-model) is much better than the methods that delete the loop structure (by the Del-Model) or keep the loop structure unchanged (by the Gen-Model).

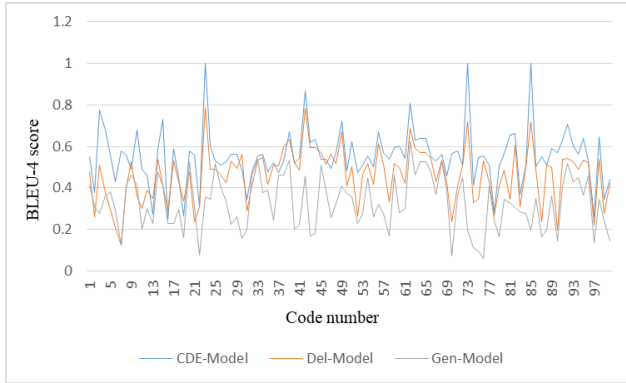


Fig. 5: BLEU-4 score on the sampled data

Third, we analyze the BLEU-4 score of the three methods on the code snippets that satisfying the three conditions (see Section III). As shown in table VII. We find that the BLEU-4 score of the CDE-Model is always better than others in all the three conditions. Specifically, the BLEU-4 score of the CDE-Model is 7% higher than the Del-Model and 84.91% higher than the Gen-Model in *method call* condition. Besides, the CDE-Model owns the best performance than others in *special nodes* condition. Whats more, the BLEU-4 score of the CDE-Model is 4.36% higher than the Del-Model and 66.16% higher than the Gen-Model in *others* condition. This results mean that the three strategies proposed in the CDE-model are very effective.

TABLE VII: BLEU-4 score in three different strategies

Models	CDE-Model	Del-Model	Gen-Model
Method call	0.42410	0.39617	0.22935
Special nodes	0.52627	0.51252	0.25674
Others	0.47175	0.45205	0.28392

### E. Case study

Why does the CDE-Model perform better than other two methods? In this subsection, we conducted a detailed case study for qualitative analysis.

Table VIII shows a code snippet with a loop body. The gold standard is written by professional developers. We can see that the code summary generated by our CDE-Model is the most closer to the gold standard compared with the Del-Model and the Gen-Models. Our method successfully captures the semantics of “empty test” while the summaries generate by the Del-Model and the Gen-Model is not meaningful.

For the poor results of Gen-Model, it may be the reason that the loop structure in normal code snippets is usually very complex. Blindly feeding loop body into the encoder-decoder LSTM network may introduce noise. This is why the word “clear” appears both in loop body and Gen-Model summary.

Although the Del-Model removes the noisy loop structure, it loses too much information and thus cannot generate good results. Therefore, in deep learning based code summarization task, it is very important to consider the inherent code structure information.

TABLE VIII: Code snippets and summaries

Code snippet	<pre>void queueIsEmpty() {     for (Node p = head; p != null; p = p.next){         Itr it = p.get();         if (it != null) {             p.clear();             it.shutdown();         }     }     head = null;     itrs = null; }</pre>
Gold standard	Called whenever the queue becomes empty
CDE-Model	Called when the buffer has been empty
Del-Model	Called to iterate the observers of this node
Gen-Model	The clear blocks that have been returned

## V. CONCLUSION & FUTURE WORK

In this paper, we present and implementation a novel code summarization method name the CDE-Model which combines the deep learning and code structure information. The major feature of the CDE-Model is that it traverses the abstract syntax tree of code snippets to manipulate the complex structural code body to generate intermediate code snippets by empirical strategies. After that, the CDE-model learns an encode-decoder LSTM network for text generation. We conduct large-scale empirical study on 1500 popular Java OSS projects in GitHub. The experimental results and the case study demonstrate that our method is effective.

In the future work, we will try to exploit more advanced sequence learning models to directly encoding AST structure for long code snippets. Besides, we will also study other deep learning methods such as deep reinforcement learning.

## ACKNOWLEDGMENT

The research is supported by the National Grand R&D Plan (Grant No. 2016-YFB1000805) and National Natural Science Foundation of China (Grant No. 61502512, 61702532, 61432020, 61472430 and 61532004).

## REFERENCES

- [1] S. Iyer, I. Konstas, A. Cheung, and L. Zettlemoyer, “Summarizing source code using a neural attention model,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2016, pp. 2073–2083.
- [2] Q. Fan, Y. Yu, G. Yin, T. Wang, and H. Wang, “Where is the road for issue reports classification based on text mining?” in *Empirical Software Engineering and Measurement (ESEM), 2017 ACM/IEEE International Symposium on*. IEEE, 2017, pp. 121–130.

- [3] Y. Zhang, H. Wang, G. Yin, T. Wang, and Y. Yu, "Social media in github: the role of @-mention in assisting software development," *Science China Information Sciences*, vol. 60, no. 3, p. 032102, 2017.
- [4] A. E. Prieto, J.-N. Mazón, A. Lozano-Tello, and L.-D. Ibáñez, "Supporting open dataset publication decisions based on open source software reuse," 2018.
- [5] P. Rodeghero, C. McMillan, P. W. McBurney, N. Bosch, and S. D'Mello, "Improving automated source code summarization via an eye-tracking study of programmers," in *Proceedings of the 36th International Conference on Software Engineering*. ACM, 2014, pp. 390–401.
- [6] I. J. Mujhid, J. C. Santos, R. Gopalakrishnan, and M. Mirakhorli, "A search engine for finding and reusing architecturally significant code," *Journal of Systems and Software*, vol. 130, pp. 81–93, 2017.
- [7] C. Vassallo, S. Panichella, M. Di Penta, and G. Canfora, "Codes: Mining source code descriptions from developers discussions," in *Proceedings of the 22nd International Conference on Program Comprehension*. ACM, 2014, pp. 106–109.
- [8] X. Wang, L. Pollock, and K. Vijay-Shanker, "Automatically generating natural language descriptions for object-related statement sequences," in *Software Analysis, Evolution and Reengineering (SANER), 2017 IEEE 24th International Conference on*. IEEE, 2017, pp. 205–216.
- [9] E. Wong, J. Yang, and L. Tan, "Autocomment: Mining question and answer sites for automatic comment generation," in *Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on*. IEEE, 2013, pp. 562–567.
- [10] S. Badihi and A. Heydarnoori, "Crowdsummarizer: Automated generation of code summaries for java programs through crowdsourcing," *IEEE Software*, vol. 34, no. 2, pp. 71–80, 2017.
- [11] L. Moreno, J. Aponte, G. Sridhara, A. Marcus, L. Pollock, and K. Vijay-Shanker, "Automatic generation of natural language summaries for java classes," in *Program Comprehension (ICPC), 2013 IEEE 21st International Conference on*. IEEE, 2013, pp. 23–32.
- [12] P. W. McBurney and C. McMillan, "Automatic source code summarization of context for java methods," *IEEE Transactions on Software Engineering*, vol. 42, no. 2, pp. 103–119, 2016.
- [13] E. Hill, L. Pollock, and K. Vijay-Shanker, "Automatically capturing source code context of nl-queries for software maintenance and reuse," in *Software Engineering, 2009. ICSE 2009. IEEE 31st International Conference on*. IEEE, 2009, pp. 232–242.
- [14] R. Paulus, C. Xiong, and R. Socher, "A deep reinforced model for abstractive summarization," *arXiv preprint arXiv:1705.04304*, 2017.
- [15] P. Loyola, E. Marrese-Taylor, and Y. Matsuo, "A neural architecture for generating natural language descriptions from source code changes," *arXiv preprint arXiv:1704.04856*, 2017.
- [16] R. Hundt, "Loop recognition in c++/java/go/scala," *Proceedings of Scala Days*, vol. 2011, p. 38, 2011.
- [17] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.
- [18] A. M. Rush, S. Chopra, and J. Weston, "A neural attention model for abstractive sentence summarization," *arXiv preprint arXiv:1509.00685*, 2015.
- [19] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.
- [20] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.
- [21] H. Wang, G. Yin, X. Li, and X. Li, "Trustie: a software development platform for crowdsourcing," in *Crowdsourcing*. Springer, 2015, pp. 165–190.
- [22] G. Gousios and D. Spinellis, "Ghtorrent: Github's data from a firehose," in *Mining software repositories (msr), 2012 9th ieee working conference on*. IEEE, 2012, pp. 12–21.
- [23] L. Zeng, G. Yin, T. Wang, Y. Yu, Q. Fan, Z.-X. Li, J. Yu, and H. Wang, "Kraken: A continuous incremental data acquisition system for github and git repositories," No. 38 A, Xueqing Road, Haidian District, Beijing, China, 2017, pp. 144 – 149, data extraction;Development activity;Development history;GitHub;Incremental data;Open source communities;Regular patterns;Rest API,
- [24] R. Hosseini and P. Brusilovsky, "Javaparser: A fine-grain concept indexing tool for java problems," in *CEUR Workshop Proceedings*, vol. 1009. University of Pittsburgh, 2013, pp. 60–63.
- [25] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2002, pp. 311–318.

# Reverse Engineering Encapsulated Components from Object-Oriented Legacy Code

Rehman Arshad, Kung-Kiu Lau

*rehman.arshad, kung-kiu.lau @manchester.ac.uk*

*School of Computer Science, University of Manchester*

*M13 9PL, United Kingdom*

## Abstract

*Current component-directed reverse engineering approaches extract ADL-based components from legacy systems. ADL-based components need to be configured at code level for reuse, they cannot provide re-deposition after composition for future reuse and they cannot provide flexible re-usability as one has to bind all the ports in order to compose them. This paper proposes a solution to these issues by extracting X-MAN components from legacy systems. In this paper, we explain our component model and mapping from object-oriented code to X-MAN clusters using basic scenarios of our rule base.*

**Key Words** —Reverse Engineering, Component Based Development<sup>1</sup>

## 1. Introduction

The term legacy systems usually refers to such software systems that are outdated, lack proper documentation and cannot support a new feature without breaking another logic yet they are vital to an organisation [5]. Unfortunately, most legacy code was designed with non-modular approach that cannot exploit the luxury of re-usability. For many companies, maintenance or comprehension of legacy code is crucial because some of their functions are too valuable to be discarded and too expensive to reproduce from scratch.

Component based development is a domain that revolves around the construction of systems from pre-built software units i.e., re-usability. Components extraction can reconstruct a legacy system as modular executable architectural

units that can be reused across many systems. Component-directed<sup>2</sup> reverse engineering consists of following steps: 1) Capture the source code in appropriate notation (graph nodes etc.). 2) Define a rule base to map the extracted notation to abstraction model. 3) Formation of clusters i.e., code re-structuring 4) Mapping of clusters to a component model. Output of component-directed reverse engineering approaches is dependent on the definition of component each approach uses. Most approaches use loose definition of component. For them, a component is consisted of methods that belong together as they offer a specific functionality of the system. Such components can be giant classes, clusters or re-formation of the source code to get better cohesion and loose coupling. These approaches neither defines the extraction of explicit interfaces nor the composition mechanism of the extracted components (e.g., [9]). Such components are not feasible for reuse as non-explicit architecture cannot help in achieving a good re-usability.

Few like us, follow the szyperski's definition of components. This definition defines component as "A unit of composition with contractually specified interfaces and explicit context dependencies only" [15]. These approaches extract explicit architecture (components with well-defined composition and interfaces).

Almost all the current reverse engineering approaches that extract explicit components are based on ADLs<sup>3</sup>. ADLs define required and provided services as ports (composition mechanism of ADLs). Ports use (indirect) method calls at code level to compose components together. ADL-based components have three major shortcomings from re-

<sup>1</sup>DOI reference number: 10.18293/SEKE2018-111

<sup>2</sup>Reverse Engineering that aims for the extraction of components.

<sup>3</sup>Components based on architecture description languages.

usability point of view: 1) Inability to select/de-select/alter ports without changing the code manually at all required places (for every single composition) to compose the components after retrieval. 2) One has to bind all the ports in order to reuse an ADL component i.e., non-flexible reusability. 3) It is impossible to re-deposit<sup>4</sup> a configured composition of components for reuse (e.g., composite component). Components have to be retrieved and configured as many times as the same composition is required. To the best of our knowledge, no such component-directed reverse engineering approach exists that can: do code-independent composition, allow to reuse the components without binding all services (ports) and support the re-deposition of composed components for further reuse or composition.

This paper presents a reverse engineering approach that can resolve the above stated issues. The mapping from extracted clusters to meta-model of our component model X-MAN [10] and working of our tool has already been explained in [4] (white boxes in Figure 1). In this paper, we explain how we: 1) Capture the object-oriented source code. 2) Map the captured notation to X-MAN clusters based on our rule base, by stating basic scenarios (red boxes in Figure 1). Section II of this paper compares our approach with other approaches that extract explicit architecture. Section III explains X-MAN component model. Section IV presents our approach using an example. Section V include conclusion and future work.

## 2. Related Work

There are quite a few approaches that follow szyperski's definition of components for reverse engineering.

*JAVACompExt* [3] is a heuristic based approach that extracts Abstract Data Type (ADT) components. The approach by Antoun *et al.* [1] re-engineers Java code into ArchJava components. Chouambe *et al.* [7] produces composite components from Java source code. Pattern-based Reverse Engineering of Design Components [12] extracts design components based on the structural descriptions of design patterns. A Reverse Engineering Approach to Subsystem Structure Identification [14] re-structures the system into a hierarchy of subsystems along with their high-level abstract representation as components. Washizaki [16] detects reusable part of object-oriented classes and transforms classes into JavaBeans components. *Archimatrix* [8] reconstructs the architecture in form of components from the source code after removing design deficiencies. Quality centric approach [11] focuses on quality of explicit interfaces by following a semantic-correctness model. Alshara *et al.* [2] extracts OSGi or SOFA components from object-oriented code. Components extraction in memory-constrained environments [16] identifies reusable part of an

<sup>4</sup>The term re-deposit-ability means ability to re-deposit the composed components after retrieval for future reuse.

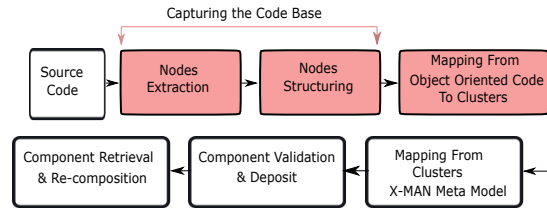


Figure 1: RX-MAN

Approach	Re-Composition	Repository Deposit	Componentization Independency	Automated	Component Model
JAVACompExt	✗	✗	✗	✓	UML
Antoun et al.	✗	✗	✓	✗	ArchJava
Design Components	✗	✗	✓	✓	UML
Subsystem Structure Identification	✗	✗	✓	✓	ADL
Chouambe et al.	✗	✓	✗	✓	EJB
Washizaki	✗	✓	✓	✓	JavaBeans
Archimatrix	✗	✗	✓	✗	ADL
Alshara et al.	✗	✓	✓	✗	SOFA
Quality Centric	✗	✗	✓	✓	ADL
Memory Constrained..	✗	✓	✓	✓	JavaBeans
RX-MAN	✓	✓	✓	✓	X-MAN

Table 1: Approaches based on Explicit Components

object oriented code and refactors the relative or surrounded code to reuse the identified part.

Few major shortcomings with these approaches are lack of automation, inability to retrieve from repository and inability to achieve code-independent composition of the extracted components. These approaches however, extract explicitly defined components with required and provided services. In Table 1, attribute *Repository Deposit* means whether an approach is based on a component model that supports repository or not. *JAVACompExt*, Antoun's approach and Design Components are based on component models that do not support repository whereas, Subsystem Structure Identification, Archimatrix and Quality centric approach do not define or discuss the deposition of components via a repository. Lack of repository decreases reusability as components cannot be configured and preserved for retrieval. The attribute *Automated* shows whether an approach is automated or needs manual assistance. *Component Model* shows the component model that is followed for extraction of components. *Componentization independency* shows whether an approach is only applicable on source systems that are designed as separate packages.

Out of all the explicit approaches, our approach (that we call RX-MAN) is the only one that: supports component repository as part of its implementation, does not need code-level configurations for reuse, is automated, does not restrict to bind all the ports of a component being reused and supports composition of the re-deposited components. A well-known framework partially relevant to our approach is MoDisco [6]. MoDisco uses Architecture-driven modernization (ADM) to construct the Knowledge Discovery Meta-model (KDM). The core difference is that our approach aims for a specific meta-model (X-MAN) as a transformation model whereas, MoDisco aims for a cus-

tomised meta-models based on legacy technology and requirements<sup>5</sup>.

### 3. X-MAN Component Model

Unlike ADL-based components, X-MAN component model is based on encapsulation i.e., an X-MAN component only has provided methods and no required ones. An atomic X-MAN component consists of a computation unit that has the implementation of methods and exposed functionality of specific methods (the methods that can be selected for composition). Methods are exposed as interfaces which are implemented in the computation unit. Any exposed method can be selected before instantiating a component and method's inputs and outputs can be used with the exposed methods of other X-MAN components. Computation only takes place in a computation unit, which is why this component model is encapsulated [13].

In case of a composite component, encapsulation is preserved by composition because a composite component consists of two or more atomic components composed together by composition connectors. Composition connectors in X-MAN are control structures that direct the route of execution. *Sequencer (SEQ)* composition connector provides sequencing of execution between two or more than two components and *Selector (SEL)* provides branching based on specific conditions<sup>6</sup>. If two components A and B will be instantiated with one exposed method each and composed by a sequencer, then there will be only two methods that will be involved in this composition. Basic semantics of X-MAN component model are shown in Figure 2 (lollipop in the Figure is a notation used to show the presence of exposed methods). One computation unit cannot interact with other units directly but only via composition connectors. Control of the components exists outside of computation units and that is why one does not need code-level configurations to reuse the components. Any component can be reused by composing it with others using appropriate composition connector [13] and exposed methods.

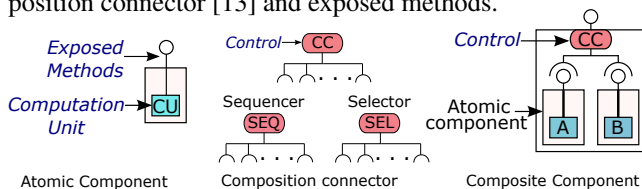


Figure 2: X-MAN Component Model

In ADL-based component models, control cannot be separated from computation and therefore, one needs code-level configurations to recompose required and provided

<sup>5</sup>The implementation of our approach and MoDisco uses many common frameworks e.g., Ecore, eclipse modelling framework (EMF) etc.

<sup>6</sup>With *SEQ* (sequencing), *SEL* (branching) and *LOOP* (looping), X-MAN is Turing complete.

```
public class A {
    public int provideSpeed(int speed)
    {speed=100; this .returnSpeed(speed);}
    public int returnSpeed(int topSpeed){ this .saveInLog(topSpeed); return
    topSpeed;}
    private void saveInLog(int value){ System.out. println ("Value is
    saved");}}
```

Figure 3: Scenario 1

```
package com.A.scenario;
import java . util .*;
public interface A {
    public int provideSpeed ( int speed);
    public int returnSpeed ( int topSpeed);}

package com.A.scenario;
import java . util .*;
public class Aimpl implements A {public Aimpl() {}
    public int provideSpeed ( int speed) {speed = 100; return speed;}
    public int returnSpeed ( int topSpeed){ this .saveInLog(topSpeed);
    return topSpeed;}
    private void saveInLog(int value){ System.out. println ("Value is
    saved");}}
```

Figure 4: Mapped Code from Scenario 1

services. X-MAN component model also supports re-deposition of components after composition and composed integrations of components can be retrieved for future reuse. One does not need to bind all the ports at code level like ADLs but only need to select the exposed methods and appropriate composition connector for a composition.

### 4. Our Approach: RX-MAN

This section uses an example of *Brake Control System* to demonstrate the code capturing and mapping of code from object-oriented classes to X-MAN clusters. Before showing the mapping in terms of an example, section below demonstrates few basic scenarios to show the rules of mapping.

#### 4.1. Mapping from Source Code to X-MAN clusters

In RX-MAN, each input is a class, bunch of classes or a program (object-oriented code base). The input is mapped using a rule base against defined scenarios and output is one or more than one X-MAN components. The mapping is based on interactions and invocations of methods. Below are few basic scenarios to show the mapping rule base.

**1) Single Class with no interaction:** A single non-interactive class is the most trivial scenario in RX-MAN. In this scenario, all the methods that call each other belong to the same class. Output of this scenario would be one X-MAN component with all the methods of the class mapped to computation unit. Figure 3 is showing a non interactive single Java class. In Figure 3, methods *provideSpeed*

```
public class A {
    B obj= new B();
    public void provideSpeed(int speed){speed=100;
    obj .evaluateSpeed(speed);}
    public class B{
    public int evaluateSpeed (int topSpeed){int maxSpeed=200; return
    maxSpeed-topSpeed;}}
```

Figure 5: Scenario 2

---

```

public class A{
    B obj= new B();
    public void provideSpeed(int speed){speed=100;
        obj.evaluateSpeed(speed);}
public class B{
    public int evaluateSpeed(int topSpeed){int maxSpeed=200; int
        recordSpeed=maxSpeed-topSpeed; this.saveInLog(recordSpeed);
        return recordSpeed;}
    private void saveInLog(int recordValue){system.out.println("Value
        Logged Successfully"); }}

```

---

Figure 6: Scenario 3

and *returnSpeed* are marked as exposed methods. Method *saveInLog* is in the computation unit along with other two methods but it cannot be used as an exposed method because its modifier is private. The exposed methods of this component are mapped as an interface and computation unit has implementation of all the methods. Figure 4 shows the notation of mapped code of scenario 1 (inside X-MAN component). Figure 7(a) shows the notation of X-MAN component mapped from this scenario. Red boxes in Figure 7(a) shows the exposed functionality of this component i.e. exposed methods.

**2) Two Classes with public-public methods interaction:** Next possible scenario is the interaction of two Java classes in a code base. As our approach is based on interaction and invocation of methods, modifiers of methods play an important role in defining a scenario. Figure 5 is showing an example of two Java classes that interact with each other via methods with public modifiers. In this scenario, output would be just one X-MAN component. All the callers would be placed in one computation unit along with the methods they called. If a method M is in invocation list of more than one methods, it would be placed in the computation unit only once to avoid redundancy. This scenario assumes that all the interactions are between public methods and no method is neither invoking any private method nor dealing with any private class level variable. Figure 7(b) is showing the X-MAN component mapped for two Java classes of scenario 2<sup>7</sup>.

**3) Two Classes with private-public OR public-private methods interaction:** This scenario has more possible outcomes than the previous two. If a private or a public method in Class A calls a public method in Class B, there are following possible scenarios.

1) Public method in Class B is neither accessing any private variable of the class nor it is calling any private method of B. In this case, such public method will be placed along with its caller in the same computation unit.

2) If method in Class B uses private variable of Class B or it calls some other private method of B, it cannot be simply placed with its caller. In this case:

a) If caller is private, the public method of B will be placed in both components (computation unit of A and computation unit of B as its dealing with private entities of both

---

<sup>7</sup>In case of void methods, output of an exposed method is boolean that indicates termination of execution of that method

classes).

b) If caller is public, public method of B will only be part of computation unit of B. Its caller can access it using composition connector or by data input/output of an exposed method.

Figure 6 shows a scenario of public-private case. Method *provideSpeed* of Class A has method *evaluateSpeed* of Class B in its invocation list. Method *evaluateSpeed* is accessing private method *saveInLog* of Class B. Output of this scenario would be two X-MAN components. One component would have one method i.e. *provideSpeed*. The other component would have *evaluateSpeed* and *saveInLog* in its computation unit and *evaluateSpeed* would be the exposed method of second component. Figure 7(c) shows two components mapped from scenario 3. The purpose of explaining the above scenarios is to provide comprehension of the basic mapping mechanism. Clustering of methods based on their invocations and modifiers provide much better cohesion as only those methods would belong to same component that are associated and have loose coupling with rest of the components. To apply these scenarios on a full code base, one needs an appropriate notation that can capture the whole legacy code and preserve the relation and dependencies among all the entities. To capture the code base, our approach uses a customised parser that is written specifically for RX-MAN.

## 4.2. Capturing the Code Base

The customised parser used in this approach is based on Abstract Syntax Tree (AST) parser. The designed parser is more powerful than the default AST parser as it also extracts and maps invocation nodes from each method node in the code base. If a method A invokes method B, and method B invokes method C then our parser extracts and connects all nodes of the method C to method A as both are indirectly connected by method B. AST parser extracts one big tree of nodes from a code base in which all the nodes are connected hierarchically e.g., starting node would be compilation unit (class level or package level) connected with its sub nodes i.e., class declarations, class variables etc. Each class declaration node is further connected to its method nodes and each method node is connected with its sub nodes. This hierarchy of nodes goes till the last level which is simple name nodes i.e., name of local variables etc.

It is impossible to trace and cluster the chain of all possible method interactions and invocations from this one big complex tree. Therefore, RX-MAN parser indexes each method of the code base and connect all associated nodes with every method. Figure 8 is showing the extraction of nodes using RX-MAN parser. Each method node index has information about its parent class, parent package and class variable this method uses. Along with this information, each method node index is connected with all the method

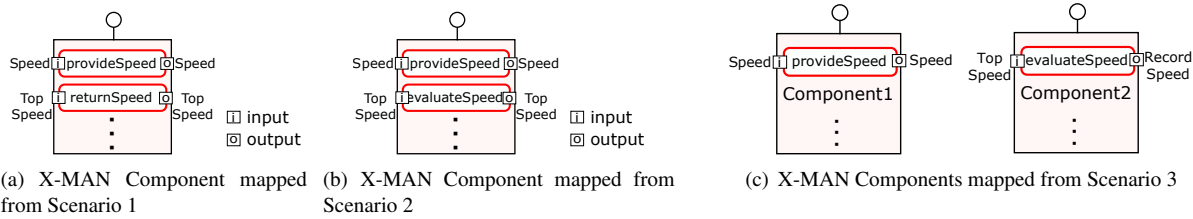


Figure 7: X-MAN: Components Mapped From Scenarios

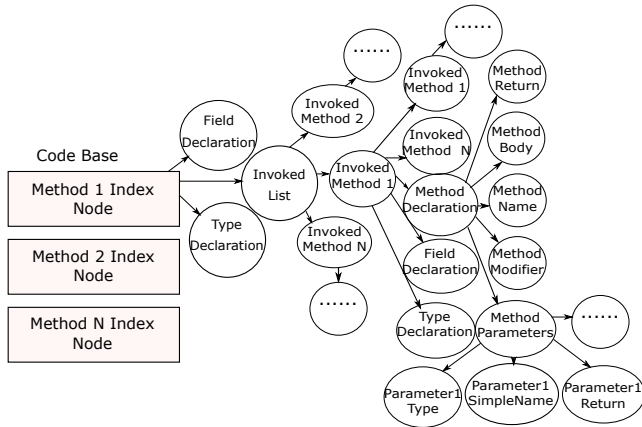


Figure 8: RX-MAN Parser

```

package vehicle . control . speedcontrol ;
import vehicle . brakecontrol . BrakeControl;
public class SpeedMonitoring {
    BrakeControl obj;
    public SpeedMonitoring(){}
    public double collisionTimeCalculator (double speed, double distance )
    {speedMonitoringValue(distance /speed);
    return distance /speed;}
    public boolean speedMonitoringValue(double collisionEstimate )
    {boolean time=false;
    if ( collisionEstimate <15)
    {time=true; obj. collisionParametersActivation (time);}
    else {obj. collisionParametersActivation (time);} return time;}}

package vehicle . brakecontrol ;
public class BrakeControl {
    public BrakeControl(){}
    public boolean collisionParametersActivation (boolean flag ){
    if ( flag ==true){BrakeSystemActivation (flag );}
    else {TimeTriggerValue(flag );} return flag ;}
    private void BrakeSystemActivation(boolean value){
    System.out. println ("Brakes Applied");}
    public boolean TimeTriggerValue(boolean value){return value;}

```

Figure 9: Brake Control System

it invokes directly or indirectly. This mapping makes sure that no indirect invocation goes undetected. In short, starting from each method in a code base, each method node index is connected with whole chain of invocations it causes in a code base (Figure 8). Therefore, each cluster of RX-MAN is consisted of restructured associated nodes based on rules of method's interactions and invocations.

### 4.3. Example: Brake Control System

Fig 9 shows a simple example of brake control system that is reverse engineered using our approach. In the given example, there are two classes. Class *SpeedMonitoring* has methods *collisionTimeCalculator* (for calculating time till collision) and *speedMonitoringValue* (for automatic brake

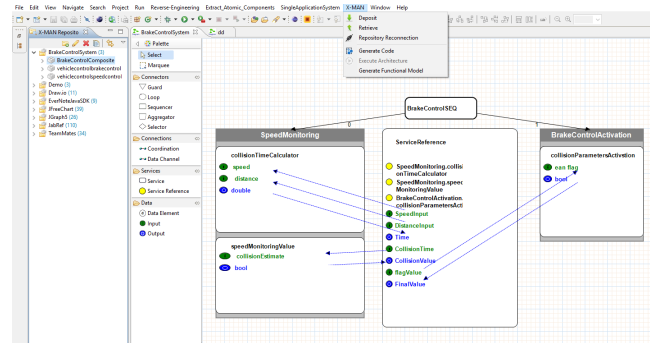


Figure 10: Deposition of A Composite Component mode if time till collision is less than 15 seconds). Method *speedMonitoringValue* invokes *collisionParametersActivation* from Class *BrakeControl*. Depending on the value of time, method *collisionParametersActivation* either invokes *BrakeSystemActivation* or *TimeTriggerValue*. According to our approach, method *speedMonitoringValue* has one method node against its invocation node i.e., *collisionParametersActivation* and method *collisionParametersActivation* has two method nodes against its invocation node i.e., *BrakeSystemActivation* and *TimeTriggerValue* (hence two indirect invocation nodes against *speedMonitoringValue*). As the method *BrakeSystemActivation* is private (scenario 3) therefore, the approach will map the whole code to two clusters.

RX-MAN tool maps these clusters to X-MAN meta-model and extracts two components. First cluster has methods *speedMonitoringValue* and *collisionTimeCalculator* (both will be mapped as exposed methods of an X-MAN component). Second cluster has methods *collisionParametersActivation*, *BrakeSystemActivation* and *TimeTriggerValue* (from this cluster method *BrakeSystemActivation* cannot be mapped as an exposed method as it is private).

Fig 10 is showing a possible case of composition of RX-MAN using a composition connector sequencer (SEQ). *SpeedMonitoring* component (extracted from first cluster) will be triggered first as this route has 0 (lower number means higher priority) and component *BrakeControlActivation* (extracted from second cluster) will be triggered after that. It is one valid case of composition as the component *BrakeControlActivation* will perform its execution after getting *collisionValue* from component *SpeedMonitoring*. Fig 10 is also showing that this composite component has been deposited in the *BrakeControlSystem* (X-MAN repository at left) and can be instantiated in future to be reused or re-

Code Base	Java Classes	X-MAN Components	Abstraction Ratio	Total Methods	Processing Time of RX-MAN
Draw.io	<b>101</b>	<b>11</b>	<b>9%</b>	<b>892</b>	<b>06 Secs</b>
EverNote SDK	<b>106</b>	<b>09</b>	<b>12%</b>	<b>3910</b>	<b>08 Mins, 11 Secs</b>
JabRef	<b>935</b>	<b>110</b>	<b>8.5%</b>	<b>6434</b>	<b>06 Mins, 28 Secs</b>
JFree Chart	<b>993</b>	<b>39</b>	<b>25.46%</b>	<b>10274</b>	<b>20 Mins, 43 Secs</b>
JGraph5	<b>171</b>	<b>26</b>	<b>7%</b>	<b>3482</b>	<b>39 Secs</b>
TeamMates	<b>815</b>	<b>34</b>	<b>23.97%</b>	<b>7519</b>	<b>05 Mins, 76 Secs</b>

Figure 11: Evaluation Cases

composed further.

The proposed approach has been applied on six legacy code basis, available for empirical evaluation at *GitHub* and *Quality Corpus*. Figure 11<sup>8</sup> is showing the summarisation of results, obtained by our approach.

## 5. Discussion and Conclusion

This paper presents two important steps of our approach: code capturing and mapping from object-oriented code to X-MAN clusters. We also demonstrated an example of Brake Control System and show a valid case of composition in our tool.

The biggest threat to validity of RX-MAN is the lack of consideration to important relations in an object-oriented language e.g., aggregation, composition and inheritance etc. These relations, if mapped, can provide much better cohesion and hence better re-usability. Future work includes expanding this approach beyond interactions of methods to map control statements in the code (*if*, *switch*, *loops* etc.) to composition connectors of X-MAN. To the best of our knowledge, ours is the only approach that can reuse and compose the extracted components without any code-level configurations with ability of re-deposition of components.

## References

- [1] Marwan Abi-Antoun, Jonathan Aldrich, and Wesley Coelho. A case study in re-engineering to enforce architectural control flow and data sharing. *Journal of Systems and Software*, 80(2):240–264, 2007.
- [2] Zakarea Al-Shara, Abdelhak-Djamel Seriai, Chouki Tibermacine, Hinde Lilia Bouziane, Christophe Dony, and Anas Shatnawi. Materializing architecture recovered from oo source code in component-based languages. In *ECSA: European Conference on Software Architecture*, 2016.
- [3] Nicolas Anquetil, Jean-Claude Royer, Pascal Andre, Gilles Ardourel, Petr Hnetyinka, Tomas Poch, Dragos Petrascu, and Vladliela Petrascu. Javacompext: Extracting architectural elements from java source code. In *Reverse Engineering, 2009. WCRE'09. 16th Working Conference on*, pages 317–318. IEEE, 2009.
- [4] R. Arshad and K.-K. Lau. Extracting executable architecture from legacy code using static reverse engi-

<sup>8</sup>Abstraction Ratio means average size of components in terms of code classes

- neering. In *Proceedings of Twelfth International Conference on Software Engineering Advances*, pages 55–59. IARIA, 2017.
- [5] K. Bennett. Legacy systems: coping with success. *IEEE Software*, 12(1):19–23, Jan 1995.
- [6] Hugo Bruneliere, Jordi Cabot, Grégoire Dupé, and Frédéric Madiot. Modisco: A model driven reverse engineering framework. *Information and Software Technology*, 56(8):1012–1032, 2014.
- [7] Landry Chouambe, Benjamin Klatt, and Klaus Krogmann. Reverse engineering software-models of component-based systems. In *Software Maintenance and Reengineering, 2008. CSMR 2008. 12th European Conference on*, pages 93–102. IEEE, 2008.
- [8] Markus Detten, Marie Christin Platenius, and Stefan Becker. Reengineering component-based software systems with archimetric. *Softw. Syst. Model.*, 13(4):1239–1268, October 2014.
- [9] J. M. Favre, F. Duclos, J. Estublier, R. Sanlaville, and J. J. Auffret. Reverse engineering a large component-based software product. In *Proceedings Fifth European Conference on Software Maintenance and Reengineering*, pages 95–104, 2001.
- [10] Nannan He, Daniel Kroening, Thomas Wahl, Kung-Kiu Lau, Faris Taweel, C Tran, Philipp Rümmer, and S Sharma. Component-based design and verification in X-MAN. *Proc. Embedded Real Time Software and Systems*, 2012.
- [11] S. Kebir, A. D. Seriai, S. Chardigny, and A. Chaoui. Quality-centric approach for software component identification from object-oriented code. In *2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture*, pages 181–190, Aug 2012.
- [12] Rudolf K Keller, Reinhard Schauer, Sébastien Robertaille, and Patrick Pagé. Pattern-based reverse-engineering of design components. In *Proceedings of the 21st international conference on Software engineering*, pages 226–235. ACM, 1999.
- [13] Lau Kung-kiu et al. *An Introduction To Component-based Software Development*, volume 3. World Scientific, 2017.
- [14] Hausi A Müller, Mehmet A Orgun, Scott R Tilley, and James S Uhl. A reverse-engineering approach to subsystem structure identification. *Journal of Software: Evolution and Process*, 5(4):181–204, 1993.
- [15] Clemens Szyperski. *Component software: beyond object-oriented programming*. Pearson Education, 2002.
- [16] Hironori Washizaki and Yoshiaki Fukazawa. Extracting components from object-oriented programs for reuse in memory-constrained environments.



# Leveraging the Power of Component-based Development for Front-End Components: Insights from a Study of React Applications

Chen YANG\*, Yan LIU✉†, Yiwei LIN§  
School of Software Engineering, Tongji University  
Shanghai, China

Email: \*1610833@tongji.edu.cn, †yanliu.sse@tongji.edu.cn, §1532790@tongji.edu.cn

Jia Yu‡  
SEEBURGER China Inc.  
Shanghai, China  
Email: ‡j.yu@seeburger.com

**Abstract**—Classic design patterns, architectural styles, and design principles have been introduced and enhanced in Web front-end development. Recently, component-based architecture, successfully introduced in React.js, has tended to replace MVC and other MV\* patterns in front-end frameworks. However, we still know little about design strategies for leveraging the power of component-based development. We conducted a study to explore the use of components in React-based applications from two levels. Three private repositories were analyzed to get practical insights into the nature, limitations and potentials of CBD for front-end implementations. Our research started with an aerial view, where we examined the dependency, connectivity, and overall of components. Quite different architectural and programming styles were observed; these can be easily attributed to the lack of front-end component design paradigms. Meanwhile, all cases exhibit similar component connectivity and dependency patterns, which enlighten the study to categorize them further. Next, the study zoomed in on the architectural elements level, where we classified front-end components into four categories. Our observations suggest that design components on the architectural elements level may dramatically boost the power of component-based front-end development.

**Keywords**—Component-Based Development, Web Front-End, React Framework, Case Study

## I. INTRODUCTION

Along with the development of Internet and cloud computing, business systems form a unique web-based development style. Due to the plethora of applications served by JavaScript and varieties of programming needs, JavaScript Frameworks have been developed to facilitate the work of Web programmers[1]. The use of React for building web applications has given rise to the popularization of Component-Based Development (CBD) for front-end development. React is mainly applied to developing applications with declarative views and composable components[2]. Owing to the differences between programming languages, scenarios, and implementation method, front-end CBD has a unique nature. However, few studies have been done in this area[6]. In this paper, we explored the implementations of CBD for front-end, and the discoveries stimulated us to categorize front-

end components. We hope that our findings can lead to the potential improvements in CBD.

This paper makes following contributions: (1) Creation of an experimental exploration process for front-end CBD from a code observation view. (2) Derivation of significant discoveries, such as *orphan components*, connectivity, and dependency patterns from the exploration process. (3) Tentative proposal of a four-category classification of front-end components, mapping of categories to the selected cases, and confirmation of the rationality of the classification.

The paper is organized as follows. In Section II, the scope will be introduced. Section III illustrates the exploration process. In Section IV and Section V, the different levels in the exploration process will be elaborated. In Section VI, the progress in CBD and front-end development will be traced. The conclusion and future work are in Section VII.

## II. RESEARCH SCOPE

### A. Case Study

CBD for front-ends is at the initial stage, and it's obviously beneficial to explore an old theory under a new situation by doing a case study. Just like follows, we started with selecting independent and complete repositories which can reflect real corporate situations. Then we proceeded to preprocess, where a Javascript analytical tool chain was integrated to extract information from the source code. Besides, same observation views and priorities were applied to the three selected repositories. Next, we continued our exploration to get insights from two levels. Discoveries and conclusions were reached throughout the whole process.

### B. Study Scope

To carry out the exploration, we used repositories from private corporate software environments as cases. Repositories released as open source are normally libraries or projects created by individuals. Component libraries act as the basis of other applications, yet few relations can be extracted from these loose components. Elementary, exploratory, or instructional repositories are unable to reflect the stressful, complicated, and systematic corporative application scenarios.

Three private repositories with complete processes were selected, involving different groups; all utilize React to achieve the effect of CBD. These repositories all possess high separability and use AJAX to handle interactions with the back-end through standard JSON-based REST API. Therefore, these repositories are fully equivalent. Discoveries of our case study did not interfere with the programmers behavior since we commenced our research after the repositories had developed.

A brief description of the selected repositories is presented in Table I. React is the *View* layer in *MVC* patterns, so we simply focused on the visible components. Thus only the JS/JSX source code were extracted, while CSS and configuration were not taken into account. Besides, a React-based Web application relies on many different kinds of third-party libraries; we merely focused on libraries related to *View*, paying no attention to those related to language, framework, etc.

TABLE I  
OVERVIEW OF CASE PROJECTS

Project agname	LOC	Start - End date	Duration (week)	Contributors	Third-party library/View	Usage Scenario
BD	6,929	2016.12.26-2017.05.22	21	6	8	life service provider
MY	6,523	2016.10.24-2017.03.08	19	6	14	financial management
FE	26,807	2016.08.07-2016.11.08	13	3	9	fund analysis

In order to guarantee the validity of results, we integrated a Javascript analytical tool chain. Firstly, due to the incompatibility between the ES6 in React-based projects and *Esprima*<sup>1</sup>, the raw source code without irrelevant files was compiled into the ES5 format via *Babel*<sup>2</sup>. By *Estraverse*<sup>3</sup>, we traversed, extracted, and persisted the preliminary data, including components, relations, properties, functions from the abstract syntax tree generated by *Esprima*. Lastly, we combined or disassembled this information and converted it into an understandable visualization.

### III. EXPLORATION PROCESS

The theory of CBD has become mature after 50 years of back-end development[4]. However, little is known about the design strategies for harnessing CBD capabilities for front-end development. So, we created an exploration process in an attempt to figure out the current situation of front-end CBD. Fig.1 illustrates the basic workflow of our exploration process from three key phases. Progressive relationship exists between the phases, and they also influence each other conversely. In this paper, we summarize the major conclusions in the second phase due to space limitations. A brief description of the first phase are given to pave the way for the second. Although the elaboration of the *component timeline* isn't presented in this paper, evidence suggests that it deserves further investigation.

The process starts with global features at the aerial view. We extracted components and relations from source code and got observations from three different aspects. Above all, we did an overall observation. Afterwards, certain features

(see Section IV) are selected to analyze the state or extent of all components connected in *connectivity* aspect. Finally, regarding the *dependency* aspect, there are many remarkable features associated with the degree of *dependency* between one component and others. Significant observations on the aerial view provided the basis for launching a thorough analysis.

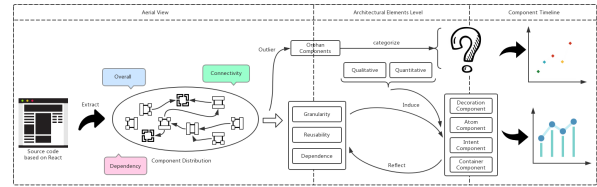


Fig. 1. Exploration Process

Meaningful discoveries could be derived from the aerial view, which we generalized into three aspects. Firstly *Granularity* refers to the encapsulation capability of components, specifically the number of attributes and functions. We also made discoveries about *Reusability*, like the reuse degree and component reuse pattern. Valuable relation features were obtained from the *Dependence* among components. Besides, abnormal discoveries like orphan components and groups were also made. Further analyzing the timeline of orphan components can help us investigate the accumulation process.

With the help of external features emerging from the *aerial view*, we proceeded with our exploration at the *architectural elements level*, where we focused on the nature of one component itself. We mapped components into four categories considering their typical usage scenarios, and examples were presented to facilitate the readers understanding. By attaching internal qualitative and quantitative features, each component was categorized uniquely according to its distinctive nature. Similarly, these categories also reflect external features.

### IV. OBSERVATIONS FROM THE AERIAL VIEW

#### A. Feature Selection and Case Exhibition

To get a better idea of the aerial view, a brief summary of the key features derived from these aspects is shown in Table II, and important attributes that represent the key features quantitatively are listed too. Statistical features such as scale (Table II, 1.1) are shown directly. Graphs in Fig.2 are created to facilitate the understanding of the remaining features that cannot be captured in a numerical form. Moreover, attributes like Scale and Code Clone were calculated using SonarQube<sup>4</sup>, and attributes like Maximum Path were gained by combining or disassembling the basic information extracted in Section II.B. The dependency graphs are drawn by Gephi<sup>5</sup>. Finally, Table III shows the observations and explanations of the cases.

The dependency graphs describe the state of components and relations from a global perspective. As shown in Fig.3, the internal invocation structure of the project can be regarded as

<sup>1</sup>Esprima: <http://esprima.org/>

<sup>2</sup>Babel: <http://babeljs.io/>

<sup>3</sup>Estraverse: <https://github.com/estools/estraverse>

<sup>4</sup>SonarQube: <https://www.sonarqube.org/>

<sup>5</sup>Gephi: <https://gephi.org/>

graphs, where nodes represent components and edges represent relations between them. The degree of a node in graphs is the number of edges attached to the given node. The nodes in gray, red, blue, and green respectively represent the ordinary, leading indegree, leading outdegree, and leading degree nodes. The size and shade of an edge indicate the weight of it, determined by the number of relations between one component and others. In addition, these graphs are directed graphs, where the direction of an arrow denotes the callee.

TABLE II  
ATTRIBUTES DERIVED FROM FEATURES

Aspect	Key Feature	Attribute	BD	MY	FE
1.Overall	1.1. Scale	Scale	6929	6523	26807
	1.2. Code Clone	Code Clone	7.6%	10.7%	24.1%
	1.3. Components and Relations Summary	Components Relations	95 122	153 241	76 114
2.Connectivity	2.1. Global Relations Pattern	Maximum Path	3	4	3
		Average Path Length	1.355	1.482	1.371
		Maximum Relation Weight	3	9	3
		Relation Weight Distribution	As shown in Fig. 2. (a)		
2.2. Disconnected Components	Disconnected Set	4	7	2	
	Disconnected Components	17/15.2%	23/13.1%	14/15.6%	
3.Dependency	3.1. Relation Strength	Average Degree	2.568	3.15	3
		Maximum Indegree	5	15	16
		Maximum Outdegree	33	8	8
		Average Weighted Degree	1.337	1.98	1.671
		Maximum Weighted Indegree	5	29	16
	Maximum Weighted Outdegree	33	11	10	
	3.2. Relations Distribution	Indegree Distribution	As shown in Fig. 2. (b)		
Outdegree Distribution		As shown in Fig. 2. (c)			
Weighted Indegree Distribution		As shown in Fig. 2. (d)			
Weighted Outdegree Distribution		As shown in Fig. 2. (e)			

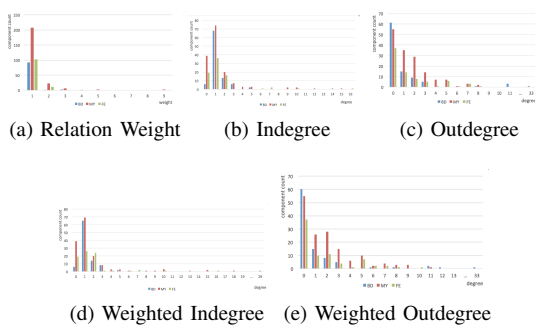


Fig. 2. Attributes Distribution

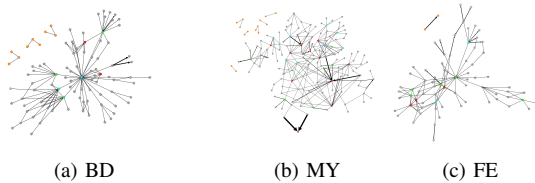


Fig. 3. Dependency Graphs

### B. Discovery

In light of the observations, our preliminary discoveries regarding the three aspects are as follows: **Granularity:** (i) Components with fine granularity tend to possess a higher

TABLE III  
OBSERVATIONS FROM THREE ASPECTS

Aspect	Observations	Explanations / Possibilities
Overall	An asymmetry exists between the number of components & relations and the scale of the respective project.	Different coding styles may account for this phenomenon, <i>FE</i> exhibits a lack of decomposition, while <i>MY</i> tended to be overencapsulated.
	A project with larger components ( <i>FE</i> ) has the highest code clone rate (24.1%).	Similar tiny functions are implemented in the form of code clones between components.
Connectivity	Complex multi-layer nesting rarely exists.	Nestings of components in React come at a cost.
	The relation weight of the dependency graphs are generally very small.	Cannot determine if a higher relation weight is beneficial, yet the callee component with larger relation weight (9), importing third-party components directly with nothing attached, can be considered as a kind of over-encapsulation.
	<i>Orphan component</i> (a component that is independent of the primary components set) and <i>group</i> exist with a fairly high ratio (14%).	Reasons such as code examples, strategy, and requirements changes led to the legacy of orphan groups
Dependency	Components with smaller degrees account for the majority; few differences exist between distributions with weight and those without.	Simple dependency is dominant.
	Different top-ranked in-degree components perform different functions; Over-encapsulation exists in components with an extremely high in-degree.	Components with a higher in-degree tend to either be (1) basic components applied in different pages, or (2) relatively complex components implemented a specific function.
	Different top-ranked out-degree components show different features.	Components with a higher out-degree tend to be (1) page-level components that invoke many other components; or (2) normal components that invoke over-encapsulation components.

in-degree; (ii) components with rough granularity tend to possess a higher out-degree; (iii) a potential correlation may exist between granularity and code clones. **Reusability:** (i) Reusability didn't achieve the optimal effect; (ii) the over-encapsulation phenomenon may affect the reusability of components. **Dependence:**(i) Simple dependency plays a major role; (ii) orphan components deserve more discussion.

### V. INSIGHTS FROM THE ARCHITECTURAL ELEMENTS LEVEL

In the exploration of the *aerial view*, quite different architectural and programming styles were observed. These can be easily attributed to the lack of front-end component design paradigms, which motivated us to perform a thorough analysis.

For existing front-end development, efforts have been made regarding different types of business logics encapsulation. For example, React is regarded as a presentation (view) layer, and often used with Redux, which is separated into an action (controller) layer and a reducer (model) layer as well, to interact with interfaces[3]. The appearance of stateless components in React also indicates that components in front-end gradually perform their respective roles. So that a layered architecture for front-end is one possible way to improve development by providing more controllability. According to programming experiences and design specifications for building components, we tentatively propose a classification methodology.

#### A. A Four-Category Classification of Front-End Components

We propose a four-category classification of components for React-based applications in Table IV considering typical usage scenarios. As space is limited, a graphical example to help readers understand the component categories is available online<sup>6</sup>. It's strongly recommended for readers who want to understand the usage scenarios and code features of different categories. Reasonable use of the four-category classification may have a positive effect on good design.

<sup>6</sup>Example: <https://github.com/Ada12/RCCE/blob/master/example.md>

TABLE IV  
CLASSIFICATION OF THE COMPONENTS

Component	Definitions
<b>Decoration Component</b>	A <i>Decoration Component</i> is a tiny component that is only responsible for decorative functionalities with extreme dependence upon parent components, and pays no attention to its own state and lifecycle.
<b>Atom Component</b>	An <i>Atom Component</i> is a basic component that tends to be an inseparable unit of functionality with complete lifecycle management. It can be a constituent of other more complicated components.
<b>Intent Component</b>	An <i>Intent Component</i> is a more complex component, which can be thought of as the glue between components. It implements a complete presentational business process, and can be composed of <i>Decoration Components</i> and <i>Atom Components</i> .
<b>Container Component</b>	A <i>Container Component</i> is a page-level component that manages states exposed by subcomponents unifiedly, and composed of the components mentioned above. It's responsible for communications between different subcomponents and interacted with external interfaces.

B. Findings from the Case Study

In order to observe the categories in real-world React-based projects, we continued the analysis of our cases. Although we mapped the categories according to usage scenarios and qualitative features, definite quantitative features should still be used as benchmarks. In addition, the preliminary discoveries summed up in Section IV were also taken into account. Table V lists the features we utilized to categorize the components. Table VI shows the mapping of components and features, in which the symbol  $\checkmark$  represents one component possesses a given feature, and symbol  $\uparrow$  or  $\downarrow$  represents the component possesses a larger or smaller value of a given feature.

TABLE V  
SUMMARY OF BASIC FEATURES

Observation Aspect	Quantitative	Qualitative	Discoveries
Key Element	1.1 state; 1.2 props;	2.1 usage scenarios	3.1 indegree
	1.3 invoke third-party library	2.2 business coupling	3.2 outdegree
	1.4 invoke other component	2.3 scale	3.3 reusability
	1.5 life cycle management		
	1.6 interact with interfaces		

TABLE VI  
MAPPING OF COMPONENT AND FEATURES

Component	Features									
	1.1	1.2	1.3	1.4	1.5	1.6	2.3	3.1	3.2	3.3
Decoration		$\checkmark$					$\downarrow$	$\uparrow$	$\downarrow$	$\uparrow$
Atom	$\checkmark$	$\checkmark$	$\checkmark$		$\checkmark$			$\downarrow$	$\uparrow$	$\uparrow$
Intent	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$			$\uparrow$	$\downarrow$	$\uparrow$
Container	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$\uparrow$	$\downarrow$	$\uparrow$	$\downarrow$

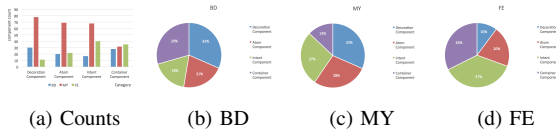


Fig. 4. Component Distribution by Category

By the category criteria of Table V and VI, we categorized the three cases manually; the results are shown in Fig.4. We found that (1) project ('FE') which possesses a lack of decomposition and a higher code clone rate, has less *Decoration* and *Atom Components*, due to the un reusable tiny functions; (2) project ('BD') with high quality demands of user experience tend to have more *Decoration Components*; (3) more *Intent Components* exist in project ('MY') that focuses a great deal on business process.

C. Conclusion

In conclusion: (1) Significant differences exist between the various component categories; (2) different categories can represent specific usage scenarios; (3) a slight variation in distribution exists for categories under different scenarios.

VI. RELATED WORK

It has been fifty years since software components was firstly proposed[4]. CBD developed rapidly and gradually formed its own philosophy. Alan[5] indicated that a component is a deliverable piece of functionality that can independently provide interface access to its services. Tassio[6] investigated 1231 studies dating from 1984 to 2012, and summarized the domains where CBSE has been applied; yet most of them were related to the server side, only one showed solicitude for providing better services for the front-end. These all indicate that little research has been done on CBD for front-ends.

Based on the keyword JavaScript, we explored papers in 15 conferences in the field of SE over past five years. 48 papers involving a variety of different areas, matched the condition. However, none of them pay close attention to the field of CBD for front-end development. In addition, some papers explored the related area. Cappiello[7] defined a quality model for building blocks of mashup applications. Magnusson[8] found that a certain framework and the way to implement its data flow pattern are the main reasons that impact maintainability.

VII. CONCLUSION AND FUTURE WORK

In this paper, we carried out an exploration tentatively, and the constant outpouring of discoveries prompt us to propose a four-category classification of front-end components. By the researches above, we concluded that (1) Hard-to-maintain phenomena such as lack of decomposition and over-encapsulation may be avoided by adopting a good design prior to CBD; (2) Components for front-end may have more capabilities of prefactoring, so components can be created purposefully according to the categories we proposed. In the future, we will continue our exploration on the *component timeline*, *orphan components*, and component code clones.

REFERENCES

- [1] Andreas Gizas, F., Sotiris P. Christodoulou, S., Theodore S. Papatheodorou, S.: Comparative evaluation of javascript frameworks. In: WWW '12 Companion Proceedings of the 21st International Conference on World Wide Web, 513-514(2012).
- [2] "React home," <https://facebook.github.io/react/>, accessed: Dec.15, 2017.
- [3] "React&Redux MVC," <https://hackernoon.com/thinking-in-redux-when-all-youve-known-is-mvc-c78a74d35133>, accessed: Feb.5, 2018.
- [4] Brown A W. Component-Based Development[J]. 2000.
- [5] Alan W. Brown, Large-Scale, Component Based Development, Prentice Hall PTR, Upper Saddle River, NJ, 2000
- [6] Vale T, Crnkovic I, Almeida E S D, et al. Twenty-eight years of component-based software engineering[J]. Journal of Systems & Software, 2016, 111:128-148.
- [7] Cappiello C, Daniel F, Matera M. A quality model for mashup components[C]//International Conference on Web Engineering. Springer, Berlin, Heidelberg, 2009: 236-250.
- [8] Magnusson E, Grenmyr D. An Investigation of Data Flow Patterns Impact on Maintainability When Implementing Additional Functionality[J].2016.

# A Lightweight Approach to Detect Memory Leaks in JavaScript

Ju Qian

College of Computer Science and Technology,  
Nanjing University of Aeronautics and Astronautics  
Nanjing 210016, China  
jqian@nuaa.edu.cn

Long Wang, Xiaoyu Zhou

School of Computer Science and Engineering,  
Southeast University  
Nanjing 211189, China  
zhouxy@seu.edu.cn

**Abstract**—Although with garbage collection support, many JavaScript programs still suffer from memory leaks. These leaks can affect application performance and even cause crashes, especially for single page websites. The existing work on JavaScript memory leak mainly focuses on the static detection of leaks toward certain leak patterns. The application scope of such approaches are limited. Previous techniques used for detecting memory leaks in Java-like languages might be extended to JavaScript. However, how to apply these techniques in JavaScript is still a problem. In this paper, we firstly present many common leak detection heuristics used in garbage-collected languages and investigate their effectiveness on JavaScript. According to the investigation results, we then propose a lightweight multi-snapshots based dynamic leak detection method for JavaScript. The initial experimental results show that the proposed approach is effective.

**Keywords**- *memory leak; JavaScript; dynamic analysis*

## I. INTRODUCTION

JavaScript is a popular language mainly used to develop dynamic web pages. Although with automatic memory management, JavaScript still suffers from the memory leak problem. If a useless object in JavaScript is unintentionally referenced by some long-term living references, the object cannot be reclaimed by a garbage collector. It will hence be leaked and occupy unnecessary memory.

For traditional web applications, the memory leaks in JavaScript may not cause serious problems, since the web pages of an application are usually frequently switched and the leaked memory can be reclaimed when a browser discards old pages. However, modern web applications are often single-paged. In those applications, the web pages of an application are no longer frequently switched. A web page may be alive for hours or even days using Ajax technologies to update contents without completely refreshing the page. Many rich client web applications, such as Google Gmail and Microsoft Office 365, follow this style. In such cases, the leaked memory can be largely accumulated, which can degrade the performance of web applications or even cause the applications to crash due to out of memory error.

To address the issue, early research detects circular references to catch memory leaks in the old browsers. In some old browsers like IE6, the DOM objects are garbage-collected with reference counting, while the other JavaScript objects are garbage-collected using some kinds of mark and sweep algorithms. Useless objects in DOM-related reference cycles may not be

effectively reclaimed. To help find the leaked objects, the SIEve/Drip tool [1] tracks all DOM nodes to find the ones involved in reference cycles. It then lets the users to determine which DOM nodes are leaked. Microsoft developed another similar tool named JavaScript Leaks Detector [2]. It reports circular reference caused leaks by simulating IE6 and IE7's garbage collection (GC) mechanisms. In recent browsers, the GC algorithms have already been improved. The unnecessary memory caused by DOM-related reference circles now can be automatically freed by a browser. Therefore, such memory leaks do not need additional efforts to detect and fix any more.

For the memory leaks caused by reachable useless objects, some static techniques have been proposed to detect certain leak patterns in web frameworks [3, 4]. A typical tool in this category is Leak Finder [3]. It detects memory leaks in Google's Closure library. The tool finds goog.Disposable objects in a heap snapshot and inspects these objects to check whether certain easily to leak objects are freed. Its memory detection capability is limited to the Closure library. Pienaar and Hundt proposed another more advanced tool JSWhiz [4]. The tool summarizes several leak patterns in the Closure library. It can statically detects many kinds of memory leaks based on the abstract syntax tree and type system of JavaScript source code. Even though, the application scope of such work is limited, since the used leak patterns are often bound to certain types of applications. Jensen et al. [5] and Rudafshani and Ward [6] also proposed approaches to detect memory leaks in JavaScript. Their approaches need to track the allocations and accesses of objects and hence can be very costly for large programs. More general and lightweight JavaScript memory detection tools are still in demand.

One insight for developing such general JavaScript memory leak detection techniques is to extend the existing techniques for Java-like languages [7] to JavaScript. Java suffers from memory leak problems similar to JavaScript's. To detect memory leaks in Java, one kind of approach locates leaks according to the growth trends of heap structures, e.g., [8-11]. Another kind of technique detects memory leaks according to the structural information within the heap, such as the ownership relation [12], the data structure similarity and reoccurrence [13], etc. Besides, some other approaches also find leaks using the object lifetime information, such as the age of objects [7] and the staleness of objects (how long the object have not be used) [14].

Although effective for Java, it is still unclear whether these techniques are still suitable for JavaScript, since JavaScript has

many individual characteristics that are different from Java, such as the dynamic type system and the prototype-based inheritance. To this end, this paper firstly studies the effectiveness of different memory leak detection heuristics which are borrowed from Java on JavaScript and then presents a dynamic approach to detect JavaScript memory leaks on the basis of JavaScript's own characteristics and the existing leak detection heuristics. The approach collects heap snapshots for web applications, and uses a lightweight statistical method combining many heuristics to recommend suspicious leaking objects. The initial experimental results show that the proposed approach can effectively detect memory leaks and hence can be helpful for the users.

## II. MEMORY LEAK DETECTION HEURISTICS

The general memory leaks caused by unbroken reference in garbage-collected languages are hard to be precisely detected by static analyses. For such leaks, dynamic analyses are often preferred. Even though there is a rich literature on the dynamic analysis techniques for memory leaks, most of the existing techniques are based on a few core detection heuristics.

### A. Leak detection heuristics

Table 1 shows many leak detection heuristics (or their core metrics) used by previous research. Before introducing their details, some basic concepts are firstly explained.

**GC Roots:** Root objects or references where a garbage collector starts its analysis. Typical GC roots include stack variables, static fields, class objects, etc.

**Ownership:** If in an object reference graph, every path from GC roots to a node  $n$  going through a node  $d$ , we say  $d$  owns  $n$ .

**Leak Root:** Root objects or references which directly or indirectly reference the whole leaked data structure. A leak root can represent a collection of leaked objects.

**Fringe:** Fringe [8] refers to the boundaries between the old objects and the newly created objects in the object reference graph of a heap snapshot.

In the introduction of leak detection heuristics, we suppose the heap change history forms a sequence of heap snapshots  $\mathcal{H} = (H_1, H_2, \dots, H_n)$ , where  $H_i$  is the  $i$ -th snapshot in the heap change history.

**DCR:** For an object type  $T$ , let  $D(T)$  and  $C(T)$  be the numbers of  $T$ 's instances destructed and constructed in a heap snapshot, respectively.  $DCR(T) = D(T) / C(T)$  is said to be the destruction/construction rate of  $T$ . If  $DCR(T)$  is continuously low in a heap snapshot sequence,  $T$  can be considered as a probably leaked object type [9].

**TIV:** Given two heap snapshots  $H_i$  and  $H_{i+1}$ , assume the numbers of objects of a type  $T$  in  $H_i$  and  $H_{i+1}$  is  $V_i$  and  $V_{i+1}$ , respectively. Then, from  $H_i$  to  $H_{i+1}$ , the increase volume of type  $T$  is  $TIV(T) = (V_{i+1} - V_i)$ . The types with high TIV values are more likely to be the ones with instances leaked.

**TPFI:** Assume the numbers of references between two object types  $T_1$  and  $T_2$  are  $R_i$  and  $R_{i+1}$  in two sequential heap snapshots  $H_i$  and  $H_{i+1}$ , respectively. Then, the type point-from relationship increment between  $T_1$  and  $T_2$  from snapshot  $H_i$  to snapshot  $H_{i+1}$  is  $TPFI = (R_{i+1} - R_i)$ . The larger TPFI, the more likely that the involved types are with objects leaked [10].

**LN:** Leaf nodes in an object reference graph are not likely to be the root causes of memory leaks. Therefore, it is better to not

TABLE 1. Leak detection heuristics (or their core metrics)

abbreviation	heuristics or their metric
DCR	Destruction/Construction Rate
TIV	Type Increase Volume
TPFI	Type Point-from Increase
LN	Leaf Nodes
IMN	Immutable Nodes
INN	Internal Nodes
NON	Non-owner Nodes
NAI	No Age Intersection
NF	No Fringe
OSR	On-stack Reachability
FOC	Fringe Ownership Count
NOC	New Ownership Count
SOC	Similar Object Count
LS	Life Span

directly report them as the leak diagnosis results.

**IMN:** Immutable objects with sizes not changed in different heap snapshots are unlikely to be leak roots.

**INN:** Internal objects maintained by the language runtime (e.g., JavaScript VM) are unlikely to be leaked.

**NON:** An object not owning any other objects is said to be a non-owner object. The non-owner objects are often close to GC roots, and their referenced objects are shared by other references. These non-owner objects are less likely to be leak roots.

**NAI:** The age of an object describes how long ago it has been created. In a heap snapshot, the objects created in the current heap snapshot and not holding references to the objects created in old snapshots, or the objects created in old snapshots and not holding references to the objects created in the current snapshot are said to be no age intersection objects. NAI objects are unlikely to be leak roots. If a new object does not reference old objects, it is likely to be a temporal object. If no new object is attached to an old data structure, then the old data structure is likely to be stable.

**NF:** No fringe objects refers to the objects owning no objects on the fringe. Memory leaked objects are often connected with fringe objects. No fringe objects are unlikely to be leaked, while the objects referencing to both fringe and no fringe objects have more possibility to be the leak causes [8].

**OSR:** The objects directly accessible from stack variables are more likely to be temporary objects instead of leak roots.

**FOC:** If an object owns more objects on the fringe, it is more likely to be a leak root object.

**NOC:** Objects owning a lot of newly created objects are more likely to be leak roots.

**SOC:** In a heap snapshot, if an object has more similar objects, then there will be high possibility that such type of objects are leaked.

**LS:** If an object firstly appears in a snapshot  $H_i$  and finally disappears since snapshot  $H_j$ , we say its life span is  $LS = (j - i)$ . The longer life span, the larger possibility that the object is leaked.

### B. Effectiveness of leak detection heuristics on JavaScript

We conducted experiments on some JavaScript programs to analyze the effectiveness of the above heuristics. The results are discussed as following.

#### (1) The type memory growth based heuristics (TG)

The DCR and TIV heuristics detect memory leaks according to the memory growth trends of each type. These heuristics are effective for JavaScript. However, the leak sources detected by them are mostly basic types like Object, Array, HTIMDivElement, etc.

This is because JavaScript uses a prototype-based inheritance mechanism, and the types dynamically extended from a root type like *Object* by attaching or removing properties at runtime are difficult to be distinguished from the root type. A basic type can have too many sub-types dynamically extending from it. Only knowing that objects of some basic types are leaked is not very helpful for leak diagnosis. Heuristics DCR and TIV must be combined with techniques that can classify objects with finer granularity to effectively help locating leak sources.

(2) The reference growth based heuristic (RG)

The TPFH heuristic detects memory leaks according to the growth of reference relationships between types. It can rank the leak causing reference relationships in high position. However, most of the reported results are relationships between basic object types. Such relationships are too rough for leak diagnosis. The reasons are similar to that of the DCR/TIV heuristics, which are also due to the very flexible nature of the JavaScript type system.

(3) The heap structure based heuristics (HS)

Heuristics LN, IMN, INN, NON, NAI, NF, OSR, FOC, and NOC mainly detect leaks by analyzing the structural attributes of objects on a single or two sequentially obtained heap snapshots. Our experiments show that applying heuristics LN, IMN, INN, NON, NAI, NF, and OSR can filter out a large number of objects that are unlikely to be leak roots and heuristics FOC and NOC are effective for suspicious leak root ranking. However, determining ownership relationships can be costly for large heap snapshots.

(4) The data structure similarity based heuristic (DSS)

Heuristic SOC can be used to partition objects into similarity groups and then analyze the properties of these groups to identify leak sources. In Java, we can at least use the type information to distinguish similar objects. However, in the prototype-based JavaScript language, many objects are created by dynamically extending the root *Object* type and the actual type information is hard to determine. Therefore, there need some other techniques to help determine the similarity between objects. Besides, we found the SOC heuristic should better be used together with TG heuristics to get more valuable results. The similarity groups can be viewed as finer-grained resolution of object types or data structures. Such grouping also can benefit many different methods which depend on type or data structure information.

(5) The object lifetime based heuristic (OL)

Heuristic LS detects memory leaks according to the object lifetime information. With this heuristic, we may detect a large number of individual leaked objects instead of a few object types or data structures. Because in JavaScript, objects are not with their types distinguished with fine granularity, such results do not provide clear clue for further leak diagnosis. Besides, when roughly tracking the lifetime of objects according to their occurrences in heap snapshots, without monitoring the uses (reads or writes) of objects, heuristic LS can easily lead to false alarms.

### III. A LIGHTWEIGHT LEAK DETECTION METHOD FOR JAVASCRIPT

According to the above findings, we believe an effective and lightweight way for JavaScript memory leak detection is to combine the TG, DSS, and some HS heuristics for leak object identification. We may follow the DSS heuristics to get a better resolution of types or data structures. The TG heuristics can be used to rank suspicious objects, and we can use the HS heuristics to filter out unlikely leaked objects. Under such idea, this section

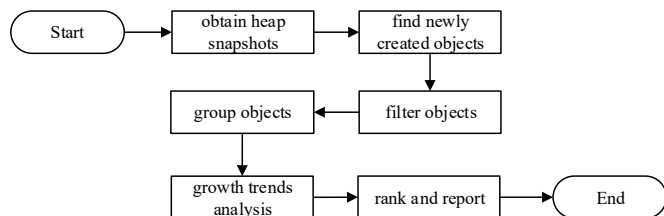


Figure 1. The workflow of the MS method

presents a lightweight multi-snapshots based leak detection method (the MS method). The method takes the characteristics of JavaScript language into account and can be more helpful for JavaScript memory leak diagnosis.

#### A. The detection method

The proposed method is a dynamic approach which detects memory leaks based on two or more heap snapshots obtained from a memory leaking program. The workflow of the method is shown in Figure 1. It takes 6 main steps.

In the first step, we obtain a sequence of heap snapshots from the execution of the target program. Then, the heap snapshots will be parsed and we traverse the object reference graphs embedded in the heap snapshots and compare every two adjacent snapshots to locate the newly created objects in a snapshot. The objects occurred in the current snapshot but do not occur in a previous one are considered as the newly created objects.

After that, the objects unlikely to cause memory leaks will be filtered out according to heuristics IMN, INN, NAI, and OSR.

Next, we classify the newly created objects into partitions according to the connections between objects. For two newly created objects, if they are connected, then the two objects will be put into the same partition. Each partition can be regarded as an individual data structure. We then follow the idea of heuristic SOC to categorize these data structures into similarity groups. Instead of doing object level similarity analysis, we find common parent objects of the objects in the above partitions on the object reference graph. A common parent object can be viewed as the representing node of one or more partitions. The data structures referenced by a common parent object are usually similar. Finding common parent objects works as a kind of data structure level similarity grouping. This can result in fewer groups compared to object level similarity grouping. Each common parent object can be regarded as a candidate leak root. By inspecting these leak roots, it will be easier to diagnose the root causes of memory leaks.

For the calculated candidate leak roots, we will further analyze their memory growth in the heap snapshots. Unlike doing memory growth analysis at type-level, the previous grouping can make the analysis results easier for further inspection. The total size of the objects in each object group represented by a common parent object is used as an approximation of the occupied memory of a leak root. If the occupied memory continues to grow, then the candidate is considered suspicious; otherwise, it will be excluded from the leak detection results.

After all the heap snapshots have been analyzed, we rank the candidate leak roots according to their totally occupied memory and the number of objects in the categorized object groups. The top ranked candidates will be reported as leak detection results.

To better show which objects are leaked, we use a chain of object property names that used to reach a leak root in the object reference graph as the identification of the leak root.

TABLE 2. The experimental subjects

Name	Library	Source
JQueryWeb	JQuery	<a href="http://javascript.info/tutorial/memory-leaks">http://javascript.info/tutorial/memory-leaks</a>
ExtWeb	ExtJS	<a href="http://www.sencha.com/forum/showthread.php?263439-ExtJS-Memory-Leak">http://www.sencha.com/forum/showthread.php?263439-ExtJS-Memory-Leak</a>
YuiWeb	YUI	<a href="http://yuilib.com/trac-archiver/tickets/2530415.html">http://yuilib.com/trac-archiver/tickets/2530415.html</a>
DojoWeb	Dojo	<a href="http://www.ibm.com/developerworks/cn/web/wa-sieve/">http://www.ibm.com/developerworks/cn/web/wa-sieve/</a>
MeteorWeb	Meteor	<a href="https://github.com/meteor/meteor/issues/1157">https://github.com/meteor/meteor/issues/1157</a>
BackWeb	Backbone	<a href="http://plnkr.co/edit/xfJWIF?p=info">http://plnkr.co/edit/xfJWIF?p=info</a>
AngularWeb	Angular	<a href="https://github.com/angular/angular.js/issues/4864">https://github.com/angular/angular.js/issues/4864</a>

### B. Experimental analysis

We conducted an initial experimental study on 7 JavaScript programs using popular libraries JQuery, ExtJS, etc. to validate the effectiveness of the proposed approach. The subjects are listed in TABLE 2. In the experiment, we use the Chrome browser to run the subject programs for a while and then use Chrome DevTools to obtain snapshots at different time points. The snapshots are exported to local files for further analysis. Each obtained heap snapshot can be viewed as an object reference graph. These snapshots are parsed and analyzed with Java language.

TABLE 3 shows the effects of different analysis steps in our lightweight leak detection method. The table only lists the experimental data when analyzing two adjacent snapshots. In the table, column *#new* shows the number of identified newly created objects. Column *#filter* shows the number of remaining objects after doing object filtering. Column *#partition* lists the number of categorized newly created object connection partitions, and column *#parent* shows the number of calculated common parent nodes for the object partitions. From the table, we can see that object filtering can greatly reduce the number of the objects to need be analyzed, and the object partitioning and common parent grouping can effectively categorize objects into suspicious object groups.

The final results of our lightweight leak detection method are shown in the rightmost two columns of TABLE 3. The results indicate that our proposed method can detect memory leaks in high precision. The reported numbers of suspicious leak roots are small, which can reduce the effort of further leak diagnosis and fixing.

TABLE 4 shows the analysis time consumed by different leak detection methods on the same snapshots. TG, RG, HS, and OL stand for the detection methods with different groups of heuristics applied, respectively. From these data, we can see that our lightweight multi-snapshots based method consumes very little time. This is because we only analyze the newly created objects on each snapshot, which greatly reduces the number of objects need to be processed. We use a lightweight method to calculate metrics for suspicious leak root ranking, which also reduces the analysis cost.

TABLE 3. Effects of different analysis steps

Subject	#new	#filter	#partition	#parent	#detected leak roots	#actual leak roots
JQueryWeb	95101	28431	14221	2	1	1
ExtWeb	19334	906	534	34	9	3
YuiWeb	12151	7050	937	14	2	1
DojoWeb	1805	290	84	2	1	1
MeteorWeb	165896	77609	4681	197	5	2
BackWeb	9030	2510	421	36	2	1
AngularWeb	3800	856	599	21	5	3

TABLE 4. Analysis time of different methods (ms)

Subject	TG	RG	HS	OL	MS
JQueryWeb	384	550	16325	465	198
ExtWeb	960	4466	5497	4168	179
YuiWeb	341	557	4436	560	547
DojoWeb	211	1175	1069	1113	135
MeteorWeb	638	1242	91332	1146	395
BackWeb	550	923	4540	827	553
AngularWeb	326	737	1116	691	190

### IV. CONCLUSION

In this paper, we firstly investigate the effectiveness of many common leak detection heuristics on JavaScript programs. Based on the investigation results, we propose a lightweight multi-snapshots based leak detection method for JavaScript. The method combines many effective heuristics and takes the characteristics of JavaScript language into consideration. Our experimental results show that it is both effective and efficient. In the future, we plan to further improve the method and conduct more experiments on more subjects to further validate its effectiveness.

### ACKNOWLEDGMENT

This work is supported by the China Defense Industrial Technology Development Program (Grant No. JCKY2016206B001 and JCKY2014206C002), the Science and Technology Planning Project of Jiangsu Province (BY2016003-02), and the National Natural Science Foundation of China (Grant No. 61472175).

### REFERENCES

- [1] IE/Sieve. <http://home.online.nl/jsrosman/>
- [2] JavaScript Memory Leak Detector. <http://blogs.msdn.com/b/gpde/archive/2009/08/03/javascript-memory-leak-detector-v2.aspx>
- [3] Leak Finder for Javascript. <http://code.google.com/p/leak-finder-for-javascript/>
- [4] J. A. Pienaar, R. Hundt. JSWhiz: Static analysis for JavaScript memory leaks. IEEE/ACM International Symposium on Code Generation and Optimization (CGO), 2013, pp. 1-11.
- [5] S. H. Jensen, M. Sridharan, K. Sen, S. Chandra. Meminsight: Platform-independent memory debugging for JavaScript. In Symposium on the Foundations of Software Engineering, 2015.
- [6] M. Rudafshani, P. AS Ward, LeakSpot: detection and diagnosis of memory leaks in JavaScript applications, Software: Practice and Experience, 47(1): 97-123, 2017.
- [7] V. Šor, S. N. Srirama. Memory leak detection in Java: Taxonomy and classification of approaches. Journal of Systems and Software, 2014.
- [8] N. Mitchell, G. Sevitsky. LeakBot: An automated and lightweight tool for diagnosing memory leaks in large Java applications. In the European Conference on Object-Oriented Programming, 2003.
- [9] K. Chen, J. B. Chen. Aspect-based instrumentation for locating memory leaks in Java programs. In Annual International Computer Software and Applications Conference (COMPSAC), 2007.
- [10] M. Jump, K. S. McKinley. Cork: dynamic memory leak detection for garbage-collected languages. ACM SIGPLAN Notices. 2007, 42(1): 31-38.
- [11] J. Qian, D. Zhou. Prioritizing test cases for memory leaks in Android applications, Journal of Computer Science and Technology, 31(5), 2016.
- [12] D. Rayside, L. Mendel. Object ownership profiling: a technique for finding and fixing memory leaks. In International Conference on Automated Software Engineering, 2007, pp. 194-203.
- [13] E. K. Maxwell, G. Back, N. Ramakrishnan. Diagnosing memory leaks using graph mining on heap dumps. In SIGKDD International Conference on Knowledge Discovery and Data Mining, 2010.
- [14] H. Yu, X. Shi, and W. Feng. LeakTracer: Tracing leaks along the way. In 15th International Working Conference on Source Code Analysis and Manipulation (SCAM), 2015 pp. 181-190.



# Pseudo-Exhaustive Verification of Rule Based Systems

D. Richard Kuhn<sup>1</sup>, Dylan Yaga<sup>1</sup>, Raghu N. Kacker<sup>1</sup>, Yu Lei<sup>2</sup>, Vincent Hu<sup>1</sup>

<sup>1</sup> National Institute of  
Standards and Technology  
Gaithersburg, MD 20899, USA  
{kuhn,dylan.yaga,raghu.kacker}@nist.gov

<sup>2</sup> Computer Science & Engineering  
University of Texas at Arlington  
Arlington, TX, USA  
ylei@uta.edu

**Abstract** — Rule-based systems are important in application domains such as artificial intelligence and business rule engines. When translated into an implementation, simple expressions in rules may map to a large body of code that requires testing. We show how rule-based systems may be tested efficiently, using combinatorial methods and a constraint solver in a test method that is *pseudo-exhaustive*, which we define as exhaustive testing of all combinations of variable values on which a decision is dependent. The method has been implemented in a tool that can be applied to testing and verification for a wide range of applications.

**Keywords** — *combinatorial testing; constraint solvers; formal methods; t-way testing; rule-based systems; test automation*

## I. INTRODUCTION

Rule-based systems have been important in a variety of application domains for many years. Some of the earliest artificial intelligence systems (AI) were designed to evaluate large rule sets, and this approach continues to be important for AI. In other domains, business rule engines automate complex enterprise resource planning (ERP) problems [1]. The terms used in rules may be expressed as Boolean (dichotomous) or relational conditions on inputs, values from databases, and environmental conditions such as time of day. Thus even a rule that contains only a few simple conditionals may invoke significant processing involved in computing the values used in the rule conditions. A rule-based system must work for any set of inputs, and can be implemented with a wide variety of rule engines. For example, JBoss, Oracle Policy Automation, OpenRules, Drools, IBM ODM, and many other tools exist to process rules supplied by users. But as with conventional software, exhaustive testing is nearly always intractable. This paper generalizes a practical method developed for testing access control systems [2], and introduces a tool that implements this method.

The approach to testing rule-based systems is *pseudo-exhaustive*, which we define as exhaustive testing of all combinations of variable values on which a decision is dependent. This approach is analogous to pseudo-exhaustive methods for testing combinational circuits [3], where the verification problem is reduced by exhaustively testing only the

subset of inputs on which an output is dependent, or by partitioning the circuit and exhaustively testing each segment. The general concept of exhaustively testing subsets of variable values on which a decision is dependent is applied here to rule-based systems by transforming rule conditions to disjunctive normal form, then considering each term separately [2].

Testing a rule-based system requires showing that the rules as specified,  $P$ , are correctly implemented. The implementation  $P'$  must be shown to produce the same response as  $P$  for any combination of input values used in rules. That is, for input values  $x_1, \dots, x_n$ ,  $P'(x_1, \dots, x_n) = P(x_1, \dots, x_n)$ . Positive testing to show that a rule produces a specified result is easy: instantiate conditions to true for each antecedent associated with the result and verify that the system returns the designated result. Negative testing, showing that no combination of input values will produce the same result when it should not, is much more difficult. With  $n$  Boolean variables there are  $2^n$  possible combinations of variables. For example, it would not be unusual to have 50 Boolean variables, resulting in  $2^{50} \approx 10^{15}$  combinations, which would appear to make full negative testing intractable. In this paper, we show how combinatorial methods can be used to make this testing problem practical, given assumptions that apply to many or most rule-based systems.

## II. TEST CONSTRUCTION

We describe the derivation of complete test cases from rules converted to  $k$ -DNF structure (disjunctive normal form where no term contains more than  $k$  literals, and a *term* is a conjunction of one or more literals within the disjunction), using a constraint solver and a covering array generator. Two arrays are constructed for each possible rule consequent, such that every test in each array should produce the same result, with variations indicating an error. The method may be applied to rule systems with multiple outputs, where outputs are either discrete values or are defined by a predicate or expression with a Boolean result.

Rules are assumed to be given as expressions made up of variables with logical connectives in an antecedent, with a consequent given as a discrete value or simple predicate, structured as shown below where  $R_i$  are predicates evaluating the values of one or more variables, and  $result_i$  is the result expected when conditions of  $R_i$  evaluate to true:

$$(R_1 \rightarrow result_1) (R_2 \rightarrow result_2) \dots (R_m \rightarrow result_m)$$

else → default

which is equivalent to:

$(R_1 \rightarrow result_1) (R_2 \rightarrow result_2) \dots (R_m \rightarrow result_m)$   
 $(\sim R_1) (\sim R_2) \dots (\sim R_m) \rightarrow default$

Each  $R_i$  may include multiple variables, conditions, and logical connectives. It is required that the rule antecedents  $R_i$  are mutually exclusive, i.e., for any set of input variable values, only one antecedent will be matched. We believe this requirement is not overly restrictive, as in most applications it would be an error for matches of more than one rule. (It would be possible to use the constraint solver to check that rule antecedents are mutually exclusive, but this feature has not been implemented.)

**Example 1:** Suppose we have a rule set as shown below:

```
if (a && (c && !d || e)) R1;
else if (!a && b && !c) R2;
else exit();
```

This code can be mapped to the following expression (note second line is "else", i.e., negation of predicates for R1 and R2):

$(a(c\bar{d}+e) \rightarrow R_1) (\bar{a} b \bar{c} \rightarrow R_2)$   
 $((\sim(a(c\bar{d}+e)))(\sim(\bar{a} b \bar{c}))) \rightarrow exit$

Literals can be conditions, such as age>18, or Boolean variables such as employee (yes, no), but the structure will be a series of expressions specifying subsets of conditions that produce each result, followed by a default rule when none of the attribute expressions have been instantiated to true.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0
2	0	0	1	0	1	1	1	1	1	0	0	0	0	0	1
3	0	1	0	0	0	1	0	1	1	1	1	1	1	0	0
4	0	1	1	1	1	0	0	1	0	0	0	1	0	0	1
5	1	0	0	0	1	0	0	1	0	0	1	1	1	1	0
6	1	0	1	1	0	1	1	1	0	1	0	1	1	1	0
7	1	1	0	1	1	1	0	0	0	1	0	0	1	0	1
8	1	1	1	0	0	0	1	1	1	1	1	0	1	1	1
9	1	0	0	0	0	1	1	0	1	0	0	1	1	1	1
10	0	1	1	1	0	1	1	0	1	0	1	1	0	1	0
11	0	1	0	0	1	0	1	0	0	0	1	0	1	1	1
12	1	0	1	1	1	0	0	0	1	0	1	0	1	0	0
13	1	0	0	1	1	0	1	1	1	1	1	1	0	1	1
14	1	0	1	0	0	1	0	0	0	0	1	1	0	0	1
15	0	1	1	0	0	1	1	0	0	1	1	0	0	0	0
16	0	1	1	0	1	0	0	0	1	1	0	1	1	1	1
17	1	1	0	1	0	0	1	1	1	0	0	0	0	0	0
18	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0
19	0	1	0	0	1	0	0	1	0	1	0	0	0	0	0
20	1	1	1	1	0	0	1	0	0	1	1	1	0	1	1
21	1	0	0	0	1	1	0	0	1	0	0	0	0	1	0
22	0	1	1	1	0	1	1	1	0	1	0	0	1	1	1

Figure 1. 3-way covering array of 15 boolean parameters

To make testing tractable, we use combinatorial methods [2][4]. To see the advantages of a combinatorial approach, refer to Fig. 1, which shows a covering array of 15 boolean variables. A covering array is an  $N \times k$  array of  $N$  rows and  $k$  variables. In every  $N \times t$  subarray, each  $t$ -tuple occurs at least once. In software testing, each row of the covering array represents a test, with one column for each parameter that is varied in testing. For example, Fig. 1 shows a complete 3-way covering array that includes all 3-way combinations of binary values for 15

parameters in only 22 tests. The size of a  $t$ -way covering array of  $n$  variables with  $v$  values each is proportional to  $v^t \log n$  [6][7]. For Example 1, with five attributes and two possible decisions for each attribute, there are  $2^5 = 32$  possible rule instantiations. However, a covering array of all 3-way combinations contains only 12 rows. The number of variables for which all settings are guaranteed to be covered in a covering array is referred to as the *strength*; a 3-way array is of strength 3. We use covering arrays of variables from rules that have been converted to  $k$ -DNF form. For example,  $abc + de$  contains two terms, one with three literals and one with two, so the expression is in 3-DNF form. The covering array does not contain all possible input configurations, but it will contain all  $k$ -way combinations of variable values. Where an expression is in  $k$ -DNF, any term containing  $k$  literals that is resolved to true will clearly result in the full expression being evaluated to true. For example, an access control rule in 2-DNF form could be: "if employee && US\_citizen || auditor then grant". This rule contains one term of two attributes and one term of one attribute, so it is 2-DNF. Because a covering array of strength  $k$  contains every possible setting of all  $k$ -tuples and  $i$ -tuples for  $i < k$ , it contains every combination of values of any  $k$  literals.

As noted in the Introduction, we exhaustively test all combinations of values on which a decision is dependent. For the example above, the decision grant depends on either of two terms being true: employee && US\_citizen or auditor. Any other setting of these three variables should result in deny. A truth table of all eight possible settings of these three variables would allow exhaustive testing of this set of rules. In general, exhaustive testing is intractable for nearly all applications, but note that at most two variables are required to produce a grant result. So if we test all 2-way combinations of settings of the input variables, we have achieved exhaustive testing of all combinations of variable values on which a decision is dependent, since no decision depends on more than two variables. (Later in the paper we show how this approach scales up to larger problems, and address the effectiveness for detecting errors when implemented rules contain more variables than are included in specified rules.)

Covering array generation tools, such as ACTS [4][6], make it possible to include constraints that prevent inclusion of variable combinations that meet criteria specified in a first order logic style syntax. For example, if we are testing applications that run on various combinations of operating systems and browsers, we may include a constraint such as 'OS = "Linux" => browser != "IE"'. Constraints are typically used in situations such as this, where certain combinations do not occur in practice or are physically impossible, and therefore should not be included in tests. Modern constraint solvers such as Choco [8] and Z3 [9] make it possible to process very complex constraint sets, converting logic expressions into combinations that are invalid and can be avoided in the final array.

*Method:* Let  $R =$  rule antecedents (left side of an implication rule such as  $p$  in  $p \rightarrow q$ ) of one or more rules being tested in  $k$ -DNF, and  $T_i$  are terms (conjuncts of one or more variables or

terms) in  $R$ . We designate the result/consequent of the rule being tested as (+), and any other possible result as (-). For the example included in Example 1, terms  $T_i$  of  $R_1$  would be  $acd$ , and  $ae$ , and  $R_1$  would be designated as (+) and  $R_2$  or exit() designated as (-), for this test.

*Positive testing:* Generate a test set PTEST for which every test should produce a particular response. It must be shown that for all possible inputs, where some combination of  $k$  input values matches a (+) condition, a (+) result is returned. Construct test set PTEST = {PTEST<sub>i</sub>} with one test for each term  $T_i$  of  $R$  as follows: PTEST<sub>i</sub> =  $T_i(\bigwedge_{j \neq i} \sim T_j)$

The construction ensures that each term in P is verified to independently produce the expected response for that rule. Negating each term  $T_j, i \neq j$ , prevents masking of a fault in the presence of other combinations that would return the same result. For example, if a rule condition is  $ab + cd \rightarrow R_1$ , inputs of 1100, 1101, 1110 could be used for testing  $ab \rightarrow R_1$ . However, input 1111 would not detect the fault if the system ignores variable  $a$  or  $b$ , because the condition  $cd$  would cause a result of  $R_1$ , and no other predicates in the rule would be evaluated. One such test is required for each term in a rule, so for  $m$  rules with an average of  $p$  terms each, the number of tests required is proportional to  $mp$ .

*Negative testing:* Generate a test set NTEST for which every test should produce a response other than the result designated by the rule being tested. It must be shown that for all possible inputs, where no combination of  $k$  input values matches a rule, an alternative result is returned.

NTEST = covering array of strength  $k$ , for the set of variables in all rules, with constraints specified by  $\sim R_i$ .

Note that the structure of the rule evaluation makes it possible to use a covering array for NTEST, compressing a large number of test conditions into a few tests. Converted to  $k$ -DNF, each rule antecedent includes a sequence of conditions that are each sufficient to trigger the specified result. Because rule antecedents are mutually exclusive, masking of one combination by another can only occur for NTEST when a test produces a negative response, i.e., a response that is not a consequent of the rule instantiated in PTEST. In such a case, an error has been discovered, which can be repaired before running the test set again. Since NTEST is a covering array, the number of tests will be proportional to  $v_k \log n$ , for  $v$  values per variable (normally  $v=2$  since most will be Boolean conditions), and  $n$  variables.

Rule antecedents are assumed to be mutually exclusive (to prevent masking as discussed above), but we allow for cases where multiple rules may have the same consequent (result). In such cases, rule antecedents are combined to produce the set of conjuncts used in generating PTEST and NTEST arrays. For example, if two rules are  $R_1 \rightarrow Q_1$  and  $R_2 \rightarrow Q_1$ , then  $k$ -DNF terms for PTEST are produced from  $(R_1 + R_2)$  and constraints for NTEST are given by  $\sim(R_1 + R_2)$ . For  $m$  rules with the same

consequent (result), the number of tests is multiplied by the constant  $m$ .

**Example 2:** Table I gives a set of Boolean variables  $a$  through  $e$ , where each row defines values for the variables that determine an access control decision, either *grant* (+) or *deny* (-). Thus a covering array for the antecedent  $R$  of a rule in 3-DNF such as  $(acd + \bar{a}b\bar{c} \rightarrow grant)$  is given in Table 1. The total number of 3-way combinations covered is the number of settings of three binary variables multiplied by the number of ways of choosing three variables from five, i.e.,  $2^3 \binom{5}{3} = 80$ .

TABLE I. 3-WAY COVERING ARRAY

	a	b	c	d	e
1	0	0	0	0	0
2	0	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	0	0	1	1
6	1	0	1	0	0
7	1	1	0	0	1
8	1	1	1	1	0
9	1	1	0	0	0
10	0	0	1	1	0
11	0	0	0	0	1
12	1	1	1	1	1

TABLE II. 3-WAY COVERING ARRAY WITH CONSTRAINT  $\sim R$

	a	b	c	d	e
1	0	0	0	0	0
2	0	0	1	1	1
3	0	1	1	0	0
4	1	0	0	1	0
5	1	0	1	1	0
6	1	1	0	0	1
7	1	1	1	1	1
8	0	0	1	0	1
9	1	1	0	1	0
10	0	0	0	1	1
11	1	0	0	0	0
12	0	1	1	1	0
13	1	0	0	0	1
14	0	1	1	0	1

Table II shows a covering array for this set of variables generated using  $\sim R$  as a constraint. That is, the two terms of the rule,  $acd$  and  $\bar{a}b\bar{c}$ , have been excluded from the array, but all other 1-, 2-, and 3-way combinations can be found in the array. Because  $acd$  and  $\bar{a}b\bar{c}$  are the only conditions under which access should be granted, the array in Table II should result in a *deny* response from the system for every test. Collectively, tests include all 78 3-way settings of variables that will not instantiate the access control rule to *true*.

### III. FAULT DETECTION PROPERTIES

Now consider the faults that this method can detect. Suppose that some combination of variables exists that produces a different response than required by the rule set  $P$ , for example because of errors in code that instantiates variable values. Tests contained in PTEST and NTEST will detect a large class of

missing terms, added terms, or altered terms containing  $k$  or fewer variables. In this section we analyze faults that will be detected, and the underlying conditions in these faults. Table III illustrates the fault types and detection conditions for each.

TABLE III. EXAMPLE FAULTS AND DETECTION CONDITIONS.

	Term	C=correct term	F=faulty term	PTEST detect condition	NTEST detect condition	notes
1	missing	$abc$	--	$abc$	$none$	
2	added	--	$ab$	$none$	$ab$	
3		$abc$	$\bar{a}b$	$none$	$\bar{a}bc, \bar{a}b\bar{c}$	
4		$abc$	$ab$	$none$	$ab\bar{c}$	
5		$ab$	$abc$	--	--	<i>no fault</i>
6	altered	$abc$	$ab\bar{c}$	$abc$	$ab\bar{c}$	
7		$abc$	$ab$	$none$	$ab\bar{c}$	
8		$abc$	$\bar{a}b$	$abc$	$\bar{a}bc, \bar{a}b\bar{c}$	

*k-DNF detection property:* It is shown in [2] that collectively, tests from PTEST and NTEST will detect faults introduced by added, deleted, or altered terms with up to  $k$  variables. We can also show [2] that if more than  $k$  attributes are included in the altered term, some faults are still detected. Specifically, where a correct term has more than  $k$  variables and is not a subset of a faulty term, the fault will be detected. If a correct term is a subset of a faulty term in this case, some faults will be detected.

#### IV. SOFTWARE TOOL

The prototype research tool, Pseudo-Exhaustive Verifier (PEV), was developed in Java, and utilizes several open source external Java libraries. The software is packaged as a Java Archive (.jar) file which is directly executable as a Graphical User Interface (GUI), or can be run as a Command Line Interface (CLI) from a terminal.

The PEV software has been designed to accept rule sets comprised of Boolean variables, Boolean operators and relational expressions, implementing the algorithm described in Sect. II. The software parses the rule set, converts to Disjunctive Normal Form (DNF), inverts the DNF rule set, solves for positive conditions, and uses NIST's Automated Combinatorial Testing for Software (ACTS) tool [6] to compute a covering array for negative conditions.

*Algorithm implementation:* PEV utilizes several publicly available Java Archive libraries to generate test arrays. Transforming the input Boolean rule set to DNF is done using *jbool expressions* [10], and the Choco constraint solver [11] is used for resolving relational statements.

*Parsing:* Parsing is a critical step of the PEV software, which occurs before any testing is performed. Since the software needs to accept input from the user, any input must be modified and sanitized prior to use, to ensure compatibility with the various APIs used, as well as to catch any syntactical problems prior to testing... The parser strips extraneous whitespace, and then normalizes Boolean operators (&&, &, ||, |, !, ~), and attempts to match open and closing parenthesis. This sanitization ensures compatibility with the various APIs used

throughout the software, and catches any syntactical problems prior to testing.

The software is not restricted to Boolean expressions, and has initial support for relational expressions (e.g.,  $b < 3$ ;) . Note that a semicolon is used to identify a relational expression. During parsing, PEV will locate numeric relational expressions and replace them with temporary Boolean variables. After the replacement, the rule set is processed as normal. The relational values are solved at a later step and the results are recorded.

Once the initial input rule set is parsed, the software will convert it to Disjunctive Normal Form (DNF) to be tested. The user will be presented with a breakdown of the DNF rule set (split on the OR statements), each part of which is a positive condition that needs to be solved. Additionally, the user can set minimum and maximum values for any relational variable found in the rule set.

*Solve for positive conditions:* Each individual expression between OR operators is an expression that, once solved, will produce one positive condition. These expressions represent the only possible positive conditions for the original rule set – so it is possible to produce exhaustive positive conditions.

Consider Fig. 1, with the original input rule set:

```
emp & age>18; & (fa | emt | med) | b<3;
```

Converted to DNF, this is:

```
((age > 18; & emp & emt) | (age > 18; & emp & fa) | (age > 18; & emp & med) | b < 3;)
```

Splitting on the OR operators, there are four individual expressions for the positive conditions (replacing relational expressions with temporary Boolean variables  $tmp0 = age > 18$ ; and  $tmp1 = b < 3$ ;) :

- $tmp0 \& emp \& emt$
- $tmp0 \& emp \& fa$
- $tmp0 \& emp \& med$
- $tmp1$

To solve these expressions, any variable present is evaluated with the following rules, as shown in Table V:

- Non-negated variables evaluate to true
- Negated variables evaluate to false
- Variables not present evaluate to false

*Solve for negative conditions:* Depending on the complexity of the input rule set, it may not be feasible to produce exhaustive negative condition output combinations. By utilizing combinatorial test methods, it is possible to generate covering arrays of sufficient strength to have good test coverage. The method for producing negative conditions can be found by generating the full covering array for all the unique Boolean variables within the rule set, and using the DNF rule set as a constraint – which will remove the positive conditions from the resulting output.

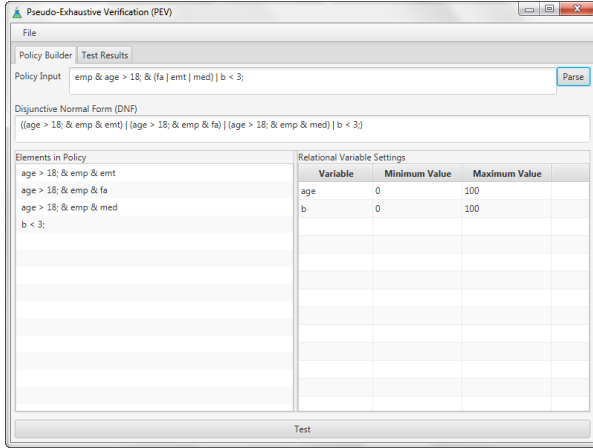


Figure 1. PEV software, after initial rule set parsed

TABLE V - SOLVED POSITIVE CONDITIONS

Expression	tmp0	tmp1	emp	emt	fa	med
tmp0 & emp & emt	1	0	1	1	0	0
tmp0 & emp & fa	1	0	1	0	1	0
tmp0 & emp & med	1	0	1	0	0	1
tmp1	0	1	0	0	0	0

A covering array for all negative conditions is computed as described in Sect. II. To perform this task, PEV creates an internal instance of the ACTS software, and passes a list of the unique Boolean variables from the rule set (including temporary Boolean replacements for relational expressions). The next step is to add the DNF rule set as a constraint to the system – so that the positive conditions are not included as negative results. Finally, the  $k$ -way combination is dynamically set after the  $k$ -DNF transform, which finds the conjunction with the largest combination of Boolean variables. In this example, the value of 3 is set (Table VI). PEV currently supports  $k = 2..6$ , because ACTS is used as the covering array generator, but there is no inherent limit to 6-way combinations and the method could support  $k > 6$ .

*Solve for relational expressions* using the Choco Expression Parser and the Choco Constraint Solver.

Relational Expression Formatting: The general format is:

```
Variable OPERATOR Integer_Value; Or
Integer_Value OPERATOR Variable;
```

Every relational expression must end with a semicolon (;), and two or more relational expressions in a row (without Boolean operators between them) will be replaced with one temporary Boolean variable during parsing. An example is shown in Table VII.

After being extracted and replaced by temporary Boolean variables, and the Positive/Negative conditions are found, an instance of Choco Expression Parser is created, and the relational expressions are passed as parameters. The minimum and maximum range for the expression to test against must be set – the PEV GUI includes a section which will allow the adjustment of every relational variable min and max values (default set to 0 to 100). These values can be adjusted prior to

testing the rule set so that a customized range can be found. The solutions to the solved expressions are then placed into the results where appropriate.

TABLE IV. SOLVED NEGATIVE CONDITIONS

tmp0	tmp1	emp	emt	fa	med
1	0	1	0	0	0
1	0	0	1	1	1
0	0	1	1	1	0
0	0	0	0	0	1
0	0	0	1	0	0
0	0	1	0	1	1
1	0	0	0	1	0
0	0	1	1	0	1
1	0	0	1	0	1
0	0	0	0	1	0
1	0	0	0	0	1
1	0	0	1	0	0

TABLE V. INPUT POLICIES AND RESULTING PARSED RULE SET

Input Rule set	Parsed Rule set
a > 10; 20 < b;    n	tmp0    n
a > 10;    20 < b;    n	tmp0    tmp1    n

*Results:* Once testing completes, PEV displays usage metrics and parameters which will result in positive conditions, and the covering array for negative conditions. At this point, the results can be saved as a comma separated value (.csv) file.

## V. TEST SET SIZE AND PRACTICAL IMPLICATIONS

The process scales easily to systems with a large number of variables and rules. Because the number of rows in a covering array grows only with  $\log n$  for  $n$  variables at a given number of values, a large increase in the number of variables requires only a few additional tests.

The most significant limitation for this approach occurs where terms in rules contain a large number of values per variable. Because the number of rows of a covering array increases with  $v^k$ , for  $v$  variable values, if terms in the rules have more than 10 to 12 values, it may not be practical to generate covering arrays. However, a large number of tests is not a barrier, because the structure of the solution resolves the oracle problem by ensuring that every test in PTEST should produce a response of (+) and every test in NTEST should produce a response of (-). Consequently, tests can be fully automated, making it possible to execute a large test set.

## VI. RELATED WORK

This paper generalizes a method developed originally for testing attribute-based access control systems [2], which had been incorporated into the Access Control Policy Testing tool ACPT [12]. The generalized method and new tool, PEV, were developed to make the method useful in development and testing for a wider range of applications. Pseudo-exhaustive test methods for circuit testing have an extensive history of application [1]. While our method is not derived from these earlier approaches, it shares the basic notion of determining dependencies, partitioning according to these dependencies, and

testing exhaustively the inputs on which an output is dependent. We have previously applied this notion to software testing in a more general form, using the observation that faults depend on a small number of inputs, by covering all 2-way to 6-way combinations of inputs [13]. This earlier work generated a test oracle using a model checker with a formal specification of a system, instantiated with inputs from a covering array.

Relatively little work has been published on testing specifically for rule-based systems. Dalal et al. [15] describe a case study of a rule-based system in an evaluation of model based testing, including the use of the combinatorial testing tool AETG. However, their testing considered only high level properties, such as whether updates correlated with the assignment of jobs during a working day. That is, no tests were generated from the rules. Rule based systems have also been used in a number of studies of test data generation [16][17], but used rules in generating tests for other software, rather than testing the rule-based systems themselves.

Among automated test generation systems, PEV falls into the class of tools with a specified test oracle, using the taxonomy of Barr et al. [14], because system rules serve as a specification of system behavior. Many such systems have been developed. The test oracles used in those systems were designed to answer the question "For a given set of inputs and initial state, what is the system output?", using a formal spec of some kind. Given such an oracle, test inputs must also be provided. Our method differs from these in that we address a narrower class of systems, but trade this limitation for complete coverage of inputs up to  $k$ -way combinations, providing testing that is pseudo-exhaustive, i.e., exhaustive for all subsets of inputs on which a rule result is dependent.

## VII. CONCLUSIONS

Rule-based systems are used extensively in applications such as enterprise resource planning and machine learning [20]. If rules contain at most  $k$  Boolean variables per conjunction, for an expression in  $k$ -DNF, then a  $k$ -way covering array can test all possible settings of such terms. Thus for any possible combination of  $n$  inputs, only  $k$  ( $k < n$ ) matter in determining the truth of the expression. In most applications, the number of conditions in conjunction will be small, even though the number of rules may be very high, possibly several hundred or even into thousands. The number of rows in a  $k$ -way covering array of Boolean variables is proportional to  $2^k \log n$ , and the ACTS covering array generator used in PEV produces arrays up to 6-way. Therefore PEV can efficiently process thousands of conditions or rules with up to six conditions per conjunction, sufficient for practical use.

The method described here was initially used in access control policy testing [2], and PEV has extended its applicability to a broader range of potential use. We are also considering methods to improve the efficiency of the PEV tool, including use of SAT solvers for generating covering arrays [18][19]. It may be possible to integrate the methods described in this paper with SAT-solver based covering array generation, to produce more compact arrays.

To make the tool more useful for practical application, features to allow import and export from common rule system formats, or decision table structures, may be helpful. We plan to investigate the possibilities depending on interest from users. We have received inquiries regarding compatibility with commercial tools, which could be considered for further development. Thus far, the major interest for this test method is for business rule systems, but it could be applied to traditional expert system applications as well.

Note: *Identification of products does not imply endorsement by NIST, nor that products identified are necessarily the best available for the purpose.*

## REFERENCES

- [1] Lu, R., & Sadiq, S. A survey of comparative business process modeling approaches. In *Intl Conf on Business Information Systems* (pp. 82-94). Springer, 2007.
- [2] Kuhn, D. R., Hu, V., Ferraiolo, D. F., Kacker, R. N., & Lei, Y. (2016, April). Pseudo-exhaustive testing of attribute based access control rules. In *Software Testing, Verification and Validation Workshops (ICSTW), 2016 IEEE Ninth International Conference on* (pp. 51-58).
- [3] McCluskey, E. J. (1984). Verification Testing: A Pseudoexhaustive Test Technique. *Computers, IEEE Transactions on*, 100(6), 541-546.
- [4] Kuhn, D. R., Kacker, R. N., & Lei, Y. (2010). SP 800-142. Practical Combinatorial Testing, NIST, Gaithersburg, MD 20899
- [5] ACTS Home Page, <http://csrc.nist.gov/acts/>
- [6] Y. Lei, R. Kacker, D.R. Kuhn, V. Okun, J. Lawrence, IPOG: A general strategy for t-way software testing. *14th intl conference on the engineering of computer-based systems*, 2007, pp 549-556
- [7] D. M. Cohen, S. R. Dalal, M. L. Fredman, and G. C. Patton, "The AETG System: An Approach to Testing Based on Combinatorial Design," *IEEE Trans. Software Eng.*, 23(7):437-444, 1997.
- [8] Jussien, N., Rochart, G., & Lorca, X. (2008). Choco: an open source java constraint programming library. *Open-Source Software for Integer and Constraint Programming (OSSICP'08)* (pp. 1-10).
- [9] De Moura, L., & Bjørner, N. (2008). Z3: An efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems* (pp. 337-340). Springer Berlin Heidelberg.
- [10] [https://github.com/bpodgursky/jbool\\_expressions](https://github.com/bpodgursky/jbool_expressions)
- [11] <https://github.com/kaktus40/choco-exppar>  
<http://www.choco-solver.org/>
- [12] ACPT Home Page, [http://csrc.nist.gov/groups/SNS/acpt/access\\_control\\_policy\\_testing.html](http://csrc.nist.gov/groups/SNS/acpt/access_control_policy_testing.html)
- [13] D. R. Kuhn, V. Okun, *Pseudo-exhaustive Testing For Software*, 30th NASA/IEEE Software Engineering Workshop, April 25-27, 2006
- [14] Barr, E. T., Harman, M., McMin, P., Shahbaz, M., & Yoo, S. (2015). The oracle problem in software testing: A survey. *IEEE transactions on software engineering*, 41(5), 507-525.
- [15] Dalal, S. R., Jain, A., Karunanithi, N., Leaton, J. M., Lott, C. M., Patton, G. C., & Horowitz, B. M. (1999, May). Model-based testing in practice. *21st Intl Conf on Software Eng.* (pp. 285-294). ACM.
- [16] Deason, W. H., Brown, D. B., Chang, K. H., & Cross, J. H. (1991). A rule-based software test data generator. *IEEE transactions on Knowledge and Data Engineering*, 3(1), 108-117.
- [17] Edvardsson, J. A survey on automatic test data generation. *2nd Conference on Computer Science and Engineering* (pp. 21-28) 1999.
- [18] Lopez-Escogido D, Torres-Jimenez J, Rodriguez-Tello E, Rangel-Valdez N. Strength two covering arrays construction using a sat representation. In *MICAI 2008: Advances in Artificial Intelligence 2008 Oct 27* (pp. 44-53). Springer Berlin Heidelberg.
- [19] Banbara M, Matsunaka H, Tamura N, Inoue K. Generating combinatorial test cases by efficient SAT encodings suitable for CDCL SAT solvers. In *Logic for Programming, Artificial Intelligence, and Reasoning 2010 Oct 10* (pp. 112-126). Springer.
- [20] Lee, C. C. (1991). A self-learning rule-based controller employing approximate reasoning and neural net concepts. *International Journal of Intelligent Systems*, 6(1), 71-93.

# Metrics for Data Uniformity of User Scenarios through User Interaction Diagrams

Douglas Hiura Longo and Patrícia Vilain  
Informatics and Statistics Department,  
Federal University of Santa Catarina,  
Florianopolis, Brazil  
douglasshiura@inf.ufsc.br, patricia.vilain@ufsc.br

**Abstract**— In the software development process, the acceptance testing may be used by non-technician users to define software requirements. In this article, we use the US-UIDs (User Scenarios through User Interaction Diagrams) as automated acceptance tests in order to provide communications and collaboration between programmers and users. We propose three metrics for measuring the data uniformity in the US-UIDs. These metrics are investigated in four projects. The resulting measures from the investigations of the four projects are used to build a scale with classes to classify the uniformity of the US-UIDs. The classes (duplication, uniformity, irregularity) were created from empiric evaluation and compared to the measures from the offered metrics. The classification purpose is to identify the US-UIDs as uniform or irregular.

**Keywords:** *US-UID; User Scenarios Through User Interactions Diagrams; ATDD; Uniformity; Acceptance Test; Automated Test; Executable Test; Executable Requirement; Quality Factor; Requirements; Requirements Specification; Metrics.*

## I. INTRODUCTION

Analogous to the Test-Driven Development (TDD) [1], the Acceptance Test-Driven Development (ATDD) includes team members with different perspectives (client, developer, tester) collaborating to write acceptance testing before deploying the functionality [2]. Teams that try ATDD generally find that only by defining acceptance tests when discussing requirements outcomes there will be a better understanding. However, the acceptance tests force us to reach a solid agreement about the exact behavior that the software should expose [3].

The User Scenarios through User Interaction Diagrams (US-UIDs) are suggested to allow that non-technician users define software functional requirements in the ATDD approach [4, 5]. The US-UIDs are used to specify primarily the values of the information exchanged between the user and the computer in tasks that represent functional requirements, mainly in information systems and can be used as automated acceptance tests [6].

To use a US-UID as an automated test, the following steps are performed: specification of the US-UID, nomination of the fixtures that represents the US-UID

elements, and creation of the glue code that will link these fixtures to the SUT (System Under Testing) code [6]. Although data uniformity problems are generated in the specification step, the identification of these problems occurs usually in all three steps of test automation.

Figure 1 shows a pair of US-UIDs to exemplify the uniformity problem in the data. The example considers only a part of the original US-UIDs to show data with uniformity problem. Both US-UIDs show the same functionality, the specification of the authentication system, but show different values to the information. The example highlights two uniformity problems, where the first is related to the values of the user inputs “Mary” and “John”. Both values are different from one another but have the same sense. The sense is clearly a user name. This sense can be extracted through the experience with people names and with data from close elements in the US-UID. In this authentication system, it is hard to deconstruct the sense of user names for these values, however, for specific systems with little known specialties between the stakeholders, the non-uniform data causes loss of sense. The second uniformity problem is related to the system outputs “Enter” (Figure 1, US-UID A) and “Log In” (Figure 1, US-UID B). Both systems outputs represent in SUT implementation the button text for the action of entering in the system. For communication purposes, the stakeholder can understand the text of these two system outputs that, although different, have the same sense. However, as an automated test, there will be a problem with the implementation. As the system outputs of a US-UID are also assertions and the assertions rely and capture the SUT values, it is inviable that an implementation, in an automated manner, could answer to two distinct values (“Enter” and “Log in”) in the same way, unless the same action is duplicated in the SUT.

The irregular data generation occurs mainly in the shared specification, i.e., when more than one stakeholder specifies US-UIDs to a same system. Thus, when there is more than one person specifying different US-UIDs, usually there is no care to keep the data uniform, as each person uses the data domain that he/she knows for the test. Other way to generate non-uniform data is by the partial repetition of a US-UID path. The partial repetition is necessary because there cannot be deviations and branches in the US-

UIDs [4, 5]. Therefore, when you repeat manually parts of the US-UIDs, it is usual to change small details and not apply the changes in all the US-UIDs, that way, there will be a uniformity problem that will be spread in all test automation steps. Furthermore, as the US-UIDs of a project increases, more difficult becomes the uniformity problems identification.

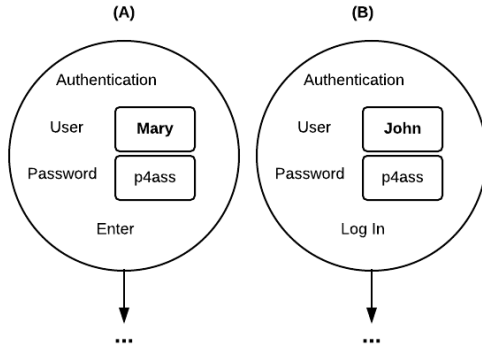


Figure 1. Two fragments of US-UIDs with data uniformity problems

The general purpose of this work is to suggest metrics for measuring data uniformity from US-UIDs. The specific purpose is the empiric evaluation of the metrics to create a classification that allows the identification of US-UIDs with irregular data.

This paper is organized as follows. The second section shows the theory basics about the US-UIDs. The proposal is detailed in the third section. The fourth section shows the projects of the evaluation. The fifth section shows the results of the evaluation. Finally, the sixth section shows the conclusions.

## II. BACKGROUND

The US-UIDs are used for specifying software requirements. The US-UIDs have been suggested as a specialization of the UID technique [7], where the abstract information is replaced with concrete values from the user scenarios. The applicability of the US-UIDs is usually made by non-technician users to create acceptance testing before the development. In agile development teams, the US-UID can be used for communication and collaboration between the stakeholders in software development.

Figure 2 shows an example of US-UID with the interactions of the sum operation using of a calculator. According to Longo and Vilain [4], this example was adopted to explain to non-technician users how to specify the US-UIDs. With the knowledge acquired from the example, non-technician users have participated in experiments to evaluate the correctness and the completeness of the US-UIDs [4, 5].

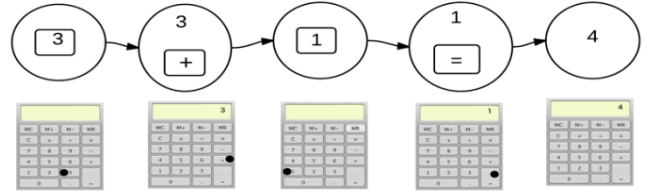


Figure 2. Example of a US-UID of a calculator sum operation, as suggested by Longo and Vilain [4, 5]

In this example, the user enters the values of the sum operation ( $3 + 1 =$ ) and the system shows the result (4). In the example, five states of interaction are shown, meaning each state of interaction (ellipse) contains the user input and the system output values. The state of interaction flow is represented by the arrow direction through the states of interaction. The initial state is the first state of interaction that follows the arrows and the end state is the last state in the flow. Table I shows the language symbols from the US-UIDs.

TABLE I. SYMBOLS FOR THE LANGUAGE OF US-UIDS [1, 2]

Symbol	Use
	<b>Ellipse</b> – represents a state of interaction.
	<b>Arrowed line</b> – represents the direction flow, i.e., the transition between interactions states.
	<b>Rectangle</b> – represents the user input, its value is represented by a set of characters placed within the rectangle.
Characters sequence	<b>Value</b> – represents the system output, where a set of characters is placed within the ellipse.

### A. Mathematical Model of the US-UIDs

The US-UIDs can be represented in a mathematical model. Thus, as suggested by Longo et al. [6], the structure of a US-UID is formed by a set of states of interaction and each state of interaction is, in turn, formed by a set of user inputs and a set of system outputs. A state of interaction is represented by:

$$\delta_i = \{\varepsilon_{i1}, \varepsilon_{i2}, \varepsilon_{ij}, \dots, \varepsilon_{in}, o_{i1}, o_{i2}, o_{il}, \dots, o_{im}\}, \quad (1)$$

$$(\forall i (i = 1; k)), (\forall j (j = 1; n)),$$

$$(\forall l (l = 1; m))$$

Given a state of interaction,  $n$  is the amount of system outputs and  $m$  is the amount of user inputs,  $o_{il}$  is the  $l$ -th system output from  $i$ -th state of interaction,  $\varepsilon_{ij}$  is the  $j$ -th user input of the  $i$ -th state of interaction. A US-UID is represented as follows:

$$\tau_t = \{\delta_{t1}, \delta_{t2}, \dots, \delta_{ti}, \dots, \delta_{tk}\}, \quad (2)$$

$$(\forall i (i = 1; k)), (\forall t (t = 1; d))$$

The  $k$  is the amount of states of interaction of the US-UID. The  $\delta_{ti}$  is the  $i$ -th state of interaction of the  $t$ -th US-



UID. As a restriction, the set must have at least a state of interaction.

### III. PROPOSAL

The lack of uniformity in data may cause problems in the communication and collaboration between the stakeholders, when defining the fixture names and the glue code. Therefore, it is important to evaluate the data uniformity in the US-UID specification step. For evaluation, easy-to-apply uniformity metrics are useful, especially, computational metrics, with measures of easy availability for the stakeholders.

This paper proposes three metrics to measure the uniformity of US-UIDs data. The proposed metrics are of absolute uniformity, absolute irregularity and relative uniformity. The metrics for absolute uniformity and absolute irregularity are created by comparing pairs of US-UIDs. A set of pairs of US-UIDs is generated from a set of US-UIDs. The set of pairs of US-UIDs is defined by:

$$\psi = \{(\tau_1, \tau_2), (\tau_1, \tau_3), \dots, (\tau_t, \tau_q), \dots, (\tau_{(d-1)}, \tau_d), \dots, (\tau_d, \tau_{(d-1)})\},$$

$$(\forall t (t = 1; d)), (\forall q (q = 1; d)), t \neq q, d > 1$$
(3)

where,  $d$  is the amount of US-UIDs from generated set. The set generated with the pairs of US-UIDs should have at least two US-UIDs. Both  $\tau_t$  and  $\tau_q$  are two any US-UIDs from a US-UIDs set.  $(\tau_t, \tau_q)$  is a pair formed by distinct US-UIDs. For the pairs formation, the restriction is that the formed pairs cannot exist with the same US-UIDs, such as  $t \neq q$ .

#### A. Metrics for the absolute uniformity

The absolute uniformity is calculated for each pair  $(\tau_t, \tau_q)$  where each pair of US-UID is split by user inputs and system outputs. The user inputs and system outputs are also compared in pairs.

A pair of uniform system outputs is formed by a system output of  $\tau_t$  and by another system output of  $\tau_q$ . The criterion to form the pairs of system outputs is that the data must be identical. The measure of the absolute uniformity of system outputs from a pair  $(\tau_t, \tau_q)$  is calculated by the following formula:

$$UniformOutput_{(\tau_t, \tau_q)} = \sum_{i=1}^k \sum_{l=1}^m \begin{cases} 1 & \text{if } o_{il} \in \tau_q \\ 0 & \text{otherwise} \end{cases}, \quad (4)$$

$$(\forall \delta_{ti} | \delta_{ti} \in \tau_t), (\forall o_{il} | o_{il} \in \delta_{ti})$$

The expression  $o_{il} \in \tau_q$  means that the system output  $o_{il}$  belongs to one of the system outputs among all the states of interaction  $\tau_q$ . The absolute uniformity of the system outputs is equal to the count of all system outputs from all states of interaction of a US-UID that have a pair compared to another US-UID. The absolute uniformity of the user inputs ( $UniformInput_{(\tau_t, \tau_q)}$ ) is built in a similar way to the metrics for system outputs.

#### B. Metrics for the absolute irregularity

In this study, absolute irregularity is the complement of absolute uniformity. The metrics for the absolute irregularity is built for a pair  $(\tau_t, \tau_q)$  of US-UIDs. The construction of this metric is similar to the metrics for absolute uniformity, in that the metrics for absolute irregularity is sectioned by user inputs and system outputs. The absolute irregularity of the system outputs from a pair  $(\tau_t, \tau_q)$  of US-UIDs is calculated in relation only to system outputs belonging to  $\tau_t$ . So, the system outputs belonging to  $\tau_t$  that do not have a pair are counted as being irregular. The criteria for not forming a pair of system outputs is that a system output belongs to  $\tau_t$  and that no other belongs to  $\tau_q$  with identical data. The absolute irregularity of the system outputs of a pair  $(\tau_t, \tau_q)$  is calculated by the following formula:

$$NonUniformOutput_{(\tau_t, \tau_q)} = \sum_{i=1}^k \sum_{l=1}^m \begin{cases} 1 & \text{if } o_{il} \notin \tau_q \\ 0 & \text{otherwise} \end{cases}, \quad (5)$$

$$(\forall \delta_{ti} | \delta_{ti} \in \tau_t), (\forall o_{il} | o_{il} \in \delta_{ti})$$

The expression  $o_{il} \notin \tau_q$  means that the system output  $o_{il}$  does not belong to the system outputs among all states of interaction  $\tau_q$ . The irregularity metrics for user inputs ( $NonUniformInput_{(\tau_t, \tau_q)}$ ) is built in a similar way to the metrics for system outputs.

#### C. Metrics for the relative uniformity

The relative uniformity metric is computed from the absolute uniformity and absolute irregularity metrics. So, for the measure of the relative uniformity we have the following equation:

$$RelativeUniformity_{(\tau_t, \tau_q)} = \frac{UniformInput_{(\tau_t, \tau_q)} + UniformOutput_{(\tau_t, \tau_q)}}{UniformInput_{(\tau_t, \tau_q)} + UniformOutput_{(\tau_t, \tau_q)} + NonUniformOutput_{(\tau_t, \tau_q)} + NonUniformInput_{(\tau_t, \tau_q)}} * 100$$
(6)

The outcome measured from the relative uniformity takes the values in the range [0%, 100%]. For each comparison pair  $(\tau_t, \tau_q)$  the relative uniformity is calculated. This way, for a set of pairs of US-UIDS, the average of the relative uniformity can be used as a quantitative value that represents the general uniformity.

#### D. Computational implementation of the metrics

The uniformity metrics have been implemented in a computational version<sup>1</sup>. The computational version allows applying the metric in a large set of US-UIDs. The version has been implemented with the paradigm of object-oriented

<sup>1</sup> <https://github.com/douglashiura/us-uid>

programming and Java programming language. The implementation has been developed to measure the US-UIDs specified in the framework Sc3n4r10 [6]. It handles the US-UIDs in files in JSON format<sup>2</sup>. This way, the files in JSON formats are converted in Java objects and then applied to the metrics.

#### IV. EVALUATION OF METRICS

To evaluate the metrics, the US-UIDs of four projects are taken into account. In each project, the metrics are applied, and the outcomes compared. The projects are specified in different ways and have a uniformity gap between them. In this section, research issues are also shown and discussed for applicability of the metrics on the projects.

##### A. Project P1: 8-Puzzle

The 8-puzzle is a game that consists in a grade with three lines and three columns. The grade has a sequence of numbers from 1 to 8 and a blank. The purpose of the game is to start in a random state and put in order the sequence of numbers. This project is comprised by the US-UIDs from Longo and Vilain [4, 5] study that measured the completeness and the correctness of the requirements specified by non-technician users. In the original experiment, fourteen participants specified the winning state of the game. During the specification, each participant should consider at least one move in the grade. Essentially, the US-UIDs specified in this project are duplicated, showing only small differences, as all participants had to specify the same requirement. This way, presumably, the data from the US-UIDs should be 100% uniform.

##### B. Project P2: Web Application

The web application is a system that has been developed for evaluation and monitoring of the courses chain from e-Tec Brazil<sup>3</sup>. The system contains a database with surveys and evaluation outcomes about Brazilian Federal Education Institutes. A student evasion module has been developed with the specifications of the US-UIDs a priori. For the evasion module, four US-UIDs have been specified. The specification work was developed by two users and two experts in US-UIDs. After being specified, the US-UIDs were automated with tests and reviewed for quality improving, where the best data uniformity was considered. In this project, the US-UIDs were developed with the best correctness and completeness as possible.

##### C. Projects P3 and P4: Messaging System

The message system is an experiment where the requirements were chosen by the participants themselves. The participants were requested to think about the requirements for an application similar to WhatsApp, Telegram, Hangout or Messenger and, then, specify the requirements as US-UIDs. Two projects were performed in the same experiment. In the project P3, a participant, along

with an expert, specified the US-UIDs. The expert reviewed the US-UIDs produced in order to minimize uniformity problems. The project P4 was performed with four participants and without the expert's help. In the project P4, we tried to simulate the situation of shared specification where the irregularity of the data of the US-UIDs occurs. For both of these projects, the controlled factor is the aid from expert. By the practical knowledge, we know that the aid from expert is significantly important for a better quality of the uniformity.

##### D. Uniformity a priori of the projects

The four projects were selected with characteristics that contribute to a gap in the uniformity of US-UID data among the projects. The unevenness on the uniformity of each project is considered for subsequent evaluation of the sensitivity of the proposed metric. Table II shows the classification a priori of the uniformity of the projects. The classification was made by an expert on US-UIDs. For the classification, the characteristics of each project and the manual review from an expert were considered. The manual identification of the uniformity problems is complex when the amount of US-UIDs increases. Other factor that complicates the manual evaluation is the number of elements belonging to the US-UIDs evaluated. For example, when there are lots of elements as states of interaction in the US-UIDs, the evaluation becomes complex as well. For the manual evaluation of the US-UIDs from an expert, the number of elements should be small, around four US-UIDs, because, above this amount, lots of doubts may be raised for the manual evaluation. The projects were evaluated in a general way by the specialist, i.e., an evaluation was performed to each project.

TABLE II. CLASSIFICATION A PRIORI OF THE PROJECT UNIFORMITY

Project	Classification of the uniformity
P1	Duplication
P2 and P3	Uniformity
P4	Irregularity

The characteristics of the three classes are:

- Duplication: occurs when the same requirement is specified in lots of US-UIDs, although small peculiarities occur in each specification.
- Uniformity: occurs when the diagrams are being specified carefully and taking into account the domain data. In projects P2 and P3, uniformity was maintained with the help of experts.
- Irregularity: occurs when the diagrams are specified by different people and each person considers only the data of their own knowledge and not of the general domain. In project P4, it was observed that uniformity was not maintained since each participant individually specified the US-UIDs and without the expert's help.

<sup>2</sup> <https://www.json.org/>

<sup>3</sup> <http://saas.etec.ufsc.br/>

## V. RESULTS OF THE METRICS EVALUATION

This section shows the results and the analysis of applicability of the measures of uniformity from projects.

### A. Relative Uniformity

The relative uniformity was calculated to all pairs of US-UID for the four projects. Figure 3 shows a graphic with the box plot of the relative uniformity of each project.

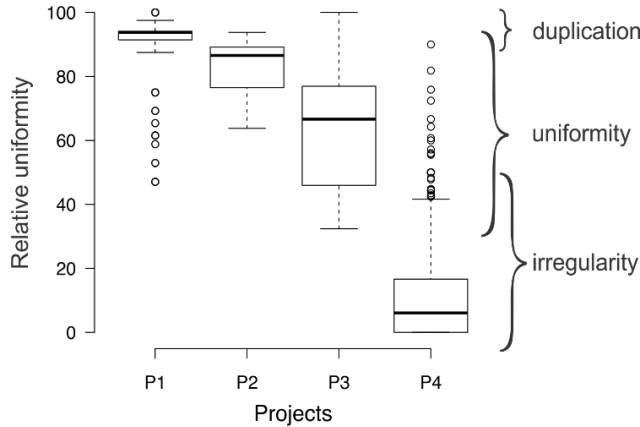


Figure 3. Visual comparison of the distribution of relative uniformity of each pair of US-UIDs for each project

On the right side of the chart there are three intervals for the three classifications of uniformity defined by the expert to each project. The relative uniformity interval for the duplication class is [47.1% to 100%]. The relative uniformity interval for the uniformity class is [32.4% to 100%]. The relative uniformity interval for the irregular class is [0% to 90%]. Ideally, the intervals should be continuous with no overlapping, however, the projects keep the real characteristics, i.e., they are not projects with all controlled data, removing discrepancies. This way, to classify the measures of relative uniformity, it is more suitable to consider a scale of classes of continuous intervals and with no overlapping. The suitable scale can be built using intervals closer to medians, but after analyzing the sensitivity of the metric.

#### 1) Sensitivity of the metrics

The result of the Kruskal-Wallis ANOVA statistical analysis for uniformity measure of the projects shows the statistical measure ( $H=422.3$ ), with degrees of freedom equal to (3) and probability of significance ( $P_{\text{value}} = 0.0000$ ). This way, for a level of significance of 5% ( $\alpha=0.05$ ), the test suggests that there is a difference in measures of uniformity between the projects, therefore, the alternative hypothesis ( $H_1$ ) is asserted. Thus, a *post hoc* test is needed to identify among which projects there is a difference in the measure of relative uniformity. Table III shows a comparison of the measures of relative uniformity among the projects.

TABLE III. COMPARISON POST HOC AMONG THE AVERAGES OF RELATIVE UNIFORMITY AS PER THE PROJECTS

Projects	$P_{\text{value}}$	Statistical Decision ( $\alpha = 0.05$ )
P1 x P2	1.0000	H0: Insensitive
P1 x P3	0.1265	H0: Insensitive
P1 x P4	0.0000	H1: Sensitive
P2 x P3	1.0000	H0: Insensitive
P2 x P4	0.0000	H1: Sensitive
P3 x P4	0.0000	H1: Sensitive

The statistical analysis suggests that there is no significant difference among the measures of uniformity in the projects P1, P2 and P3, but there is a significant difference for the measures of relative uniformity between the projects P1 and P4, P2 and P4 and P3 and P4. This way, we can notice that the classes of duplication and uniformity that are classified manually by the expert are not sensitive to the metric, i.e., the metric is not able to classify between duplication and uniformity. However, the metric is sensitive and can classify among the classes of uniformity and irregularity as per the evaluation from expert. For the projects P2 and P3 that are classified as uniform, the metric is insensitive, i.e., the metric does not measure differences, because, in fact, there are no significant differences in the measure of relative uniformity between the projects P2 and P3.

### B. Absolut Uniformity and Irregularity

The absolute uniformity is the count of the pairs of user inputs and system outputs sectioned in uniform and irregulars. Figure 4 shows the box plots for visual comparison of the absolute uniformity and irregularity of each project. The first characteristic that we can notice is that the amount of pairs of system outputs is greater than the amount of pairs of user inputs in all projects. This first characteristic is compatible with the fact that in the US-UIDs there are more system outputs than user inputs. The second characteristic is that there are more uniform system outputs in the projects that were classified as duplicate and uniform (P1, P2 and P3). Adversely, for the project classified as irregular (P4), there are more irregular system outputs. The third characteristic that we can notice is that there are less uniform user inputs than irregular ones in all projects. However, this third characteristic is less accentuated in project P3 if compared with the project P4, where we can conclude that the system outputs also influence the uniformity, but it is more difficult to analyze and keep the uniformity during the specification of the US-UIDs. The fourth characteristic is about the influence on the automation process of the tests in the uniformity. During the test automation process, US-UIDs are reviewed to improve quality. Project P2 considers this process. Also, project P2 has more uniform system outputs than others projects. In practice, the testing automation process is done by the programmers and, in this process, with the help from the stakeholders, the US-UIDs are corrected and implemented. In general, programmers are more rigorous with the review process and tend to review US-UIDs to improve uniformity.

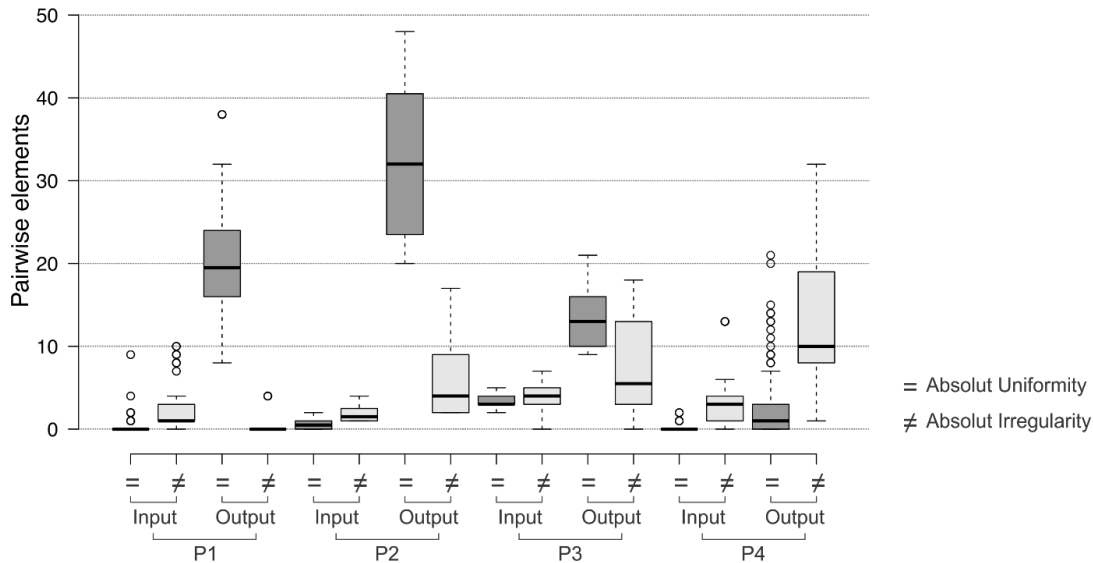


Figure 4. Visual comparison of the absolute uniformity of the user inputs and system output.

## VI. CONCLUSIONS

This paper shows three metrics for measuring data uniformity of the US-UIDs. The metric of relative uniformity, based on metrics of uniformity and absolute irregularity, is important for measuring and classifying the US-UIDs. The metric takes values from measure of uniformity in the interval [0% to 100%], where 0% is irregular and 100% is uniform.

In order to evaluate the metric, four empirically pre-evaluated projects were used by an expert. The evaluation from expert considered three classes for classifying the uniformity: duplication, uniformity, irregularity. With the results of the applicability of the metrics, it was concluded that, through the measure of relative uniformity, only two classes are suitable. The metric of relative uniformity is not sensitive for three classes of the projects. Based on the results, the suitable classes are uniformity and irregularity. The class of irregularity takes the interval [0% to 45%] from the measure of relative uniformity. The class of uniformity takes the interval [45% to 100%] from the measure of relative uniformity. The value 45% of relative uniformity is the boundary between both classes, but in both projects classified as uniform and irregular, a measure overlapping has occurred, where this point was arbitrary defined as the most suitable for boundary between both classes. So, this classification is suitable to evaluate the US-UIDs during the process of specification and reviewing them, if necessary, before starting the testing automation process.

The measures of absolute uniformity and irregularity are complementary and can be used to evaluate and compare the types of elements in the US-UIDs. The elements of type of system outputs are more present in the US-UIDs and are also more uniform, however, it was unable to create a classification like the relative uniformity one. However, through the projects, the elements of system outputs have weights of uniformity different than the user inputs.

With the specifications of uniform data of the US-UIDs, it's expected avoid reworking and improve the communication between users and developers. However, the evaluation of quality criteria not always has significant results in practice [8], so, it also should be investigated how to apply the metrics during the specification of the US-UIDs and how to guide the users in uniformity troubleshooting.

The main contributions of this work are the proposed metrics, the computational implementation of the metrics and the evaluation of four projects. Moreover, the US-UIDs used in the evaluation are available (<https://github.com/douglashiura/us-uid-uniformity>) for future investigations of this acceptance testing format for the software development.

## REFERENCES

- [1] BECK, Kent, "Test-driven development: by example," Addison-Wesley Professional, 2003.
- [2] Gärtner, Markus, "ATDD by example: a practical guide to acceptance test-driven development," Addison-Wesley, 2012.
- [3] Hendrickson, Elisabeth, "Driving development with tests: ATDD and TDD," STARWest 2008, 2008.
- [4] Longo, Douglas Hiura, and Patricia Vilain, "Creating User Scenarios through User Interaction Diagrams by Non-Technical Customers," SEKE. 2015, pp. 330-335.
- [5] Longo, Douglas Hiura, and Patricia Vilain, "User scenarios through user interaction diagrams," International Journal of Software Engineering and Knowledge Engineering 25.09n10, 2015, pp.1771-1775.
- [6] Longo, D. H., Vilain, P., da Silva, L. P., & Mello, R. D. S, "A web framework for test automation: user scenarios through user interaction diagrams," In Proceedings of the 18th International Conference on Information Integration and Web-based Applications and Services. ACM, 2016 pp. 458-467.
- [7] Vilain, P., Schwabe, D., de Souza, C., "A diagrammatic tool for representing user interaction in UML," <<UML>> 2000- The Unified Modeling Language. Springer, 2000, pp.133-147.
- [8] Lucassen, G., Dalpiaz, F., van der Werf, J. M. E., & Brinkkemper, S., "Improving user story practice with the Grimm Method: A multiple case study in the software industry," In International Working Conference on Requirements Engineering: Foundation for Software Quality. Springer, Cham, 2017, pp. 235-252.

# Feedback Topics in Modern Code Review: Automatic Identification and Impact on Changes

Janani Raghunathan, Lifei Liu, Huzefa Kagdi  
Department of Electrical Engineering and Computer Science  
Wichita State University  
Wichita, Kansas 6760, USA  
Email: {jxraghunathan, lxliu2, huzefa.kagdi}@wichita.edu

**Abstract**— Recent empirical studies show that the practice of peer-code-review improves software quality. Therein, the quality is examined from the external perspective of reducing the defects/failures, i.e., bugs, in reviewed software. There is a very little to no investigation on the impact of peer-code-review on improving the internal quality of software, i.e., what exactly is affected in code, due to this process. To this end, we conducted an empirical study on the human-to-human discourse about the code changes, which are recorded in modern code review tools in the form of review comments. Our objective of this study was to investigate the topics which are typically addressed via the textual comments. Although, there is an existing taxonomy of topics, there is no automatic approach to categorize code reviews. We present a machine-learning-based approach to automatically classify reviewer-to-reviewer and reviewer-to-developer comments on proposed code changes. We applied this approach on 468 code review comments of four open source systems, namely *eclipse*, *mylyn*, *android* and *openstack*. The results show that *Evolvability* categories are dominating topics. In an attempt to verify these observations, we analyzed the code changes that developers performed on receiving these comments. We identified several refactorings that are congruent with the topics of review comments. Refactorings are mechanisms to improve the internal structure of software. Therefore, our work provides initial empirical evidence on the effectiveness of peer-code-review on improving internal software quality.

## I. INTRODUCTION

Different types of maintenance help improve sustainability and quality of large-scale software systems. *Corrective* maintenance helps eliminating or reducing defects in the software; thereby, improving the external software quality. *Preventive* or *Perfective* types of maintenance may not necessarily address the defects or features at hand directly; however, they improve the design or code structure; thereby, improving the internal software quality. Software testing is primarily used to identify the defects and is often considered as an ubiquitous mechanism to improve the external quality. Code review is a rejuvenated phenomena with benefits in improving the software quality.

Extensive research shows the usefulness of code review in improving the external quality [1], [2], [3], [4], which is an important result. Unfortunately, there is little to no work on investigating how code review improves the internal software quality. Previous studies [5], [6], [7], [8], [9], [10], [11], [12] show that the internal quality is also critical and equally (or more) important. For example, evolvability or internal design issues could lead to code decay and/or premature degradation.

DOI reference number: 10.18293/SEKE2018-097

Peer-code review is the process of reviewers critiquing the code changes that developers submit to decide if those changes are of acceptable quality and can be integrated to the main code base of a software system. Nowadays, it is often lightweight, informal and tool-based, which is termed as Modern Code Review (MCR) [13]. MCR is popularly used in industrial and open source software-development paradigms [14], [2], [15]. The primary discourse between reviewers and developers is in the form of textual feedback about the code changes at the line level or collectively within an enabling tool, e.g., *Gerrit*.

The primary objective of this paper is to investigate how code review is effective in improving the internal software quality, which we substantiated with three research questions:

*RQ1*: What are the common topics, related to both internal and external qualities, discussed during code review?

*RQ2*: To what extent these common topics can be automatically classified?

*RQ3*: How do the developers address the review comments that reviewers provide to revise their code changes?

With respect to RQ1, we analyzed the code review repositories of four open source projects, namely *eclipse*, *mylyn*, *android* and *openstack*, which are archived in *Gerrit*. We manually investigated the three most relevant attributes of information from *Gerrit*: the patch description as it contains the reason the patch is submitted, the changed lines of code by the developers and the review comments from the reviewers for the proposed code changes. We classified each of the review comments into the most appropriate topic based on the taxonomy of Mantyla et al. [16]. Our results indicate that topics related to both external and internal qualities are found during the code review; however, those related to the internal quality are dominating. As a prime example, 75% of the defects identified during code review are evolvability type of defects and hence the majority of the comments raised by the reviewers are focussed on improving the internal software quality.

With respect to RQ2, we present an automatic approach to identify the topics found in code reviews. Our approach builds a machine-learning-based classifier to automatically categorize the code change into its appropriate topic. The results of our automatic classifier suggest that evolvability type of issues in a code can be well predicted with an average precision and recall of 0.45 and 0.41 respectively. Although, Mantyla et al. [16] proposed the taxonomy, they did not offer any automatic

solutions to classify topics or issues typically identified during the code review. We also discuss application scenarios of this automatic classification (see Section IV).

With respect to RQ3, we investigated the review comments from *mylyn*, *eclipse* and *android*. We manually studied the changes in the code for the review comments raised by the reviewers. As every review comment corresponded to a specific topic, we were able to correlate the action of a developer to a particular topic. We observed that developers adopted different refactorings to address different defect topics in code review. We investigated the structural changes that took place in different revisions of code review as a result of review comments and observed that developers did perform refactoring to address the review comments. By comparing those refactoring techniques with the respective topics, we observed a substantial congruence between the topics of feedback from reviewers and the specific (categories) of refactorings developers performed to address the review feedback in revising their code changes. Refactoring is a mechanism to improve the internal design of the software. Therefore, it is evident from our results that code review is effective in improving the internal software quality.

## II. BACKGROUND ON MCR AND TAXONOMY OF TOPICS

We define the key concepts involved in the modern code review process, which is driven by supporting infrastructure and tools, e.g., *Gerrit*.

*Code Change*: A code change is a set of modified source code files submitted in order to fix a bug or add a feature.

*Patch Description*: A brief information about the patch and the reason it is submitted. For instance, it contains information about the bug id from *Bugzilla*, if the patch is submitted to address a particular bug.

*Review*: A code review is a record of the interactions between the owner of a change and reviewers of the change including comments on the code and signoffs from reviewers.

*Owner*: An owner is the developer who makes the change in the source code and submits it for review.

*Reviewer*: A reviewer on a particular review is a developer who is assigned to and/or contributes to that review.

*Review Comment*: A review comment is textual feedback written by a reviewer about the code change during the review process. A review comment may be about the change in general or may be explicitly tied to a particular part of the change called the in-line comment.

The life-cycle of a review is as follows: Initially a developer (owner) makes changes to the source code in response to a bug report or feature request. Once complete, they submit the code change for review. The owner may indicate the intended reviewers, who are subsequently notified about the review invitation. It should be noted that the invited reviewers do not necessarily accept the invitation and contribute to the review. Reviewers then inspect the change through the code review tool (a web page in the case of *Gerrit*) and provide feedback in the form of review comments to the owner. The owner may update the change and submit the update to the review as a

result of such feedback. The code change is typically depicted by showing the difference of the code before and after the change. Eventually, a reviewer signs-off on the review, once they believe the code change is of sufficient quality to be checked into the code repository. If a change never received sign-off, it is abandoned. It is critical that code review is both effective (actually improves code changes and blocks poor code from being checked into the repository) and timely (does not act as a bottle-neck too by slowing down changes). Therefore, automation and tool support are key to its success.

Mantyla et al. [16] divided defects detected from code review into the following parent groups: evolvability, functional, and false positives. Evolvability defects are sub-divided into Documentation, Visual Representation, and Structure. The Functional category has seven groups: Resource, Check, Interface, Logic, Timing, Support and Larger defects. Each of these defect categories have a definition as defined by Mantyla et al. and C. Bird et al. in their works [16] and [17] respectively. We have used the taxonomy proposed by them to classify the reviews in our dataset. Mantyla et al. [16] have broadly referred to all types of issues found in code during code review as defects, irrespective of whether it is an external defect affecting the functionality of the software or an internal design flaw affecting the internal quality of the software. For consistency and simplicity, we adopted the same terminology. We have referred to all types of issues found by the reviewers as different types of defects or topics found in the code.

We define an evolvability defect as a defect in the code that makes the code less compliant with standards, more error-prone, or more difficult to modify, extend, or understand. The functional defects are those that cause a system failure or fail in their business logic. False positives are those class of defects which were initially suspected to be defects but later on were discovered to be as no defects during team meetings. Each category is further divided into sub-categories.

## III. EMPIRICAL STUDY: FORMULATION OF BENCHMARK

The purpose of this study was twofold: 1) to determine which specific categories were prevalent in the open-source systems under study, and 2) to curate a benchmark to assess our automatic approach for classification (see Section IV). There is no established dataset nor benchmark for our context in the literature. Additionally, our effort can be considered as an independent empirical verification or replication of the categories of Mantyla et al. [16]. That is, we address:

*RQ 1*: What are the common topics, related to both internal and external qualities, discussed during code review?

*Dataset and Methodology*: For our study, we collected the patches from four open source systems namely *eclipse*, *mylyn*, *android* and *openstack* between the periods Jan 2014 and Feb 2016 and classified the review comments into either of the categories proposed by Mantyla et al. [16] and Bosu et al. [17]. Each patch has a brief textual description called the patch description, the owner who submitted it, the files that are modified as part of the patch, the list of reviewers selected to review the patch, the line of code that changed and the

review comment that the reviewers made on the changed line of code. In order to create the benchmark, we considered the following patch selection criteria: 1) patches that were merged or abandoned, 2) that had either in-line or general review comments from the reviewers. We also considered those patches that had a relevant bug id in bug tracking system like *Bugzilla* to better understand the patch and thereby classify the review more accurately. Review #22722<sup>1</sup> from *Mylyn*, is an example of how we classified an individual review into its appropriate defect category. The reviewer *Sam Davis* commented on the changes in the file *BugzillaRestPostNewTask.java*: *This is creative but I'd rather use ImmutableList from Guava*. The owner *Frank Becker* had used an *ArrayList* and the feedback suggested the use of *ImmutableList* instead. It is evident from the changed line of code and the keywords in the review comment, namely, "use *ImmutableList*" that the reviewer is proposing an alternate approach to the owner's solution. As *Mantyla et al.* mentioned in their work [16] that the comments that suggested function call changes or a complete rethinking of the current implementation, belonged to *Solution Approach* defect category. We classified this review into *Solution Approach* category. In another example, review #52373, the reviewer commented *When you copy a big chunk of code like this, it would help to add a comment saying where it's copied from because it's a sign that we might want to create a common implementation in the future*. Similarly, in this case, after inspecting the changed line of code and the keywords from the review comment, namely, "add comment" suggest that reviewers are concerned about the documentation. Hence, we categorized this review into *Documentation* category.

*Results*: While consolidating the results of our manual analysis, we observed a total of 17 defect categories which covered the majority of the defects identified during code review. They are: Check Function, Check User Input, Check Variable, Compare, Compute, Data and Resource Manipulation, Wrong Location, Algorithm/Performance, Organization, Parameter, Solution Approach, Support, Supported by Language, Textual, Variable Initialization, Visual Representation and Compiler Error. Table I shows the distribution of defect categories within and across projects. They show that topics across evolvability and functional categories are found in code review. They cover both external and internal aspects of the reviewed code. It is also evident from these results that 75% (352 out of 468 review comments) of the defects identified during code review are evolvability type defects. Overall, we see that the internal quality aspects are dominant.

#### IV. APPROACH: AUTOMATIC CLASSIFIER

As discussed in the previous section, the review topics pervade both external and internal software qualities; however, identifying them is non-trivial, tedious, non-scalable, among other things. Therefore, we need to consider their automatic identification. That is, we address:

*RQ2*: To what extent can we automatically classify the common topics?

TABLE I  
DEFECT CATEGORY/TOPICS COMMONLY FOUND IN EACH PROJECT

Defect categories	Eclipse Platform(86)		Mylyn(108)		Android Platform(98)		OpenStack(177)	
	# of reviews	%	# of reviews	%	Total No. of reviews	%	# of reviews	%
<i>Evolvability Defects</i>								
Textual	39	45.35	25	23.36	38	38.8	51	28.8
Supported by language	8	9.3	7	6.54	10	10.2	8	4.54
Organization	6	7	14	13.08	8	8.16	36	20.45
Solution Approach	9	10.50	27	25.23	9	9.18	18	10.22
Visual Representation	8	9.3	3	2.8	18	18.37	9	5.11
<i>Functional Defects</i>								
Compare	3	3.48	4	3.73	1	1.02	3	1.7
Compute	6	6.97	2	1.86	5	5.1	6	3.4
Check Function			9	8.41	3	3.06	33	18.8
Check Variable	4	4.65	11	10.28	6	6.12	8	4.54
Check User Input								
Algorithm/ Performance							1	0.56
Wrong Location	1	1.16	3	2.8				
Data and Resource Manipulation	1	1.16						
Variable Initialization			1	0.93			2	1.13
Parameter			1	0.93				
Support			1	0.93				
Timing							1	0.56
Compiler Error								

*Methodology*. We developed a classifier to automatically categorize the reviews into appropriate defect categories or topics, using natural language processing and machine learning. The patch description contains information about the code changes (i.e., patch) the developer submitted for review. The lines of code suggest the changes in the source file that were submitted as part of the patch. The review comments contain the textual feedback the reviewers provide on the code changes. We considered these three features from the code review repository to train our model.

Each review comment along with its patch description, line of code and the respective category (label) was considered a document. The patch description and review comment of the document were preprocessed by removing the stop words and stemming. We did not perform any processing of the line of code feature because they were programming syntax and we had to preserve the information as it is required to accurately identify the defect category. After preprocessing, we performed the term-weighting where we produced a dictionary from all of the terms in our document and assigned a unique integer Id to each term appearing in it using the *tf-idf* metric.

The model was trained on the 7 most common defect categories namely Visual Representation, Supported by Language, Solution Approach, Textual, Logic, Check and Organization because our dataset did not have enough samples for all of the 17 defect categories to train our model. The labels (topics) for each review were derived from our manual investigation (see Section III). Of the 468 review comments from our manual investigation across subject systems combined, we considered 240 review comments from three open source projects namely *eclipse*, *mylyn* and *android* to train our model.

After the model was trained, we tested it with our test data that consisted of 50 test cases. The test data consisted of only the patch description and the line of code that changed. The reason for that was to not include any forward looking information as they would invalidate the results. For example, review comments are available once the code review is already under way. Therefore, there might be very little benefit in predicting the review topics at that stage. We would like to predict the topics as early as possible and with as little information as possible. Similar to our training data, the test data also underwent preprocessing like removing stop words and stemming of the patch description. We fed the processed training data to our classifiers and predicted its performance

<sup>1</sup><https://git.eclipse.org/r/#/c/22722/4>

on test data. Once the automatic classification for the test data was performed, its predicted label was compared with the identified label and the accuracy was estimated. We adopted three different machine learning algorithms that are commonly used for text classification: KNN (K Nearest Neighbors) with a K value of 7, Naive Bayes and Support Vector Classification. We observed that KNN performed the best.

*Results:* With KNN, we observed an accuracy of 20% and an average precision and recall of 0.45 and 0.41 respectively. Table II shows the results of our automatic recommendation model. Our results indicate that we were able to predict both evolvability and functional types of defects. Also, we observe that the evolvability categories namely *Textual*, *Organization*, *Visual Representation* and *Solution Approach* have much more promising levels of precision and recall. Therefore, we surmise that automatic topic classification holds a much better promise in internal quality than external quality topics. Our effort is a first step in automating the topic identification as soon as a code change is submitted for review.

TABLE II  
RESULTS OF AUTOMATIC DEFECT CLASSIFICATION

Defect categories	Precision	Recall
<i>Textual</i>	0.73	0.50
<i>Supported by Language</i>	0.00	0.00
<i>Organization</i>	0.31	0.50
<i>Solution Approach</i>	0.27	0.80
<i>Visual Representation</i>	1.00	0.50
<i>Check</i>	0.00	0.00
<i>Logic</i>	0.60	0.38
<i>Avg/ Total</i>	0.45	0.41

*Application Scenario:* This investigation also gave us an insight on the reviewers' expertise in identifying a specific type of defect. Table III shows a list of all the reviewers that participated in reviewing the 108 review comments from *mylyn* project and the number of defects they identified under each category. It is evident that *Sam Davis* has actively participated in the code review process and he is more experienced in identifying defect categories like *Solution Approach* and *Textual*. On the other hand, with *Sam Davis* as the owner for two large patches #47888<sup>2</sup> and #61091<sup>3</sup> of the total 20 patches that were investigated in *mylyn*, there were only 2 defects identified on his patch which were of the type *Visual Representation* and *Check*. This suggests that active participation in code review helps in building knowledge, improves the quality of the developer and thereby significantly reduces the defect likelihood in one's code. This table also shows that many reviewers did not contribute enough during the code review process. However, this can be prevented by recommending appropriate reviewers i.e. reviewers who are capable of identifying specific defects in a patch, based on what defect types the patch is prone to.

Other application scenarios include prioritizing the code changes (patches) based on the topics of concern, predicting their acceptance likelihood and completion time, topic-specific knowledge transfer, and overall project's development and maintenance status and overall maturity.

<sup>2</sup><https://git.eclipse.org/t/#/c/47888/>

<sup>3</sup><https://git.eclipse.org/t/#/c/61091/>

TABLE III  
REVIEWERS AND CONTRIBUTED TOPICS IN MYLYN

Reviewer	Textual	Organization	Visual Representation	Solution Approach	Supported by Language	Check	Logic
<i>Sam Davis</i>	17	13		27	4	16	9
<i>Steffen Pingel</i>	2	1		3	1		
<i>London Buttersworth</i>	3						
<i>Frank Becker</i>			1				
<i>Doug Janzen</i>					1		
<i>Colin Ritchie</i>	2				2		
<i>Blaine Lewis</i>	2		2		1		

*Discussion:* We discuss the evolution of our automatic classifier. We made several attempts to build a reasonably strong model. In our first attempt, we considered the data collected from *eclipse*, *mylyn* and *android* projects from our manual investigation and considered all the 17 defect categories that were identified during manual analysis. The training data set consisted of 240 review comments and the test data set consisted of 50 review comments. To build our classifier, we chose the following features from *Gerrit*: patch description, file name as it would be a good analysis to see the defect type with respect to the file, owner name as it would give information about the quality of the developer, line of code that was changed in order to address the bug or implement a new feature and the review comment written for the specific line of change in the code. We processed the dataset to remove the stop words and stemming and then trained our classifier to predict the defect categories of the test dataset. We used KNN, Naive Bayes and Support Vector Classification for our classifier to automatically recommend the defect category. Our first attempt was not successful. The accuracy came out to be as low as 4%. The main reason was the insufficient dataset. The samples for each defect category in the dataset were not sufficient. Hence the model could not be trained well.

Because the size of dataset was small and the number of defect categories was large, for our second trial, we reduced the total number of defect categories by combining a few of the similar categories and eliminating some of the rare defect categories. We eliminated those defect categories that had very few occurrences in the entire dataset. For instance, we combined the categories *Check User Input* with *Check Function* as they are similar. *Check Function* checks for the return value of a function and *Check User Input* asks for a test case to verify the return value of a function. Similarly, we combined *Data and Resource Manipulation* with *Variable Initialization* as both of them are related to Resource management. The *Compiler Error* category is a very rare scenario because it is highly unlikely for one to submit a file with compilation errors for code review because of the automatic pre-checks typically in place. Only one review comment belonged to the *Compiler Error* category of all the 468 review comments that were investigated. Hence, we eliminated this category in the second round. In total, we considered a list of 13 defect categories for the second trial to train our model and they are as follows: *Check Function*, *Check Variable*, *Compare*, *Compute*, *Organization*, *Parameter*, *Resource*, *Solution Approach*, *Supported by Language*, *Textual*, *Timing*, *Visual Representation*, and *Wrong Location*. We considered the same set of features as in our first trial (namely patch description, owner name, file name, line of code and review comment). The training and test datasets were the same from the first trial (i.e 240 and 50 respectively). The only difference was with the number of target defect categories



on which the model was trained. This time we obtained an accuracy of 6%. The accuracy was still low because of the small dataset and also the noise introduced in the dataset in the form of file and owner names.

After these two unsuccessful trials, we realized that there was a need to further reduce the total number of defect categories. This time we considered only the most common defect categories for our classifier. We chose only those categories that were identified to be more prevalent in code review. In other words, we chose those target categories that had enough samples in our dataset. We also removed the features like file name and owner name which were mostly acting as noise in our dataset. In our third trial, we eliminated those features and considered only the patch description, line of code and review comment to train our classifier. This was the trial that gave a significant improvement in the accuracy, precision and recall and it was discussed in detail in the previous section.

For our next trial, we considered the data from all the four projects. The training data consisted of 360 records and the test data consisted of 108 records. We considered the same features as 22 our last trial (patch description, line of code and review comment). But this time, the average precision and recall fell from our previous attempt. Upon analyzing the results, we realized that the low accuracy in general was due to the following reasons. Firstly, there was insufficient data set and insufficient samples for each defect category. Secondly, the results of the manual investigation could not be verified by the concerned developer or the reviewer, hence the accuracy of the manual investigation results, on which the automatic classifier was built, could not be verified. Thirdly, our model provides good precision and recall to identify evolvability defects; however, there are not sufficient features in the code review tool to help us perform automatic classification of functional type of defects effectively. Another important reason is the difference in the line of code feature which is an important feature in our automatic model. *openstack* is a completely different project with python as its programming; *eclipse* and *mylyn* use the same programming language which is java, whereas *android* has both C++ and Java code.

## V. IMPACT ON REVISING CODE CHANGES

We wanted to investigate how developers receive the review feedback and what actions they take to improve their code changes. Was there any relationship between the topics they were critiqued on and the corresponding actions they took to resolve them. That is, we address:

*RQ 3:* How do the developers address the review comments that reviewers provide to revise their code changes?

By analyzing the code review comments, we identified that reviewers are more inclined towards identifying the internal design flaws in the code as it is evident from our manual investigation that 75% of the defects identified by the reviewers are about the internal quality of the code. As a next step, to validate these results, we wanted to study the steps that were taken to address those review comments. We carefully studied each revision of the patch, analysed the review comment and

therefore the underlying defect category and observed the developer's code change so as to address that defect for the following projects: *eclipse*, *mylyn* and *android*.

From our empirical study, we observed that developers incorporated specific refactorings that are congruent to the defect category of the review comments. For instance, in the review #22719<sup>4</sup>, the reviewer *Steffen Pingel* suggested in his review comment that a class be split into a separate class. This comment is clearly about an issue in the organization of the code as it is about rearranging the code such that the software is more comprehensible and maintainable, and hence can be categorized as *Organization*. If we observed the next revision of the patch, we see that the developer has extracted that part of the code and created a separate class as suggested by the reviewer. In this case, the developer has incorporated *Extract class* refactoring technique.

Similarly, in another review #60407<sup>5</sup>, the reviewer suggested a naming issue in a developer's code. This defect can be compared with the Textual defect category where emphasis is given to proper naming or comments in the code which otherwise can cause misleading information. In the next revision of the changed code (patch), we observed that the developer had changed the name of the method as per the reviewer's comments. This action can be compared to the *rename* refactoring method as it is about renaming a class or a variable or a method in order to make its purpose clear.

It can be seen from our observation that majority of the defects identified during code review are evolvability type of defects and in order to address those, developers adopted refactoring techniques. Since refactoring is a mechanism to improve the internal structure of the code, it is evident that code review is useful in improving the internal software quality or the evolvability of the software. Table IV shows the mapping between various defect topics and the refactorings adopted by the developers for each of those topics.

TABLE IV  
MAPPING BETWEEN TOPICS AND REFACTORINGS

Defect categories	Refactorings
<i>Textual</i>	Rename method
<i>Organization</i>	Extract method, Extract class, Move method
<i>Solution Approach</i>	Substitute Algorithm
<i>Supported by Language</i>	Hide Method

## VI. THREATS TO VALIDITY

We discuss internal, construct, and external threats to validity.

**Misclassification of review comments:** The results of the manual investigation could not be verified by the original developer/reviewer or other proficient software engineers thereby raising the risk of researcher's bias.

**Heterogeneous Dataset:** Although all the open source projects that we considered for our research uses the same code review tool and the mechanism, the projects themselves are different in nature (e.g., their main programming languages).

<sup>4</sup><https://git.eclipse.org/r/#/c/22719/1>

<sup>5</sup><https://git.eclipse.org/r/#/c/60407/3>

The line of code is a primary feature in our classifier, which could have impacted our results.

**Insufficient Dataset:** Since manual investigation is a tedious process, we could not gather enough data for all the defect categories. Hence our dataset was relatively small.

**Insufficient Features:** Code review tool has possibly insufficient features for our automatic recommendation model to classify functional defects effectively.

**Generalization:** Although we investigated four open source systems, we do not claim that our results would generalize to every single software system.

## VII. RELATED WORK

There have been many efforts on studying the effectiveness of code review in improving the external quality. However, very few efforts have been done to emphasize the importance of code review in improving the internal software quality.

Siy and Votta [15] proposed that 75 percent of the defects found during code reviews are evolvability defects that affect the evolution of the software instead of runtime behavior. C. Bird et al. [17] in their work, identified the factors that led to useful code review. They investigated the usefulness of code review by performing an empirical study in Microsoft projects, built and verified a classification model that can distinguish between useful and not useful code review feedback. Recently, McIntosh et al. [2] empirically showed that that poor code review negatively affect software quality. In another study, McIntosh et al. [18] reported that the percentage of reviewed changes a code component underwent correlates inversely to its chance of being involved in post-release fixes.

Rigby et al. [19] examined two peer review techniques: review-then-commit and commit-then-review used by Apache server project. They measured the frequency of reviews, the level of participation in reviews, and the size of artifacts under review in their studies. Beller et al. [20] found that the types of changes due to modern code review in Open source software are similar to those in the industry and academic systems from literature, featuring a similar ratio of maintainability-related to functional problems. Kemerer et al. [3] showed that code review reduces the amount of defects in student projects. With the available data they were also able to study the impact of review rate on the inspection performance. They found high review rates (i.e., a high number of reviewed LOC/hour) to be associated with a decrease in inspection effectiveness.

## VIII. CONCLUSIONS AND FUTURE WORK

We conducted an empirical study on the types of topics in the reviewers' feedback provided to developers on their code changes. Four open source systems were the subject of this investigation: *eclipse*, *mylyn*, *android* and *openstack*. Furthermore, we presented an automated approach to predict potential topics of reviewers' feedback as soon as a developer submits their code changes for review. Lastly, we also examined the impact of these review comments on the revisions that developers perform on their changes. We found that topics

relevant to both external and internal code qualities are discussed in code review; however, those on the internal quality are dominant. Also, developers use refactorings to address those topics. The specific refactorings seem to align with the specific nature of review feedback topics. In summary, we provide evidence of the benefits of code review on internal code quality. Our future work will be directed on improving the automatic detection of these topics (e.g., accuracy) and developing their applications to further empower the peer-code-review process.

To facilitate replication, among other things, we provide access to our online appendix <http://serl.cs.wichita.edu/codereview/topicmodel>.

## REFERENCES

- [1] B. Boehm and V. R. Basili, "Software defect reduction top 10 list," *Computer*, vol. 34, pp. 135–137, Jan. 2001.
- [2] S. McIntosh, Y. Kamei, B. Adams, and A. E. Hassan, "The impact of code review coverage and code review participation on software quality: A case study of the qt, vtk, and itk projects," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR 2014, pp. 192–201, 2014.
- [3] C. Kemerer and M. Paulk, "The impact of design and code reviews on software quality: An empirical study based on psp data," *Software Engineering, IEEE Transactions on*, vol. 35, pp. 534–550, July 2009.
- [4] O. Laitenberger, "Studying the effects of code inspection and structural testing on software quality," pp. 237–246, IEEE, 1998.
- [5] R. S. Arnold, "Software restructuring," vol. 77, pp. 607–617, Apr 1989.
- [6] "Refactoring: Improving the design of existing code," (Boston, MA, USA), Addison-Wesley Longman Publishing Co., Inc., 1999.
- [7] N. Gorla, A. C. Benander, and B. A. Benander, "Debugging effort estimation using software metrics," vol. 16, pp. 223–231, Feb 1990.
- [8] W. Li and S. Henry, "Object-oriented metrics that predict maintainability," vol. 23, pp. 111 – 122, 1993.
- [9] R. J. Miara, J. A. Musselman, J. A. Navarro, and B. Shneiderman, "Program indentation and comprehensibility," vol. 26, pp. 861–867, Nov. 1983.
- [10] P. W. Oman and C. R. Cook, "Typographic style is more than cosmetic," vol. 33, (New York, NY, USA), pp. 506–520, ACM, May 1990.
- [11] H. D. Rombach, "A controlled experiment on the impact of software structure on maintainability," vol. 13, (Piscataway, NJ, USA), pp. 344–354, IEEE Press, Mar. 1987.
- [12] T. Tenny, "Program readability: procedures versus comments," vol. 14, pp. 1271–1279, Sep 1988.
- [13] C. Bird and A. Bacchelli, "Expectations, outcomes, and challenges of modern code review," in *Proceedings of the International Conference on Software Engineering*, IEEE, May 2013.
- [14] P. C. Rigby and C. Bird, "Convergent software peer review practices," in *Proceedings of the the joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering (ESEC/FSE)*, ACM, 2013.
- [15] V. Balachandran, "Reducing human effort and improving quality in peer code reviews using automatic static analysis and reviewer recommendation," in *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pp. 931–940, 2013.
- [16] M. Mantyla and C. Lassenius, "What types of defects are really discovered in code reviews?," vol. 35, pp. 430–448, May 2009.
- [17] A. Bosu, M. Greiler, and C. Bird, "Characteristics of useful code reviews: An empirical study at microsoft," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, pp. 146–156, May 2015.
- [18] R. Morales, S. McIntosh, and F. Khomh, "Do code review practices impact design quality? a case study of the qt, vtk, and itk projects," in *Proc. of the 22nd Int'l Conf. on Software Analysis, Evolution, and Reengineering (SANER)*, pp. 171–180, 2015.
- [19] P. C. Rigby, D. M. German, and M.-A. Storey, "Open source software peer review practices: A case study of the apache server," in *Proceedings of the 30th International Conference on Software Engineering*, pp. 541–550, ACM, 2008.
- [20] M. Beller, A. Bacchelli, A. Zaidman, and E. Juergens, "Modern code reviews in open-source projects: Which problems do they fix?," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR 2014, pp. 202–211, ACM, 2014.

# Expediting Binary Fuzzing with Symbolic Analysis

Luhang Xu<sup>1</sup>   Wei Dong<sup>1\*</sup>   Liangze Yin<sup>1</sup>   Weixi Jia<sup>1</sup>   Yongjun Li<sup>2</sup>

<sup>1</sup> National University of Defense Technology, Changsha, China

<sup>2</sup> Northwestern Polytechnical University, Xi'an, China

\* Corresponding author

E-mail: *me@xuluhang.cn*, *{wdong, yinliangze}@nudt.edu.cn*,  
*wxjia92@163.com*, *lyj@nwpu.edu.cn*

## Abstract

*Fuzzing is an important method for binary vulnerability mining. It can analyze binary programs without the source code of the program, which is not easy to do by other technologies. But due to the blindness of input generation, binary fuzzing often falls into traps for a long time when the new mutated inputs cannot generate unexplored paths. In this paper, we propose an efficient and flexible fuzzing framework named Tinker. It defines the Growth Rate of Path Coverage to measure the current state of fuzzing. If the fuzzing falls into low-speed or blocked states, a symbolic analysis procedure is invoked to generate a new input which can help the fuzzing jump out of the trap. In the symbolic analysis procedure, we employ dynamic execution to track the traversed nodes. The untraversed branches are then identified according to the recorded data of AFL. At last, we employ CFG to construct complete paths to these branches and a new input is generated using symbolic execution. Tinker has been implemented and the experiments on DARPA CGC benchmark show that Tinker is more efficient in vulnerability mining than state-of-the-art binary vulnerability mining tools.*

## 1. Introduction

Fuzzing is a representative method for software vulnerability mining [1]. The basic idea of fuzzing is to provide a large amount of invalid, unexpected, or randomly generated data as inputs to a program. The program is then monitored for exceptions such as crashes. Given that fuzzing does not need the inside information of a program, it is one of the most important techniques for binary program analysis.

Existing fuzzing techniques can be classified into two classes: black box fuzzing and white box fuzzing. Black box fuzzing [2] does not require any source information of the program. Researches on this technique mainly focus on effective inputs generation. White box fuzzing [3][4] has been widely studied in recent years. It usually combines fuzzing with other analysis methods such as symbolic execution [5] and dynamic blot analysis [6]. To improve the capability of vulnerability mining, white box fuzzing usually explores the inside structures of a program to guide the fuzzing process.

Fuzzing has many advantages compared with other vulnerability mining techniques. However, due to the blindness of input generation, traditional fuzzing often falls into traps

in which most of the executed paths are redundant [7][8]. To mitigate this problem, recently there has been some work using other analysis methods such as symbolic execution [9] to improve the efficiency of fuzzing. However, there are still two weaknesses for existing work: 1) Existing work is not sensitive enough to the current state of fuzzing. They usually invoke symbolic execution when fuzzing is in blocked states, without considering those low-speed states in which fuzzing may explore just several new paths for a long time. In practice, these states might be the most cases. 2) Existing work cannot handle system calls properly. As a result, they can only be applied to simple environments rather than real programs. For example, Driller [9] can run just on a simplified operating system (OS) with only seven system calls [10].

This paper proposed a new binary fuzzing approach Tinker. It first employs the defined Growth Rate of Path Coverage (GRPC) to evaluate the efficiency of fuzzing in current state. If fuzzing is trapped into a low-speed or blocked state which cannot efficiently find new paths, a symbolic analysis procedure will be invoked to generate a new valid input of the program, such that fuzzing can jump out of the trapped state and again run in a high-speed state. We employ a tool Angr for CFG generation, which has rewritten most of the system calls properly. Our method can be applied to real programs. Moreover, as our unexplored paths are analyzed from the CFG of the target binary program, the input generated in our method is often more effective.

The main contributions of this paper include:

- We proposed an efficient and flexible binary vulnerability mining approach Tinker. It employs GRPC to evaluate the current state of fuzzing. If fuzzing falls into a trap, a symbolic analysis procedure is invoked to generate an input which helps fuzzing jump out of the trap.
- We proposed an effective symbolic analysis based new input generation method. It supports real binary program analysis and can always return an effective input to an unexplored path. CFG is used in this method to construct the paths to those untraversed branches.
- We implemented Tinker and evaluated it on DARPA CGC benchmark. The experimental results demonstrate that, compared with state-of-the-art tools, Tinker can detect more binary vulnerabilities and is often more efficient for unexplored path exploration.

The rest of this paper is organized as follows. Section 2 outlines the framework of Tinker. Section 3 and 4 present our GRPC-based fuzzing evaluation and symbolic analysis based fuzzing intervention methods, respectively. Section 5 provides the experimental results. Section 6 reviews the related work, and Section 7 concludes our paper.

## 2. Method Overview

In this section, we first discuss the challenge of binary fuzzing, and then we outline the framework of Tinker.

### 2.1. The Challenge of Binary Fuzzing

Consider the example shown in Figure 1. The program contains two strings *order* and *magic\_byte*. It runs the *fault()* statement only if both *order* and *magic\_byte* equal *"vuln"*. For this example, fuzzing will enter the true branch of Line 4 only if it succeeds in mutating the value of *magic\_byte* to *"vuln"*. Given that *"vuln"* has  $4 * 8$  bits, a maximum number of  $2^{4*8}$  variations of inputs are required for fuzzing to reach Line 6. For some other long string, it will be more difficult for fuzzing to enter the true branch. A string like *"vuln"* here that hampers fuzzing to new paths in mutation is called "MAGIC-BYTE". The similar problem occurs in Line 7. In this paper, we use the growth rate of path coverage (GRPC) to measure the efficiency of fuzzing. For this example, the GRPC will keep zero until the true branches of Line 4 and Line 7 are explored.

```

1  int main(void){
2  char order[20], magic_byte[20];
3  scanf("%s\n", magic_byte);
4  if(strcmp(magic_byte, "vuln")==0)
5  {
6      gets(order);
7      if(strcmp(order, "vuln")==0)
8          fault();
9  }
10 return 0;
11 }

```

Figure 1: A challenging example for traditional fuzzing.

To further investigate this problem, we have analyzed the efficiency of traditional fuzzing for three real binary programs, as shown in Figure 2. The vertical axis  $P$  represents the number of new explored paths for each time unit (15 minutes). According to this figure, fuzzing might perform differently for different programs. 1) For the *Multipass* program, the value of  $P$  always keeps 40 on average, and we say that fuzzing is in "high-speed" state. 2) For the *Grisword* program, the value of  $P$  always keeps in just 10 on average after 4 time units, and we say that fuzzing is in "low-speed" state. 3) For the *Monster\_Game* program, the value of  $P$  reduces to zero after 3 time units, and we say that fuzzing falls into a "blocked" state.

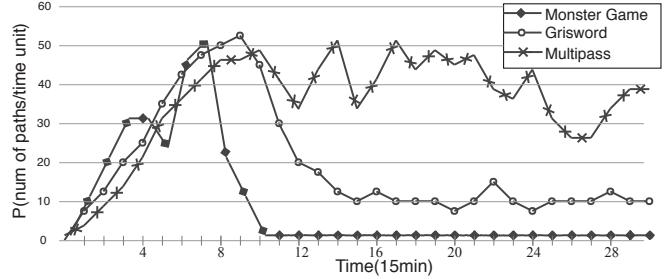


Figure 2: Efficiency of fuzzing for three real programs.

### 2.2. Framework of Tinker

When fuzzing falls into a blocked or low-speed state, it is usually difficult for fuzzing to jump out of such a state itself. In other words, it might be trapped into such a state for a long time, which significantly limits the efficiency of fuzzing. To deal with this problem, we proposed a new binary fuzzing approach called Tinker. The idea is that we continuously compute the value of GRPC during the fuzzing process to evaluate the current state of fuzzing. If we find that fuzzing is trapped into some blocked or low-speed state, we employ a symbolic analysis method to generate a new input to guide fuzzing to search the other untraversed branches, such that fuzzing always explores new paths and keeps in high-speed states for vulnerability mining.

The framework of Tinker is outlined in Figure 3. Before the fuzzing starts, the target binary program will be pre-processed to generate an instrumented binary and its control flow graph (CFG). During the process of fuzzing, we iteratively employ a GRPC-based fuzzing evaluation procedure to measure the current state of fuzzing. If the fuzzing falls into a low-speed or blocked state, we then invoke the symbolic analysis based fuzzing intervention process to generate a new input to motivate the fuzzing to a high-speed state. To generate the new input, we first select an input from fuzzing which leads to a redundant path. The instrumented binary is then executed with this input to obtain the traversed nodes of the corresponding path. According to the recorded data of AFL, we can obtain those untraversed opposite branches of the path. CFG is then employed to obtain the complete paths of these untraversed branches. At last, we select one of these paths and employ symbolic execution to generate a new input. In such a manner, whenever the fuzzing falls into a low-speed or blocked state, we can detect it in time and generate a new input that helps fuzzing jump out of the trap. In particular, in Tinker, fuzzing and symbolic-based analysis work in parallel. The intervention of symbolic-based analysis does not interrupt Fuzzing.

## 3. GRPC-based Fuzzing Evaluation

If the fuzzing falls into a blocked or low-speed state, it may spent most of its effort on iteratively exploring those redundant paths. To quantitatively evaluate the efficiency of fuzzing, we proposed the notion of Growth Rate of Path Coverage (GRPC), which defines the average number of new ex-

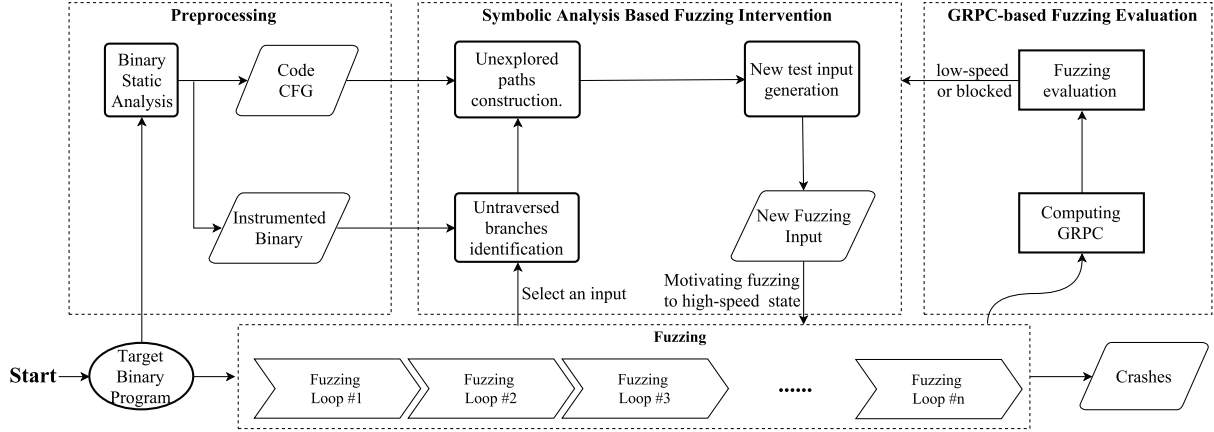


Figure 3: Framework of Tinker.

explored paths that grows per time unit over the latest period. The smaller the value of GRPC is, the lower the efficiency of fuzzing will be. If GRPC equals 0, then the fuzzing is trapped into a blocked state.

To compute GRPC, we divide the fuzzing process into intervals. Then the average number of new explored paths per time unit for the last interval can be obtained as equation (1), where  $q_m$  represents the total number of different paths that have been explored in the last  $m$  intervals, and  $\Delta t$  represents the time length of each interval. The value of  $q_m$  can be obtained from AFL directly.

$$p_m = \frac{q_m - q_{m-1}}{\Delta t}, \quad m \geq 1 \quad (1)$$

Given that there may exist some noises sometimes, one interval might not reflect the actual state of fuzzing in practice. For example, the fuzzing might fall into a low-speed state in some interval, and have a large probability of returning back to a high-speed state in the next interval. To avoid such cases, we filter out those noises by using the idea of moving average [11]. In this idea, the  $m$ th sample result is an average of the  $2n + 1$  sample values on time points  $m - n, m - n + 1, \dots, m + n - 1$  and  $m + n$ . Similarly, we can employ the  $p_m$ 's of the last  $2n + 1$  intervals to compute our GRPC  $P_m$ , as shown in equation (2).

$$P_m = \frac{1}{2n + 1} \sum_{k=0}^{2n} p_{m-k}, \quad m \geq 2n \quad (2)$$

Now the problem is how to decide the value of  $n$ . If  $n$  is too big, the fuzzing will stay in those blocked and low-speed states for a long time; if  $n$  is too small, many of those noises might not be filtered out in our method, which may also decrease the efficiency of fuzzing. According to the work of moving average, we usually set  $n = 2$  in practice [11]. Our experiments also proved that fuzzing can usually obtain the best efficiency for  $n = 2$ . For  $m < 2n$ , we use the average of all  $p_k$  for  $k \leq m$  to compute  $P_m$ .

After obtaining GRPC, we need to decide the current state of fuzzing. For any program, the GRPC  $P_m$  have a peak

value  $P_{max}$ , which can be obtained from its execution history. We say that fuzzing obtains a highest efficiency if  $P_m$  equals  $P_{max}$ . Given that the values of  $P_{max}$  for different programs differ significantly, the threshold that determines the state of fuzzing should also be different for different programs. Our idea is to decide the threshold according to the value of  $P_{max}$  for each program. To this end, we can decide the current state of fuzzing using the following definition.

**Definition 1** Suppose that the currently observed peak value of GRPC is  $P_{max}$ ,  $w$  is a threshold factor. Then the current state of fuzzing is determined as follows. Fuzzing is at

- 1) high-speed state, if  $P_m \geq P_{max} * w$ ;
- 2) low-speed state, if  $0 < P_m < P_{max} * w$ ;
- 3) blocked state, if  $P_m = 0$ .

At the beginning of fuzzing,  $P_{max}$  can be set to a small value that is greater than zero. After each interval,  $P_{max}$  will be updated to  $P_m$  if current  $P_m$  is larger. Threshold factor  $w$  can be customized by user according to the performance of fuzzing. In our experiments, we tried several values and found that setting  $w = 0.4$  can usually obtain higher efficiency. If fuzzing is at low-speed or blocked state, a symbolic analysis procedure will be started to generate a new input that enables fuzzing to jump out of the trap.

#### 4. Symbolic Analysis Based Fuzzing Intervention

The purpose of symbolic analysis based fuzzing intervention is to generate a test input which can help the fuzzing to explore new branches. The idea is that we first select an untraversed branch and then use symbolic execution to generate an input to this branch. However, given that we have no source information of the target binary program, a problem is how to obtain the path of an input and generate the path condition constraints for an untraversed branch. To address this problem, we propose to use existing disassemble techniques to generate the CFG of the target binary program first, and then we use existing binary instrumentation tools to generate

an instrumented binary which can obtain the execution path of an input. With the instrumented binary, we use dynamic execution to identify a path explored by fuzzing. According to the recorded data of AFL, we can obtain the untraversed opposite branches of the path. Then CFG is employed to generate the condition constraints of these branches, which are then solved by a constraint solver to generate valid inputs to these branches. The main steps of our symbolic analysis method are as follows.

1) **Instrumented binary and CFG generation.** To perform our symbolic analysis, we should generate an instrumented binary and the CFG of the target binary program first. For instrumented binary, dyninst [12] is a binary instrumentation tool satisfying all our requirements. Hence, we select dyninst for instrumented binary generation in our work. For CFG generation, an issue here is how to generate those system call nodes which cannot be disassembled directly. We found that Angr [13] can rewrite most of the system calls and generate a CFG with system call nodes, which makes our method work well for most real programs. Hence, we select Angr to perform the CFG generation.

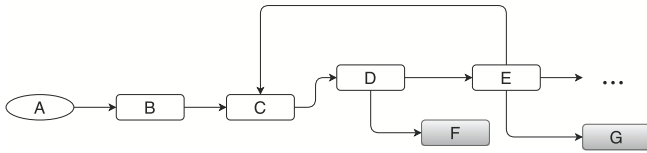


Figure 4: An example CFG used in symbolic analysis

2) **Untraversed branches identification.** Dynamic execution is a technique which can track an instrumented program under an input and record the traversed nodes. To obtain an untraversed branch, we select an input from fuzzing, and then use dynamic execution to obtain the traversed path of the input. Then for each branch of this path, we will check if the opposite branch has been explored yet. In AFL, it has recorded all the traversed nodes during the fuzzing process. If an opposite branch has not been explored yet, then an untraversed branch has been found. We will generate an input to this branch in the following steps. Given that some branch may be unreachable in a real execution, we will obtain all untraversed branches of the path in this step.

3) **Unexplored paths construction.** To generate the input of an untraversed branch, we should identify the path to it and then generate the condition constraint of the path. To achieve this, we map the path to the CFG, and then construct a subgraph of the CFG which contains just nodes of the path. For the example shown in Figure 4, the subgraphs for  $F$  and  $G$  in Figure 4 can be represented by  $G_F = \{A, B, C, D, E, F\}$ , and  $G_G = \{A, B, C, D, E, G\}$ , respectively. An issue here is that dynamic execution does not consider those system calls. In other words, there are some system call nodes which have been traversed by fuzzing but not marked as traversed in the CFG yet. We should detect these nodes and mark them as traversed in the CFG before the subgraph generation.

4) **New test input generation.** In this step, we will gen-

---

**Algorithm 1** Symbolic analysis based test input generation

**Input:** The binary  $bin$ , selected input  $fin$  from fuzzing.

**Output:** New test input  $testcase$ .

```

//Instrumented binary and CFG generation
ins_bin = Instrument(bin);
cfg = CreateCFG(bin)
//Untraversed branches identification;
path = DynamicExecution(ins_bin, fin);
n_list = UntraversedBranches(path);
//Unexplored paths construction;
PathMap(path, cfg);
MissedNodesAdd(cfg);
g_list = ∅;
for each untraversed branch  $n \in n\_list$  do
    g_list = g_list ∪ PathConstruct(n);
end for
//New test input generation;
for each subgraph  $g \in g\_list$  do
    cons = ConstraintGeneration(g);
    input = Solve(cons);
    if input is VALID then
        RETURN testcase;
    end if
end for
RETURN NULL ;
  
```

---

erate a new test input which explores an unexplored path and helps the fuzzing jump out of a trapped state. The idea is that for each subgraph we have constructed, we first construct a constraint formula according to it. The constraint formula is then solved by a constraint solver. In this paper, Z3 is employed as our constraint solver. Given a constraint, it can generate an input of the program if the constraint is satisfiable. Setting this input as the input of the binary program will generate a path to the untraversed branch. If Z3 returns unsatisfiable, then the selected path is invalid and we will generate the new input according to another unexplored path.

The whole procedure of our test input generation process is demonstrated in Algorithm 1. The input is a target binary program  $bin$ , and an input  $fin$  selected from fuzzing. We first use dyninst and Angr to obtain the instrumented binary  $ins\_bin$  and the CFG  $cfg$ , respectively. Then the function *DynamicExecution* is invoked to obtain the execution path of  $fin$ . The function *UntraversedBranches* identifies those untraversed opposite branches of the path according to the recorded data of AFL and stores them in  $n\_list$ . To generate the condition constraints of these untraversed branches, the path is mapped to  $cfg$ . The function *MissedNodesAdd* detects those missed system call nodes and adds them to the path in the CFG. For each untraversed branch  $n$ , *PathConstruct* constructs a subgraph which contains all nodes of the path leading to  $n$ . Given a subgraph, the function *ConstraintGeneration* generates a constraint for-

mula, which is then solved by Z3 using the function *Solve*. If a valid input is generated, then we have obtained a new test input and the procedure terminates. Otherwise, the selected untraversed branch is unreachable and we need to continue the loop and analyze another untraversed branch.

## 5. Experimental Evaluation

We implemented Tinker based on the open source tools AFL 2.33b [14], Angr [13], dyninst [12], and Z3 [15]. AFL is used as the fuzzer in Tinker and we extended it with an information collection interface to dynamically obtain the number of explored different paths to compute GRPC. Angr is a python-based binary analysis framework integrating a variety of existing analytic techniques. In Tinker, it is used to translate the target binary program into an intermediate representation, and generate its CFG. Dyninst is used to generate the instrumented binary and Z3 is used as the solver to generate a new input of an unexplored path.

Our experiments were performed on a machine with four Intel Core i7-6700 cores and 16GB memory. The operating system is 64-bit Ubuntu 16.04 LTS. Although AFL supports parallelism, it will produce deviations and influence the comparison results. Hence, we used only one fuzzing process in the experiments.

We use the DARPA CGC sample binaries [16, 17] as our benchmark to evaluate the vulnerability mining capability of our method. However, these binaries can only run under the DARPA Experimental Cyber Research Evaluation Environment (DECREE), which is a simplified OS with only seven system calls. Tinker aims to find vulnerabilities in real-world binaries. Therefore, we select the data set CB-multios provided by TrailofBits team [18], who migrated the DARPA CGC benchmark from DECREE to Linux.

For the 244 binary programs in CB-multios, we filtered some special types of programs that could not be used in fuzzy test analysis.

We performed our experiments on the remaining 211 programs. To evaluate the efficiency of our method, we compared the results of Tinker with that of Angr and AFL-qemu, which employ symbolic execution and pure fuzzing techniques for vulnerability mining, respectively. Both of them are state-of-the-art binary analyzers. We use two hours as the time limit. For the symbolic execution in Angr and Tinker, the depth of loop exploration is limited to 500 to avoid explosion. The more programs can be triggered to crash, the stronger a tool’s vulnerability mining ability will be.

The experimental results are shown in Figure 5. Tinker detected 116 vulnerable programs, AFL found 96 vulnerable programs, and Angr found only 47 vulnerable programs. Angr found 10 vulnerable programs which were not detected by AFL. Tinker found all the 106 vulnerable programs that AFL and Angr have found, and it found 10 more vulnerable programs which were not detected by either Angr or AFL. These results suggest that under the same time limitation and conditions, Tinker can usually find more program crashes than existing binary fuzzing and symbol execution

techniques.

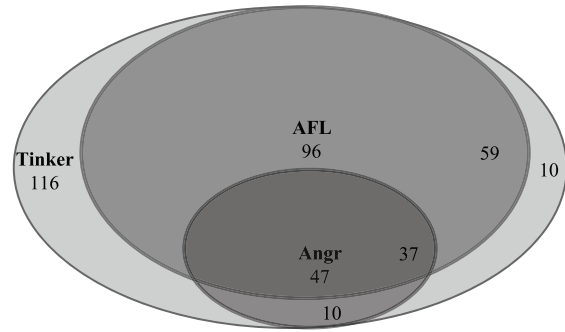


Figure 5: The crashed programs found by Tinker, AFL and Angr in DARPA CGC benchmark.

To further analyze the efficiency benefited from our method, we compare the number of explored paths within two hours between Tinker and the traditional fuzzing which contains no fuzzing intervention. All the 116 vulnerable programs that have been detected by Tinker are considered as the benchmark. The comparison is shown in Figure 6.

In this figure, the examples are divided into 3 categories based on the improvement rate. The Y-axis represents the number of examples for each category. From this figure, our method explores more paths than traditional fuzzing for all examples. Particularly, our method searched more than 20% of paths for 26 examples, searched 10%-20% more paths for 36 examples, and searched less than 10% more paths for 54 examples. We further studied the programs in which the efficiency is improved less than 10%. Most of these programs contain simple string comparisons which can be well handled by traditional fuzzing. For those programs with complex structures, our method can usually obtain more than 20% of the efficiency. In this sense, Tinker is more suitable to deep vulnerabilities in practice.

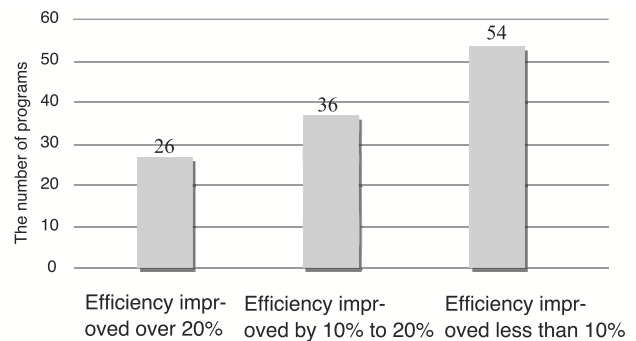


Figure 6: Efficiency benefited from fuzzing intervention.

## 6. Related Work

A variety of binary fuzzers have been proposed in recent years. Most of them focus on seed generation. Some fuzzers try to generate highly structured input, such as Skyfire [19]. The others mainly aim to improve the efficiency of fuzzing, such as AFL-lafintel [20], which converts a program into

LLVM IR [21]. Moreover, most of the fuzzers need to work at the source level. Although there exist some fuzzers such as AFL-qemu, AFL-dyninst [22] and Aflpin [23], which support binary programs without source code, their efficiency for vulnerability mining still requires further improvement.

Symbolic execution is one of the most successful techniques for binary program analysis. Tools adopting this technique include Angr [13], Mayhem [24] and S2E [25]. The main problem for this technique is the path explosion problem. In order to mitigate this problem, various approaches have been proposed (e.g. veritesting [26]). However, existing techniques and tools are still not effective enough for real binary program analysis.

To address the problems faced by above methods, white box fuzzing has been proposed and widely studied in recent years. It explores analysis methods to guide fuzzing to generate more effective test cases (e.g. Steelix [27] and Vuzzer [28]), such that the path coverage of fuzzing can be improved. However, due to the high frequency of additional analysis invoking, this technique usually brings heavy load.

Driller [9] is the closest work to Tinker. If fuzzing falls into a blocked state, it uses concolic execution to guide fuzzing to a new path. Compared with Driller, Tinker has a more efficient and flexible efficiency evaluation method which makes it more sensitive to the current state of fuzzing. In addition, Tinker employs Angr for CFG generation which has rewritten most of the system calls and can generate a CFG with those system call nodes. Hence, Tinker works well for real binary programs, while Driller can only be applied to binaries running on DEGREE.

## 7. Conclusions

This paper presents a new fuzzing framework named Tinker. Unlike traditional white box fuzzing techniques, Tinker uses GPRC to measure the current work state of fuzzing, and invokes a symbolic analysis approach when fuzzing is in blocked or low-speed states. In the symbolic analysis approach, new input is generated using symbolic execution to guide fuzzing jump out of the trap. Tinker is implemented based on open source tools AFL, Angr, dyninst and Z3. Experiments on DARPA CGC benchmark show that Tinker has higher efficiency in vulnerability mining than the other state-of-the-art fuzzing tools such as AFL and Angr. In future work, we will try to integrate existing efficient symbolic execution algorithms in to our approach so that the vulnerabilities in more complex binary programs can be detected.

## 8. Acknowledgement

We would like to thank the authors of ANGR and AFL for opening their source code. This work is funded by National Natural Science Foundation of China (No.61690203, No.61532007), and 973 National Program on Key Basic Research Project of China (No.2014CB340703).

## References

[1] M. Sutton, A. Greene, and P. Amini, *Fuzzing: brute force vulnerability discovery*. Pearson Education, 2007.

[2] M. Woo, S. K. Cha, S. Gottlieb, and D. Brumley, "Scheduling black-box mutational fuzzing," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013.

[3] P. Godefroid, A. Kiezun, and M. Y. Levin, "Grammar-based whitebox fuzzing," in *ACM Sigplan Notices*, vol. 43, no. 6. ACM, 2008.

[4] V. Ganesh, T. Leek, and M. Rinard, "Taint-based directed whitebox fuzzing," in *Proceedings of the 31st International Conference on Software Engineering*. IEEE Computer Society, 2009.

[5] J. C. King, "Symbolic execution and program testing," *Communications of the ACM*, vol. 19, no. 7, pp. 385–394, 1976.

[6] J. Newsome and D. Song, "Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software," 2005.

[7] U. Kargén and N. Shahmehri, "Turning programs against each other: high coverage fuzz-testing using binary-code mutation and dynamic slicing," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. ACM, 2015, pp. 782–792.

[8] "American Fuzzy Lop," <http://lcamtuf.coredump.cx/afl/>.

[9] N. Stephens, J. Grosen, C. Salls, A. Dutcher, R. Wang, J. Corbetta, Y. Shoshitaishvili, C. Kruegel, and G. Vigna, "Driller: Augmenting fuzzing through selective symbolic execution." in *NDSS*, vol. 16, 2016, pp. 1–16.

[10] "Cyber Grand Challenge," <http://archive.darpa.mil/cybergrandchallenge/about.html>.

[11] S. L. Marple and S. L. Marple, *Digital spectral analysis: with applications*. Prentice-Hall Englewood Cliffs, NJ, 1987, vol. 5.

[12] "Dyninst api." <http://www.dyninst.org/dyninst/>, 2005.

[13] "Angr," <http://angr.io/index.html>.

[14] "AFL-qemu," [http://lcamtuf.coredump.cx/afl/technical\\_details.txt](http://lcamtuf.coredump.cx/afl/technical_details.txt).

[15] L. De Moura and N. Bjørner, "Z3: An efficient smt solver," in *International conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2008, pp. 337–340.

[16] J. Song and J. Alves-Foss, "The darpa cyber grand challenge: A competitor's perspective," *IEEE Security & Privacy*, vol. 13, no. 6, 2015.

[17] "CyberGrandChallenge samples," <https://github.com/CyberGrandChallenge/samples>.

[18] "DARPA Challenge Binaries on linux and os x," <https://github.com/trailofbits/cb-multios/>.

[19] J. Wang, B. Chen, L. Wei, and Y. Liu, "Skyfire: Data-driven seed generation for fuzzing," in *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 2017, pp. 579–594.

[20] "Circumventing Fuzzing Roadblocks with Compiler Transformations," <https://lafintel.wordpress.com/2016/08/15/circumventing-fuzzing-roadblocks-with-compiler-transformations/>.

[21] C. Lattner and V. Adve, "Llvm: A compilation framework for life-long program analysis & transformation," in *Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization*. IEEE Computer Society, 2004.

[22] "AFL-dyninst," <https://github.com/vrtadmin/moflow/tree/master/afl-dyninst>.

[23] "AFLPIN," <https://github.com/mothran/aflpin>.

[24] S. K. Cha, T. Avgerinos, A. Rebert, and D. Brumley, "Unleashing mayhem on binary code," in *Security and Privacy (SP), 2012 IEEE Symposium on*. IEEE, 2012, pp. 380–394.

[25] V. Chipounov, V. Kuznetsov, and G. Candea, "S2e: A platform for in-vivo multi-path analysis of software systems," *ACM SIGPLAN Notices*, vol. 46, no. 3, pp. 265–278, 2011.

[26] T. Avgerinos, A. Rebert, S. K. Cha, and D. Brumley, "Enhancing symbolic execution with veritesting," in *Proceedings of the 36th International Conference on Software Engineering*. ACM, 2014.

[27] Y. Li, B. Chen, M. Chandramohan, S.-W. Lin, Y. Liu, and A. Tiu, "Steelix: program-state based binary fuzzing," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. ACM, 2017, pp. 627–637.

[28] S. Rawat, V. Jain, A. Kumar, L. Cojocar, C. Giuffrida, and H. Bos, "Vuzzer: Application-aware evolutionary fuzzing," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2017.



# Topic Modeling for Noisy Short Texts with Multiple Relations

Chiyu Liu, Zheng Liu, Tao Li and Bin Xia

Jiangsu Key Laboratory of Big Data Security & Intelligent Processing  
School of Computer Science, Nanjing University of Posts and Telecommunications  
Nanjing 210023, People's Republic of China  
{1016041229, zliu, towerlee,bxia}@njupt.edu.cn

**Abstract**—Understanding contents in social networks by inferring high-quality latent topics from short texts is a significant task in social analysis, which is challenging because social network contents are usually extremely short, noisy and full of informal vocabularies. Due to the lack of sufficient word co-occurrence instances, well-known topic modeling methods such as LDA and LSA cannot uncover high-quality topic structures. Existing research works seek to pool short texts from social networks into pseudo documents or utilize the explicit relations among these short texts such as hashtags in tweets to make classic topic modeling methods work. In this paper, we explore this problem by proposing a topic model for noisy short texts with multiple relations called MRTM (Multiple Relational Topic Modeling). MRTM exploits both explicit and implicit relations by introducing a *document-attribute distribution* and a *two-step random sampling strategy*. Extensive experiments, compared with state-of-the-art topic modeling approaches, demonstrate that MRTM can alleviate the word co-occurrence sparsity and uncover high-quality latent topics from noisy short texts.

**Index Terms**—Topic Modeling · Multiple Relations · Short Texts · Social Analysis

## I. INTRODUCTION

Topic modeling based on probabilistic graphical models with latent variables for uncovering hidden thematic structures is widely used in various applications including content recommendation [1], user profiling [2], trend detection [3], etc. Short texts, especially ones from social networks, tweets, feature short length, inordinate structure, and colloquialism. As a result, uncovering the potential topics from these short texts is not an easy task [4], [5]. Take the short texts from popular social networks such as Twitter as an example. Tweets are short, informal, and lack regular patterns, which leads to poor performance of the classic topic model Latent Dirichlet Allocation (LDA) [6], as well as many other LDA-like topic models. The underlying reason is that there are not enough word co-occurrence instances due to term sparsity in short texts.

Pooling tweets or other short texts [7] by aggregating them into pseudo documents based on their attributes has been proved to be a promising way to improve the quality of topics found by LDA-like methods. Possible attributes could be the authors of short texts [8], [9], or time periods. For tweets from social networks, hashtags [4] or burst scores

[10] could also serve as the aggregation cornerstones. These pseudo documents by aggregating short texts enrich the word co-occurrence, but on the other hand, they not only bring duplicates of short texts containing multiple attributes as well as new co-occurrence instances which do not exist, but also ignore the relations among these attributes. For example, each tweet could contain more than one hashtag, so some hashtags are likely to appear together than other hashtags. The hashtag correlations are the bridges of words in different short texts, and could help to improve the quality of topics. Wang et al. [11] proposed a Hashtag Graph based Topic Model (HGTM) for tweets, in which user-contributed hashtags are considered in the generative process of tweets. Experimental results show that it is more reliable than the simple pooling strategy.

It is not difficult to see that there are multiple attributes in short texts. Besides the explicit attributes like hashtags or users, there are many other implicit attributes such as various kinds of entities and temporal attributes. In this paper, we use labels to denote the possible values of attributes. For example, if the attribute is actors, then the corresponding labels are actor names. Tweets discussing movies could contain entities like actors, directors, as well as movie genres, movie released date, etc. Each of these attributes could be represented by a relational graph, where a vertex is a possible label of the attribute and an edge indicates the co-occurrence relations between two labels, which we will explain in detail in Section III. These relational graphs of attributes could reveal the semantic associations between labels. On the other hand, unnatural co-occurrence words about the background of short texts are noises, which impact the topic quality. For example, if the short texts are about some movies, then words like movie, film or cinema are not discriminative, while they exist in informal oral presentation [9]. Traditional solutions like TF-IDF can not handle well in short texts. Noisy words are highly related to the domain knowledge of the short texts, and not helpful in understanding the corpus.

With above observations, in this paper, we propose a topic model for noisy short texts with multiple relations, called MRTM, which can uncover meaningful topics. The main idea is to incorporate multiple relations into the generative process of short texts to produce high-quality topics measured both subjectively and objectively. Gibbs Sampling [12] is adopted to estimate the parameters in MRTM. The main contributions of this paper are summarized below.

\* Corresponding author: Zheng Liu

DOI reference number: No.10.18293/SEKE2018-022

- The proposed MRTM alleviates the sparsity of word co-occurrence in short texts by incorporating multiple relations into the generative process, resulting in a coherent generative topic model.
- MRTM further improves the quality of the uncovered topics by removing unnatural word co-occurrence instances caused by considering weakly-supervised relations.
- Extensive experiments are conducted on real data sets crawled from Microblog. The experimental results are carefully analyzed, showing that MRTM can uncover coherent topics of higher quality than the start-of-the-art approaches.

The rest of this paper is organized as follows. Section II discusses the related work. Section III describes the proposed multiple relational topic model in detail, as well as the inference process of its parameters. Section IV presents the experimental results and finally, Section V concludes the paper with possible future directions.

## II. RELATED WORK

Classic topic models such as Latent Dirichlet Allocation (LDA) [6] suffers term sparsity and noises when applied to short texts, resulting in low-quality topics due to insufficient word co-occurrence instances. Many researchers studied this problem, where the approaches can be categorized as follows.

### Pooling based strategy

Pooling short texts by aggregating them into pseudo documents based on certain attributes [8], [9], [10] has been proved to be a promising way to make LDA-like approaches work. Zhao et al. [9] analyzed the internal characteristic of short texts by introducing topic category and background words. In particular, they found users' topics are concentrated and consist of only a few words. Mehrotra et al. [4] analyzed the extensive experimental results of various pooling attributes and found that not all pooling attributes are helpful in capturing high-quality topics.

### Semantic based strategy

Using word semantics as the prior knowledge could benefit topic modeling for short texts, where the prior knowledge is pre-trained word embedding based on the large corpus. Li et al. [13] proposed a topic model based on the Dirichlet Multinomial Mixture (DMM), which is able to find more semantically related word pairs under the same topic during the sampling process. Similarly, Nguyen et al. [14] incorporated vector representations of words during topic modeling to improve the word-topic mapping. The vector representations of words are learned by using a large corpus.

### Relation based strategy

Recently, many efforts [11], [15], [16] were put on constraining LDA with semi-structured relations. By integrating this kind of prior knowledge, the generative processes in these models for short texts are more reasonable. Rosen et al. [8] proposed an author-topic model (ATM) for documents to include authorship information. Each author is associated with a topic distribution. However, it is not appropriate for short texts. Daniel et al. [15] proposed Labeled LDA to learn one-to-one mappings between topics and labels, but they

ignored the correlations between labels. Labeled-LDA is a strong supervised model that labels have equal impact on short texts. Wang et al. [11] improved the graphical models in LDA family with hashtag graph based topic model (HGTM). Unlike pooling based strategy in which tweets are aggregated into pseudo documents, it incorporated a hashtag graph into topic modeling. The key differences between HGTM and MRTM in this paper could be explained in terms of both complexity and robustness. HGTM only considers a single explicit relation, i.e., hashtags, while MRTM incorporates multiple relations into topic modeling. Moreover, MRTM integrates a new hidden variable which makes it more robust than HGTM. Note that multiple relations represented by information networks have various inherent textual information. Their rich semantics could enhance the inherent coherence among texts, and as a result, MRTM could uncover topics with better quality which is shown in Section IV.

## III. THE PROPOSED APPROACH

### A. Notations

Let  $D = \{d_1, d_2, \dots, d_m\}$  denote the corpus of short texts where corpus and  $d_i$  is a short text. Let  $W = \{w_1, w_2, \dots, w_n\}$  denote the vocabulary set of  $D$ , where  $w_i$  is a word. Let  $C = \{c_1, c_2, \dots, c_k\}$  denote the attribute sets where  $c_i$  is an attribute. In the following of this paper, we also use  $c$  to denote a certain attribute.

Take the tweets of movie reviews as an example. Fig. 1 shows the overall concept of multiple relational graphs. Recall that labels denote values of a certain attribute. A tweet could contain hashtags, as well as many other entities, such as title, actors, released time, etc., as shown in Fig. 1. Then we can construct a relation graph  $g^c = (V_c, E_c)$  for each attribute  $c$ , where  $V_c$  is the vertex set in which each vertex represents a label belong to attribute  $c$ , and  $E_c$  is the edge set in which each edge represents the relation between two vertices. For each attribute  $c$ , let  $L^c = \{l_1^c, l_2^c, \dots, l_o^c\}$  represent the label set of attribute  $c$ .  $g_{ij}^c$  is a weighted relation between label  $l_i^c$  and  $l_j^c$  vertices in the relation graph  $g^c$  of attribute  $c$ .

Let  $a$  denote the relation of actors and  $h$  denote the relation of hashtags, respectively. Then we can construct the relation graph  $g^a = (V_a, E_a)$  of actors and the relation graph  $g^h = (V_h, E_h)$  of hashtags. In the relation graph  $g^a$ , each edge indicates the relation between two actors who are likely to co-occur in one tweet. In the relation graph  $g^h$ , each edge indicates relations between two hashtags which are likely to co-occur in one tweet. The weights of the edges in both graphs are the number of co-occurrence instances of the adjacent vertices in the corpus of short texts. As shown in Fig. 1, usually there are multiple relation graphs.

### B. Multiple Relational Topic Modeling

$\theta_1^K$  is the distribution over topics with dirichlet prior parameter  $\alpha$ .  $\phi_k$  is the topic-word distribution with dirichlet prior parameter  $\beta$ .  $z$  represents the topic assignment matrix and  $z_d$  represents the topic assignment for a short text  $d$ .  $w_d$  is the word sequence of short text  $d$  and  $w_{di}$  represents the word at position  $i$  in  $d$ . Then the parameters of MRTM are as follows.

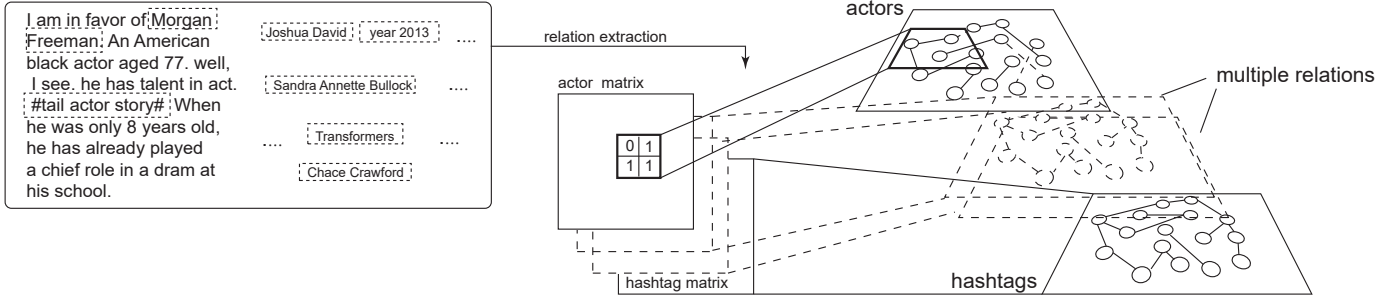


Fig. 1: Multi relations extracted from short texts

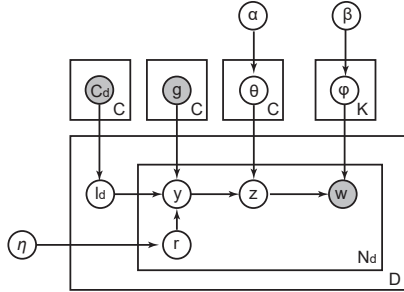


Fig. 2: The model structure of MRTM

$$\theta_c | \alpha \sim \text{Dirichlet}(\alpha) \quad (1)$$

$$\phi_k | \beta \sim \text{Dirichlet}(\beta) \quad (2)$$

$$z_{di} \sim \text{Multinomial}(\theta_{y_{di}}) \quad (3)$$

$$w_{di} \sim \text{Multinomial}(\phi_{z_{di}}) \quad (4)$$

Fig. 2 presents the structure of our proposed multiple relation topic model. A unique topic distribution is associated with each label in each attribute. Each topic is represented as a multinomial distribution over words. A short text could include two or more attributes, and attributes can serve as bridges between short texts. In order to incorporate multiple relations into our topic model, we first introduce the following concepts.

**Document-attribute distribution**  $C_d$ . Documents could prefer some attributes. For example, users talking about a movie are likely to discuss actors in the movie rather than hashtag concepts. Vice versa, users might be willing to talk about hashtag concepts than actors. We use document-attribute matrix to model this kind of preference of attributes, and each attribute has its unique distribution of topics.

**Observed attribute labels**  $l_d$ . Observed attribute labels refer to the labels existing in a short text. As prior knowledge, it is better than the simple pooling strategy [11].

**Potential attribute labels**  $y^p$ . Potential attribute labels refer to the labels inferred from the whole corpus based on the graph based attribute relation. Related labels of the observed labels are also to be utilized as prior knowledge.

Let  $l$  denote the attribute assignment and  $l_d$  be assignment vector of short text  $d$ .  $y_{di}$  represents the attribute assignment for position  $i$  in short text  $d$ . Different short texts have different attribute preference, represented by  $C_d$ , the document-attribute distribution. By introducing an indicator parameter  $r$  to decide

whether we choose the observed attribute label or potential attribute label(s), the label assignment at position  $i$  of short text  $d$ ,  $y_{di}$  is defined in Eq. 5.

$$y_{di} | C_d, g^c, r \sim \text{Bernoulli}(\eta) \quad (5)$$

where  $\eta$  controls the randomness of attribute labels. Large  $\eta$  means more randomness.

The overall generative process for MRTM is described below.

- 1) for each of the label  $l$  in attribute  $C$ ,  $l \in \{1, \dots, L\}$ , sample  $\theta_l \sim \text{Dir}(\alpha)$ ;
- 2) for each of the topic  $k \in \{1, \dots, K\}$ , draw words  $\phi_k \sim \text{Dir}(\beta)$ ;
- 3) for each of the documents  $d \in \{1, \dots, D\}$ , give all document-attribute distribution  $C_d$  and all prior knowledge  $g^c$ ;
- 4) for each word  $w_{di} \in d$ ,  $i \in \{1, \dots, N_d\}$
- 5) sample  $c_{di} \sim \text{Multinomial}(C_d)$ ;
- 6) sample observed attribute label  $l_{di} \sim \text{Uniform}(c_{di})$
- 7) sample potential attribute label related with observed label in graph  $g^c$ ,  $y^p \sim \text{Multinomial}(\text{norm}(g_{l_{di}}^{c_{di}}))$ ;
- 8) sample  $r \sim \text{Bernoulli}(\eta)$ ;
- 9) if  $r$  is 0, sample  $y_{di} = l_{di}$ ;
- 10) else if  $r$  is 1, sample  $y_{di} = y^p$ ;
- 11) draw topic  $z_{di} \sim \text{Multinomial}(\theta_{y_{di}})$ ;
- 12) draw word  $w_{di} \sim \text{Multinomial}(\phi_{z_{di}})$ .

Firstly we sample current label based on  $C_d$ . Secondly, we sample a value of  $r$  from Bernoulli distribution and decide whether this label is related with current word. If not, we sample from highly related labels from the relation graph. Through the two-step sampling from line 6 to 10, we can obtain the document-attribute assignment.  $\text{norm}(g_{l_{di}}^{c_{di}})$  in line 7 is a normalized  $L$ -dimension possibility vector, where  $L$  is the number of labels in attribute  $c$  and each element in the vector is calculated by using the following equation.

$$p(l_j | l_{di}^{c_{di}}) = \frac{g_{l_{di}, l_j}^{c_{di}}}{\sum_{l'=1}^L (g_{l_{di}, l'}^{c_{di}})} \quad (6)$$

This model adds word co-occurrence under graph structure when latent relationships between short texts are found, and we filter unnatural word co-occurrence caused by merely aggregating short texts. In our experiment after introducing actor networks or hash-tag network, we enhance the semantic information in movie reviews. More exciting is that we can

add more relations to this framework in general tasks based on short text topic modeling.

### C. Model Parameter Inference

The joint distribution of the latent variables is

$$p(\mathbf{w}|\theta, \phi, r, \mathbf{l}, G) = \prod_{d=1}^D p(\mathbf{w}_d|\theta, \phi, r, \mathbf{l}_d, \mathbf{g}^c). \quad (7)$$

Assuming that attribute-topic distribution  $\mathbf{l}_d$  and topic-word distribution  $\phi$  are independent, we have

$$\begin{aligned} p(\mathbf{w}_d|\theta, \phi, r, \mathbf{l}_d, \mathbf{g}^c) &= \prod_{i=1}^{N_d} p(w_{di}|\theta, \phi, r, \mathbf{l}_d, \mathbf{g}^c) \\ &= \prod_{i=1}^{N_d} \sum_{c=1}^C \sum_{k=1}^K p(w_{di}, z_{di} = k, y_{di} = c | \\ &\quad \theta, \phi, r, \mathbf{l}_d, \mathbf{g}^c) \\ &= \prod_{i=1}^{N_d} \sum_{c=1}^C \sum_{k=1}^K \phi_{w_{di}} \theta_{kc} p_{cy_{di}} \end{aligned} \quad (8)$$

where  $p_{cy_{di}} = p(y_{di} = c|r, \mathbf{l}_d, \mathbf{g}^c)$  represents the assignment possibility of attribute  $c$  on attribute-topic distribution  $\mathbf{l}_d$  and potential labels.

According to the two-step sampling mentioned in Section III-B, the assignment of  $c$  is associated with relation graph  $\mathbf{g}^c$ . Then we have

$$\begin{aligned} p(y_{di} = c|r, \mathbf{l}_d, \mathbf{g}^c) &= \left\{ p(l_{di}^c = c|l_d) p(y_{di} = c|l_{di}^c) \right\}^r \times \\ &\quad \left\{ \sum_{l'=1}^L p(l_{di}^c = l'|C_d) p(y_{di} = c|l_{di}^c = l', \mathbf{g}^c) \right\}^{1-r} \end{aligned} \quad (9)$$

It is computational infeasible to estimate directly the conditional probability distribution  $p(\mathbf{w}_d|\theta, \phi, r, \mathbf{l}_d, \mathbf{g}^c)$ , like many other topic modeling approaches, we adopt Gibbs sampling [12] to approximate it as follows.

$$\begin{aligned} p(z_{di} = k, y_{di} = c, r_{di} = u | \\ w_{di} = v, \mathbf{z}_{-di}, \mathbf{y}_{-di}, \mathbf{w}_{-di}, C, G, \alpha, \beta, \eta) \\ \propto \frac{N_{vk, -di}^{VK} + \beta}{\sum_{v'} N_{v'k, -di}^{V'K} + V\beta} \cdot \frac{N_{kc, -di}^{KC} + \alpha}{\sum_{k'} N_{k'c, -di}^{K'C} + K\alpha} \cdot p_{cy_{di}} \end{aligned} \quad (10)$$

Let  $N^{KC}$  be the matrix recording the number of times that a topic is assigned to some attribute. Let  $N^{VK}$  be the matrix recording the number of times that a real word is assigned to some topic. After iterative sampling, the final  $\theta_c$  and  $\phi_k$  are as follows.

$$\begin{aligned} \hat{\theta}_c &\propto \frac{N_{kc}^{KC} + \alpha}{\sum_{k'} N_{k'c}^{K'C} + K\alpha} \\ \hat{\phi}_k &\propto \frac{N_{vk}^{VK} + \beta}{\sum_{v'} N_{v'k}^{V'K} + V\beta} \end{aligned} \quad (11)$$

## IV. EXPERIMENTAL EVALUATION

In this section, we report our experimental results. The quality of uncovered topics of short texts is evaluated using both subjective and objective metrics. All experiments are done on a PC with Intel i5 CPU at 2.3 GHz and 8GB memory, running Windows 10. All algorithms are implemented in Python.

### A. Datasets and Settings

We collected more than 150,000 tweets from Microblog<sup>1</sup>, which is a Chinese social network site similar to twitter. All tweets are in Chinese, and related to Chinese movies released in 2017 in order to narrow down the domain of the potential topics for analysis. Unlike English, sentences in Chinese do not contain spaces between words. We applied JieBa<sup>2</sup> (an open-source NLP tool for Chinese) to segment sentences into words and remove stop words.

It is worth noting that the relations utilized in the proposed topic model is widely available in short texts in many applications. In particular, these attributes, labels, and relations could be manually defined, automatically learned, or extracted from existing knowledge bases. In the experiments, we extract relations by using both knowledge based matching and RegExp based matching.

### B. Subjective Quality Evaluation

We report the uncovered topics of each method and evaluate their quality in a subjective view in this section. We conducted the experiments on tweets about five most popular Chinese movies released in 2017. The characteristics of the movies and the tweets are presented in Table II. For readers not speaking Chinese, we translated all the Chinese words into English in Table II, as well as ones in the following table, where phrases in italic are movie titles, and phrases with underline are actor names.

We compared the proposed MRTM with the state-of-the-art topic model for short texts, i.e., HGTM [11], as well as the classic LDA topic model [6] as a baseline. Recall that HGTM is the topic model for short texts based on the hashtag graph. In all models, the topic number  $K = 150$ , the latent variable  $\alpha = 0.5$ , and  $\beta = 0.1$ . We set  $\eta = 0.5$  in HGTM as indicated in [11]. The number of iteration times for Gibbs sampling is set to be 1000 in all experiments.

Table I shows the top 10 words from top 1 topic uncovered by each models ranking based on probabilities. The irrelevant words found by LDA are marked with label irrelevant in parentheses. New words found by HGTM and MRTM are in bold. We also summarized the new words of HGTM and MRTM in Table III, by categorizing all new words into two groups, major actors and relevant words consistent with movie genres.

By careful analysis of the discovered topics together with the original tweets, we have the following observations. For Movie #1 and #2, the topic found by LDA contains many

<sup>1</sup><http://www.weibo.com>

<sup>2</sup><https://github.com/fxsjy/jieba>

TABLE I: Top 1 topic discovered by LDA, HGTM and MRTM

	Movie #1	Movie #2	Movie #3	Movie #4	Movie #5
LDA	<i>Taohua, Sansheng, film, Sanshi, Shili, happiness, summer-vacation, interesting, movie season, not bad</i>	film, sacrifice, suspect, <b>duo(irrelevant), meng(irrelevant),</b> superise, not bad, <b>both(irrelevant),</b> propaganda, acting skill	support, <i>Zhanlang,</i> movie season, happiness, interesting, film, good, summer vacation, dream, Chinese	memory, film, master, acting skill, plot, reversal, interesting, murderer, really, <u>Huang Bo</u>	new year movie, movie-season, not good, strongest, <u>Jackie Chan,</u> more and more, interesting, film, <i>Yoga,</i> not bad
HGTM	movie season, summer-vacation, happiness, <i>Sansheng, Sanshi, Shili, Taohua,</i> film, <b>Yang Yang, special effects</b>	film, suspect, sacrifice, like, support, <b>awesome,</b> fighting, superise, acting skill, propaganda	Chinese, <i>Zhanlang,</i> <b>Wu Jing,</b> film, box-office, <b>pride,</b> support, <b>strike, poke,</b> enjoy	memory, master, <u>Huang Bo,</u> <b>Duan Yi Hong,</b> film, expect, eyesight, plot, reversal, murderer	<b>Kung Fu, Yoga,</b> strongest, interesting, <u>Jackie Chan,</u> <b>Zhang Yi Xin,</b> brother, excellent, <b>laugh,</b> like
MRTM	summer vacation, <b>Yang Yang,</b> interesting, <b>special effects,</b> <i>Sansheng, Sanshi, Shili, Taohua,</i> film, movie season	suspect, film, sacrifice, awesome, like, superise, acting-skill, love, enjoy, <b>fear</b>	<i>Zhanlang,</i> <b>Wu Jing,</b> patriotic, support, strong, <b>strike,</b> Chinese, film, scene, enjoy	memory, master, <u>Huang Bo,</u> <b>Duan Yi Hong,</b> murderer, plot, reversal, film, <b>killer,</b> except	<b>Kung Fu, Jackie Chan,</b> interesting, <i>Yoga,</i> <b>laugh,</b> <b>zhang yi xin,</b> love, excellent, <b>indian, funny</b>

TABLE II: The characteristics of movies and tweets

ID	Movie Title	Movie Genre	# of Tweets
#1	Sansheng Sanshi Shili Taohua	romance, fantasy	9,932
#2	The Devotion Of Suspect X	feature, crime	9,215
#3	Wolf Warriors II	action, military	19,230
#4	Battle of Memories	suspense, crime	9,355
#5	Kung-Fu Yoga	comedy, adventure	7,375

TABLE III: Words discovered by HGTM and MRTM

ID	Words of leading actor	Words related to movie genres
#1	<u>Yang Yang</u>	special effects
#2	-	awesome, fear
#3	<u>Wu Jing</u>	pride, strike, poke
#4	<u>Huang Bo, Duan Yi Hong</u>	killer
#5	<u>Zhang Yi Xin</u>	<i>Kung Fu,</i> laugh, Indian, funny

irrelevant words. For Movie #3, #4, and #5, the topic contains major actors names such as Huang Bo (A Chinese actor) and Jackie Chan. In general, it seems that words in LDA's topics provide an overview but also bring some unnatural words such as 'not bad' or 'film'(background noise, [9]) which caused by unnatural co-occurrence.

For HGTM and MRTM, in Movie #1, they both found out the leading actor 'Yang Yang' with different rankings. More substantive words have higher probabilities than words in the movie title, such as *Sansheng, Sanshi, Shili, Taohua*. In Movie #2 and #3, both HGTM and MRTM can get rid of irreverent words and find new words. In Movie #4, movie genre related word like 'killer' is within the top 10-word list, substituting the irrelevant word 'really'. In Movie #5, word 'laugh' shows this movie is a comedy. The word 'Indian' appearing in MRTM's

topics is because the story of the movie took place in India.

### C. Objective Semantic Coherence

In this section, we report the uncovered topics of each method and evaluate their quality in an objective view. We employed Pointwise Mutual Information (PMI) [17] to measure the topic coherence, which has been proved to be an effective measure for topic quality [18]. Given a topic  $t$  and its top  $K$  words  $W^t = (w_1^t, w_2^t, \dots, w_K^t)$  (The top  $K$  words with highest probabilities.),  $P(w)$  denotes the document frequency of word  $w$  and  $P(w_l, w'_l)$  is the probability  $w_l$  and  $w'_l$  co-occur. The metric is defined for a specific topic  $t$  as

$$PMI(w_l^t, w_{l'}^t) = \log\left(\frac{P(w_l^t, w_{l'}^t)}{P(w_l^t)P(w_{l'}^t)}\right). \quad (12)$$

Then the coherence of a top- $K$  models is the summarization of the PMI scores of each topic as follows. Larger coherence score shows better topic partition.

$$Coherence(t, W^t) = \sum_{i=2}^K \sum_{j=1}^i \log\left(\frac{P(w_i^t, w_j^t) + \epsilon}{P(w_i^t)P(w_j^t)}\right) \quad (13)$$

We filtered the tweet corpus by removing tweets of unpopular movies and split the remaining tweets into two data sets, MR1 and MR2. MR1 has 2,3452 tweets with the average length 6.96, while MR2 has 7,329 tweets with the average length 7.97. We compared MRTM with the classic topic model Latent Dirichlet Allocation (**LDA**) [6], the author topic model (**ATM**) [8], and the hashtag graph based topic model (**HGTM**) [11].

We extract top 10 words for each topic generated by each model for computing the coherence. The number of topics is 100,  $\alpha = 0.3$  and  $\beta = 0.1$ . The number of iterations of Gibbs sampling is 200 which is enough for uncovering topics. We conduct all the experiments repeatedly for 5 times and report the mean value of each measure in Table IV. For each topic,

TABLE IV: Average PMI and coherence scores of in MR1 and MR2

Model	dataset	MR1					MR2				
	$K$	5	10	15	20	30	5	10	15	20	30
LDA [6]	Average PMI	-6.3264	-7.9251	-8.3323	-8.6869	-9.1779	-3.8294	-5.6971	-6.9512	-8.0397	-8.9713
	Coherence	-31.632	-79.251	-124.9845	-173.738	-275.337	-19.147	-56.971	-104.268	-160.794	-269.139
ATM [8]	Average PMI	-7.8479	-8.9783	-9.6743	-9.5824	-9.546	-6.3908	-7.5662	-8.0043	-8.1444	-8.6616
	Coherence	-39.2395	-89.783	-145.1145	-191.648	-286.38	-31.954	-75.662	-120.0645	-162.888	-259.848
HGTM [11]	Average PMI	<b>-1.9698</b>	-3.9027	-5.0757	-5.4254	-6.243	-3.0478	-3.6876	-4.2665	-4.8512	-5.1454
	Coherence	<b>-9.849</b>	-39.027	-76.1355	-108.508	-187.29	-15.239	-36.876	-63.9975	-97.024	-154.362
MRTM	Average PMI	-2.3854	<b>-2.7416</b>	<b>-3.0652</b>	<b>-3.3337</b>	<b>-3.6853</b>	<b>-2.7831</b>	<b>-3.2135</b>	<b>-3.9521</b>	<b>-4.3683</b>	<b>-4.8573</b>
	Coherence	-11.927	<b>-27.416</b>	<b>-45.978</b>	<b>-66.674</b>	<b>-110.559</b>	<b>-13.9155</b>	<b>-32.135</b>	<b>-59.2815</b>	<b>-87.366</b>	<b>-145.719</b>

we only keep  $K$  words with the largest probabilities, and the average PMI indicates the average PMI scores of all topics.

MRTM achieves lowest scores in all settings except when topic length is 5 in MR1. Classic LDA and ATM show lower coherence because they cannot handle the sparsity of short texts. For all approaches, the coherence becomes unstable when  $K$  goes larger generally. However, an interesting fact we found is that using hashtags is more stable than using other relations, and with multiple relations, we can further lower the trend of increasing. Another observation is related to the range of average PMI and coherence along with the change of  $K$ . The range in MRTM is much smaller than the one in HGTM, i.e., 60% approximately. With smaller range of average PMI and coherence, MRTM is more robust than HGTM with respect to  $K$ .

It is essential to incorporate multiple relations appropriately. Otherwise, it might deteriorate the quality of discovered topics. In the proposed MRTM, we introduce new latent document attribute layer and incorporate multiple relations to obtain high coherent topics.

## V. CONCLUSION

A Multiple Relation Topic Model (MRTM) is proposed in this paper with the aim of overcoming the difficulties of sparsity and informality caused by noisy short texts. By incorporating explicit and implicit relations among short texts into the generative process of short texts, MRTM can uncover high-quality topics. Extensive experiments demonstrate that MRTM can achieve better performance than both the classic topic model approach LDA, and the state-of-the-art topic modeling approaches, i.e., ATM and HGTM. Possible future directions include accelerating the sampling speed and trading off between explicit and implicit relations of short texts.

## ACKNOWLEDGEMENTS

This work is supported in part by Jiangsu Provincial Natural Science Foundation of China under Grant BK20171447, Jiangsu Provincial University Natural Science Research of China under Grant 17KJB520024, Jiangsu Key Laboratory of Big Data Security & Intelligent Processing under Grant BDSIP1803 and Nanjing University of Posts and Telecommunications under Grant No. NY215045.

## REFERENCES

- [1] R. Krestel, P. Fankhauser, and W. Nejdl, "Latent dirichlet allocation for tag recommendation," in *Proceedings of the third ACM conference on Recommender systems*. ACM, 2009, pp. 61–68.
- [2] A. McCallum, X. Wang, and A. Corrada-Emmanuel, "Topic and role discovery in social networks with experiments on enron and academic email," *Journal of Artificial Intelligence Research*, vol. 30, pp. 249–272, 2007.
- [3] J. H. Lau, N. Collier, and T. Baldwin, "Online trend analysis with topic models: twitter trends detection topic model online," in *International Conference on Computational Linguistics*, 2012, pp. 1519–1534.
- [4] R. Mehrotra, S. Sanner, W. Buntine, and L. Xie, "Improving lda topic models for microblogs via tweet pooling and automatic labeling," in *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2013, pp. 889–892.
- [5] X. Yan, J. Guo, Y. Lan, and X. Cheng, "A bitern topic model for short texts," in *Proceedings of the 22nd international conference on World Wide Web*. ACM, 2013, pp. 1445–1456.
- [6] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, 2003.
- [7] Y. Zuo, J. Wu, H. Zhang, H. Lin, F. Wang, K. Xu, and H. Xiong, "Topic modeling of short texts: A pseudo-document view," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 2105–2114.
- [8] M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth, "The author-topic model for authors and documents," in *Proceedings of the 20th conference on Uncertainty in artificial intelligence*. AUAI Press, 2004, pp. 487–494.
- [9] W. X. Zhao, J. Jiang, J. Weng, J. He, E.-P. Lim, H. Yan, and X. Li, "Comparing twitter and traditional media using topic models," in *European Conference on Information Retrieval*. Springer, 2011, pp. 338–349.
- [10] M. Naaman, H. Becker, and L. Gravano, "Hip and trendy: Characterizing emerging trends on twitter," *Journal of the Association for Information Science and Technology*, vol. 62, no. 5, pp. 902–918, 2011.
- [11] Y. Wang, J. Liu, J. Qu, Y. Huang, J. Chen, and X. Feng, "Hashtag graph based topic model for tweet mining," pp. 1025–1030, 2014.
- [12] T. Griffiths, "Gibbs sampling in the generative model of latent dirichlet allocation," 2002.
- [13] C. Li, H. Wang, Z. Zhang, A. Sun, and Z. Ma, "Topic modeling for short texts with auxiliary word embeddings," in *International Acm Sigir Conference on Research & Development in Information Retrieval*, 2016, pp. 165–174.
- [14] D. Q. Nguyen, R. Billingsley, L. Du, and M. Johnson, "Improving topic models with latent feature word representations," *Transactions of the Association for Computational Linguistics*, vol. 3, pp. 299–313, 2015.
- [15] D. Ramage, D. Hall, R. Nallapati, and C. D. Manning, "Labeled lda: a supervised topic model for credit attribution in multi-labeled corpora," in *Conference on Empirical Methods in Natural Language Processing: Volume*, 2009, pp. 248–256.
- [16] S. Li, J. Li, and R. Pan, "Tag-weighted topic model for mining semi-structured documents," in *International Joint Conference on Artificial Intelligence*, 2013, pp. 2855–2861.
- [17] D. Newman, J. H. Lau, K. Grieser, and T. Baldwin, "Automatic evaluation of topic coherence," in *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 2-4, 2010, Los Angeles, California, USA*, 2010, pp. 100–108.
- [18] A. Fang, C. Macdonald, I. Ounis, and P. Habel, *Topics in tweets: A user study of topic coherence metrics for Twitter data*, 2016.

# Svega: Answering Natural Language Questions over Knowledge Base with Semantic Matching

Gaofeng Li, Pingpeng Yuan, and Hai Jin

Services Computing Technology and System Lab. / Cluster and Grid Computing Lab. / Big Data Technology and System Lab.  
School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China  
Email: {gaofengli, ppyuan, hjin}@hust.edu.cn

**Abstract**—Nowadays, more and more large scale knowledge bases are available for public access. Although these knowledge bases have their inherent access interfaces, such as SPARQL, they are generally unfriendly to end users. An intuitive way to bridge the gap between users and knowledge bases is to enable users to ask questions with natural language interface and return desired answers directly. Here the challenge is how to discover the query intention of users. Another challenge is how to obtain accurate answers from knowledge bases. In this paper, we model the query intention with a graph based on an entity-driven method. Consequently, the core problem of natural language question answering can be treated as subgraph matching over knowledge bases. For a query graph, there is a huge number of candidate mappings in a knowledge base, including ambiguities. Thus, a semantic vector is proposed to address disambiguation by evaluating the semantic similarity between edges in a query graph and paths in a knowledge base. By this way, our system can extract accurate answers directly without any offline work. Extensive experiments over the series of QALD challenges show the effectiveness of our system Svega in terms of recall and precision against other state-of-the-art systems.

## I. INTRODUCTION

Nowadays, many knowledge bases, such as DBpedia [1], Yago [2], are available for public access. Distinct from document bases of information retrieval systems, knowledge bases integrate substantial small facts, known as triples (subject-predicate-object). To access these knowledge bases, SPARQL, a SQL-like query language, is provided as a standard interface. However, SPARQL is unfriendly to general users because of its complex syntax. Although keyword search, commonly applied in information retrieval, is simple and convenient for end users, it may not work properly when employing keyword search to retrieve answers from knowledge bases, because it is difficult for keywords to express the query intention completely. For example, the keywords of the question “*Who starred in the films that were directed by Stanley Kubrick?*” may be “*star*”, “*film*”, and “*Stanley Kubrick*”. However, if only taking into account these keywords, the real query intention, *actors of the films*, will be missing. Compared with SPARQL and keyword, natural language is not only intuitive to end users, but also able to express users’ query intention accurately. Thus it is important to bridge the gap between unstructured natural language and structured knowledge bases.

One way to bridge the gap is to translate a natural language question to a structured query and then extract answers from the mapping results of the query. However, it is difficult to convert a natural language question to a structured query. One reason is that most structured queries need to specify the query statement accurately, while some query statements have ambiguities between natural language and knowledge bases. Considering the above example, the phrase “*Stanley Kubrick*” may refer to <Stanley\_Kburick> or <Stanley\_Kubrick\_Archive> in knowledge bases. The exact meaning of phrases depends on the context of questions and knowledge bases. In order to clarify the meaning of phrases and obtain correct mappings from knowledge bases, some QA (*Question Answering*) systems provide candidates for users to interactively choose [3], which actually is a controlled natural language. The above example question can be converted into “*Who dbp:starring the dbo:Film that were-dbp:director res:Stanley\_Kubrick?*” by replacing the phrases with the words of a knowledge base. This approach facilitates QA systems, but it requires users to do much. Other QA systems, such as gAnswer [4], automatically map phrases in the natural language into words in knowledge bases based on a dictionary, which is generally built manually or using machine learning approach. However, it is impossible for the offline-built dictionary to indicate comprehensive mapping relationships between phrases in the natural language and knowledge bases.

Here we present an online natural language question answering system Svega. An entity-driven method is proposed to translate a natural language question into a query graph. Consequently, the problem of natural language question answering is converted to subgraph matching over knowledge bases. To address disambiguation, we first search isomorphism subgraphs of the query graph from a knowledge base based on vertex mapping. Then a semantic vector is proposed to filter ambiguities by evaluating semantic similarity. Our contributions are as follows.

- 1) An online question answering system Svega is presented. Svega provides a natural language interface for general users to retrieve desired answers from knowledge bases directly, without any offline work.
- 2) Query graph is constructed to model the query intention of a natural language question by adopting an entity-

driven approach. In this approach, entities are identified firstly and then the query intention is explored by extracting predicates based on grammatical relationships.

- 3) A semantic vector is proposed to represent the semantics of paths (namely edges in query graph and paths in knowledge base). Thus, the problem of disambiguation is addressed by evaluating the semantic similarity between a query graph and mapping subgraphs online.
- 4) Extensive experiments on the standards of QALD series competitions are conducted and the experimental results demonstrate that Svega has a competitive performance in terms of recall and precision.

## II. RELATED WORK

There are many QA systems available. The traditional QA systems can only return texts related to keywords. Although some extending QA systems, such as [5], consider semantics of results, they still can not retrieve answers directly. The development of knowledge bases, storing fine-grained facts, makes it possible to directly return desired answers [6], [7], [8], [9]. Since the inherent interfaces, such as SPARQL, provided by knowledge bases are too complex for general users, current QA systems allow users to employ natural language to access knowledge bases. According to the restriction on natural language, these systems can be broadly classified into two categories: controlled natural languages, which restrict the grammar and vocabulary in order to reduce ambiguity and complexity, and un-controlled natural language.

**Controlled natural language.** These systems using this approach provide some candidate entities and predicates for users' choice [3], [7]. Then users choose words from candidates to indicate their query intention. For instance, the example can be transferred to “*Who dbp:starring the dbo:Film that were-dbp:director res:Stanley\_Kubrick?*”. In the question, “*dbp:starring*”, “*dbo:Film*”, “*dbp:director*”, and “*res:Stanley\_Kubrick*” are not natural language words, but entities and predicates from a knowledge graph. Thus, this way limits the expressiveness and usability of QA systems [3], but improves the correctness to answer questions.

**Un-controlled natural language.** Different from the approaches with controlled natural languages, users can answer questions using any words in these systems. Then the QA systems usually translate natural language questions into structured queries, such as SPARQL. In this approach, the key is how to identify entities and map entities into words of knowledge bases. For example, Xser [8] maps phrases into words of knowledge bases using an ad-hoc lexicon. CASIA [9] detects phrases by grammatical token and then uses Markov Logic Network to resolve ambiguities. gAnswer [4] builds a paraphrase dictionary offline to achieve the mapping of predicate. However, the dictionary built offline can not indicate the exact meaning of each predicate because language is live. All in all, these systems address disambiguation only by taking into account the semantic of single words instead of context of the words.

## III. PRELIMINARY AND OVERVIEW OF SVEGA

### A. Preliminary

The core issues of answering natural language questions over knowledge bases is to bridge the gap between structured knowledge bases and unstructured natural language question. Regarding this problem, we first model the query intention of a natural language question with query graph  $Q$ .

**Definition 1: (Query Graph).** A query graph is denoted as  $Q = (V, E)$ , where  $V$  is a set of vertices, corresponding to entities or variables. Specially, the query intention or variable is represented as “?”. The edge between vertex  $v_i, v_j$  can be denoted as *triple*  $T = \langle v_i, r, v_j \rangle$ , where  $r$  represents the relation between  $v_i, v_j$ .

Except variable vertices, there are two kinds of vertices in query graph. We name a vertex with an entity label as a *Key Vertex* (KV). The other vertices are not assigned a label because they are not indicated in questions, which are named as *Hidden Vertex* (HV) since they are implicit. For example, Fig.1 shows the query graph of the running question.  $\langle \text{Stanley\_Kubrick, directed, HV} \rangle$  is a triple, in which  $\langle \text{Stanley\_Kubrick} \rangle$  is a KV. The graph also has a HV, which is the set of films that were directed by *Stanley Kubrick*.

Now, answering natural language question is actually to find matches of a query graph over knowledge bases. When mapping a query graph to a knowledge base, ambiguities will be introduced. For a candidate mapping of an entity in  $Q$ , if there is no isomorphism subgraph containing it, it must be an ambiguity. For example, the vertex  $\langle \text{Stanley\_Kubrick\_Archive} \rangle$  in Fig.1 is a ambiguity. Based on this idea, we address disambiguation with subgraph matching, which considers the schema of knowledge bases.

**Definition 2: (Match).** A subgraph  $S$  in a knowledge base is a match of query graph  $Q$  if and only if the following conditions hold:

- 1)  $\forall v \in V$  of  $Q$ ,  $\exists u \in S$  where the vertex  $u$  is a mapping of the vertex  $v$ ;
- 2)  $\forall e : (v_i, v_j) \in E$  of  $Q$ ,  $\exists (u_i, u_j) \in S$ , associated with  $e$ , and where the two vertices  $u_i$  and  $u_j$  are mappings of  $v_i$  and  $v_j$  respectively.

The mappings of vertex “?” in  $Q$  will be answers of the query. In Fig.1, vertex  $\langle \text{Walter\_Cartier} \rangle$  and  $\langle \text{Stadmueller} \rangle$  are the matches of vertex “?”. Thus, they are two answers of the running question.

### B. Overview of Svega

Our approach mainly consists two parts: *Entity-driven Query Graph Construction* and *Disambiguation with Query Graph Mapping*. Fig.1 shows the framework of our system.

**Entity-driven Query Graph Construction.** In order to model the query intention of a natural language question with  $Q$ , we extract *triples* (see Definition 1) from natural language question firstly, because *triple* has a simpler structure. Based on this idea, we propose an entity-driven approach to extract *triples* and construct  $Q$  by joining the same vertex of *triples*. The details will be described in Sect. IV.



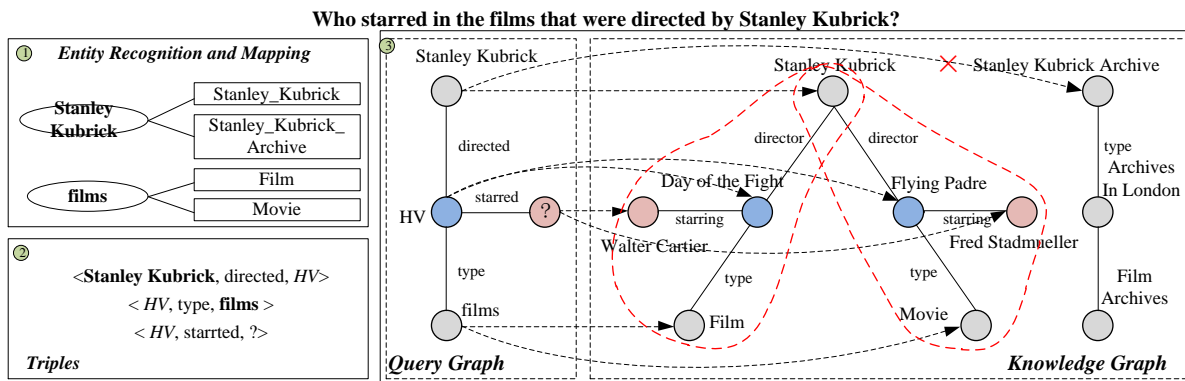


Fig. 1: System Overview

**Disambiguation with Query Graph Mapping.** Naturally, a query graph matching mainly includes vertices and edges mapping, which will introduce ambiguities. To address disambiguation, we first search isomorphism subgraphs of  $Q$  based on vertices mapping. Then a semantic vector is proposed to prune ambiguities by evaluating the semantic similarity. The details will be described in Sect. V.

#### IV. ENTITY-DRIVEN QUERY GRAPH CONSTRUCTION

In a query graph, any edge with two endpoints can be represented as a triple. Thus, we first extract *triples* from a natural language question. Each *triple* is composed by two entities (vertices in a graph) and the relation between them (edge between two vertices). We then construct the query graph of the questions from the *triples*. To extract triples, the system needs to identify entities and relationship between entities indicated in natural language questions. The words to show relationship are various in natural language while the form of entity is relatively simple. They can be verb and adjective phrases etc. Here, we first identify entities and then propose an entity-driven approach to extract relational phrases based on the grammatical relationships between words.

##### A. Entity Identification

Here, we employ DBpedia Spotlight [10] and Stanford Parser [11] to analyze a sentence and generate a *dependency tree* to indicate grammatical relationships between words. The output is generally a dependency tree. For example, Fig.2 is the tree of the example question. There are many auxiliary words (“in”, “that” etc) in a dependency tree while a query graph only contains key entities and relationships between them. So a dependency tree is still far from a query graph. In order to construct  $Q$ , we need to know both the key entities and their semantics. For instance, the category words (e.g. “film” in the running example) generally indicate ‘is-a’ relationship. In the case, we will add some extra nodes and edges to show this. As shown in Fig.2, we will add a node and an edge labeled “type” which represents the fact that “film” is a “type”.

Some questions may contain hidden information. The example question “films that were directed by Stanley Kubrick” has

a hidden entity “that” which represents all the films directed by *Stanley Kubrick*. Here, we denote it by a *HV*. *HVs* can be identified based on the structural feature of vertices in a query graph. If a vertex is adjacent to only one vertex, it is a *KV*, otherwise it is a *HV*. So identifying *HVs* needs more information. In the running example, “film” and “Stanley Kubrick” can be recognized in this stage, and *HVs* will be inferred in the following stage.

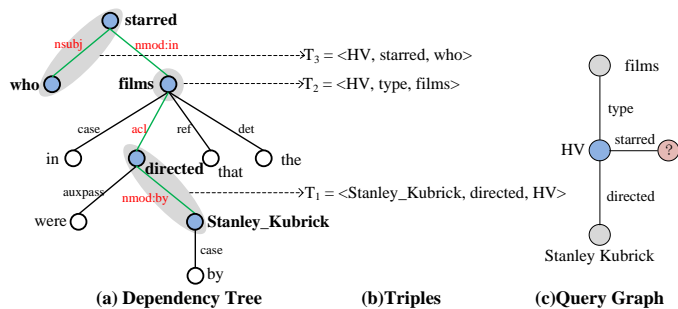


Fig. 2: Examples of Dependency Tree, Triple, and Query Graph

##### B. Recognizing Relation

Relation phrases generally co-occur with the corresponding entities. So they can be recognized from the words that have grammatical relationships with the obtained entities. Although a *dependency tree* [12] can reveal grammatical relationships between words in a sentence, the words connecting entities is not only the words that represent relations, but also auxiliary words. Here a relation priority is proposed to denote the possibility of a word to represent relation. The priority is computed based on the possibility of the grammatical relationships which are listed as follow in a non-ascending order<sup>1</sup>: *nsubj*, *subj*, *obj*, *dobj*, *nsubjpass*, *pobj*, *nmod:\**, *amod*, *prep*, *acl*, *auxpass*, *case*, *ref*, *det*.

In Fig.2, the word “were”, connecting to the word “directed”, will be pruned, because the edge label “auxpass”,

<sup>1</sup>These grammatical relationships are defined in [12]. For example, “nsubj” refers to a noun phrase which is the syntactic of a clause.

---

**Algorithm 1:** Extracting  $T$ s from a dependency tree

---

**Input:** Dependency tree, recognized entities set  $RES$ **Output:**  $T$ sDefine a variable  $var$ ;**for** each unvisited element in  $RES$  **do**    The element,  $e$ , is the member of a  $T$  and set  $var = e$ ;    Carry out traversal in the dependency tree from  $var$ ;

Select the edge with a highest-priority relationship;

**if** the  $V$  is a class word **then**        Insert a  $HV$  to current  $T$ ;        Build a new  $T$ , composed by  $V$ ,  $type$  and a  $HV$ ;

Continue;

**else**        Insert  $V$  and  $HV$  to the current  $T$ ;        Build a new  $T$  and insert a  $HV$  to the new  $T$ ;        Set  $var = V$ ;

Continue;

**end****end**

---

representing passive auxiliary, has a low priority. Simultaneously,  $HVs$  can be inferred according to the proposed structural feature of vertices.

Here, Algorithm 1 is used to extract *triples*. It starts from the entities recognized but not category words. For instance, the entity “Stanley\_Kubrick” will be an element of  $T_1$ . Then, because the priority of “*nmod:by*” is higher than “*case*”, “*directed*” is selected as a relation. Then, the class word “*films*” will be traveled and it will introduce the relation “*type*”. So here a  $HV$  is needed because it will connect two edges with label “*directed*” and “*type*” respectively. So  $\langle Stanley\_Kubrick, directed, HV \rangle$  and  $\langle HV, type, film \rangle$  are two triples extracted from the question. Next, “*starred*” and “*Who*” are travelled, and they are composed the  $T_3$ . Finally, the obtained three  $T$ s are composed together to build the  $Q$  by joining the same vertex.

## V. DISAMBIGUATION WITH QUERY GRAPH MAPPING

Once the query graph is built, we need to find subgraphs from a knowledge base, which match the semantics of the query graph. It is a NP hard problem to find the best mapping [4], and ambiguities will be introduced. Here we first filter ambiguities based on structural mapping, then a semantic vector is proposed to achieve semantic mapping.

### A. Structural Mapping Based on Vertices

Query graph mapping includes vertices and edges mapping. Due to this reason, we first extract isomorphism subgraphs based on vertices mapping. The general way to map vertices, with entity label, from query graph to knowledge bases is by computing string similarity or edit distance. By this way, an entity in the natural language may correspond to several candidates in a knowledge base. For instance, the possible mappings of the entity “Stanley Kubrick” in the running question include  $\langle Stanley\_Kubrick \rangle$ ,  $\langle Stanley\_Kubrick\_Archive \rangle$

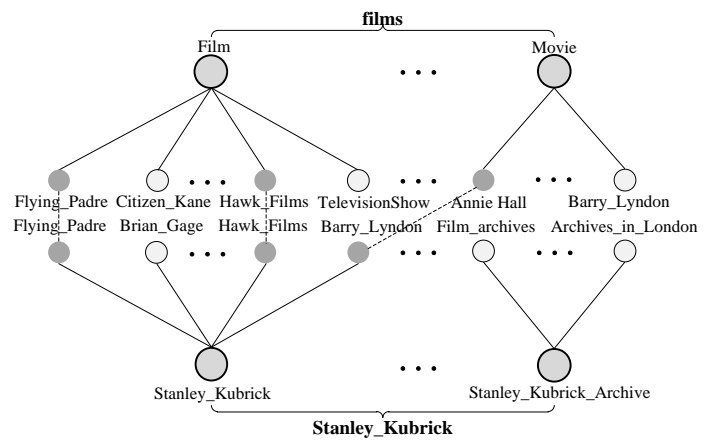


Fig. 3: Query Graph Mapping Based on Entities

---

**Algorithm 2:** Pruning paths based on entities

---

**Input:** Knowledge graph  $KG$ , query graph  $Q$ **Output:** mapping results of  $Q$ **for** each vertex  $V_i$  in the  $Q$  **do**    Obtain the mapping candidate set  $C_i$  of  $V_i$ ;    **for** each element  $c_j$  in  $C_i$  **do**        Carry out BFS in  $KG$  from the vertex  $c_j$ ;    **end**    **end****for** each edge between two vertex  $V_i$  and  $V_j$  in the  $Q$  **do**    Select two elements from the two candidate sets  $C_i$  and  $C_j$  respectively and find paths between them;**end**

---

etc. Moreover, in order to ensure that all answers can be obtained, synonyms should also be considered, which is an entity linking problem. In our work, Lookup [13] is adopted to link an entity to candidate entities in a knowledge graph. Up to now, all vertices in a query graph will have some candidate results and the corresponding paths only occur between these candidate results, so we can prune the paths that have no connection with the entities mapping results. Based on this idea, we propose the Algorithm 2 to prune paths in the knowledge graph. Fig.3 shows temporary results of the running example, and some ambiguities, such as  $\langle Stanley\_Kubrick\_Archive \rangle$ , will be filtered because it is isolation in the mapping results.

### B. Semantic Mapping with Path Vector

After obtaining isomorphism subgraphs of a query graph, there are still ambiguities in candidates, because only the structure of a query graph is considered, while the semantics of a query graph is not taken into account. For example, the vertex  $\langle Hawk\_Films \rangle$ , included in the mapping results, is not a correct answer of the running question, and the predicate, between  $\langle Hawk\_Films \rangle$  and  $\langle Stanley\_Kubrick \rangle$  in the knowledge graph, is irrelevant to the corresponding relation “*directed*” in  $Q$ . Consequently, the pruning method should be executed according to the semantic confidence.

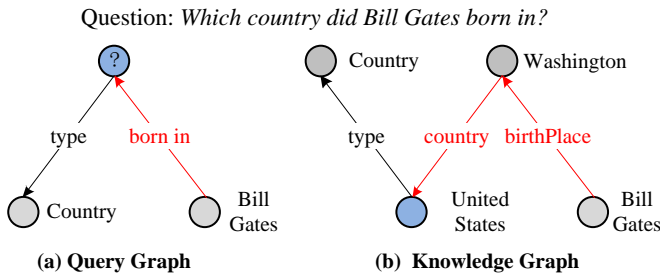


Fig. 4: Example of Edge and Mapped Path

An edge with a predicate label in a query graph may be mapped to a path in a knowledge graph, as shown in Fig.4. So, the key is how to represent the semantics of them and evaluate the similarity between them. Here, we propose semantic vector to quantify the semantic information of predicates and paths.

1) *Path Vector*: Generally, the semantic vector obtained by Glove [14] can only represent a single word. In knowledge bases, a path contains several edges, each of which has its labels. For example, in the path  $\langle birthPlace, country \rangle$ , “*birthPlace*” and “*country*” will have a semantic vector representation respectively. In addition, semantics of edges in a knowledge graph between different vertices are irrelevant. So we propose the method that composing the semantic vectors together based on the synthesis of vector additions to compute the path vector  $\alpha$ .

2) *Predicate Vector*: Generally, a predicate is a phrase. We also need a method to synthesize the predicate vector. Different from paths in knowledge graph, the words in a phrase are relevant. For example, the words “*born*” and “*in*” are composed together to represent the relationship that someone is born in a place. The contribution of each word in the predicate is different. The word “*in*” occurs in many phrases, while the word “*born*” only appears in a few phrases. Thus, the word “*born*” is more important than “*in*”. However, we can not ignore “*in*” because it indicates a born place instead of time or anything else. Thereby, we use the *tf-idf* proposed firstly in information retrieval to measure the importance of a word to a predicate phrase.

Assume a predicate phrase  $p$  is composed of word  $w_i$  ( $i = 1, \dots, n$ ). The *tf*-value of  $w_i$  is defined as follows:

$$tf(w_i, p) = |\{w_i \mid w_i \in p\}| \quad (1)$$

The *idf*-value of word  $w_i$  over the phrase dictionary  $d$  is defined as follows:

$$idf(w_i, d) = \log \frac{|d|}{|\{p \in d \mid w_i \in p\}| + 1} \quad (2)$$

Thus, the *tf-idf* value of  $w_i$  can be computed as following:

$$tf - idf(w_i, p, d) = tf(w_i, p) * idf(w_i, d) \quad (3)$$

According to the *tf-idf* value of each word, we define the importance weight of word  $w_i$  in predicate  $p$  as following:

$$\psi(w_i, p) = tf - idf(w_i, p, d) \quad (4)$$

Since each  $w_i$  has a vector  $v$ , we can compute the predicate vector  $\beta$  as follows:

$$\beta = \sum_{w_i \in p} \psi(w_i, p) * v_i \quad (5)$$

### C. Ranking Subgraph Matches

The subgraph matching mainly includes vertices and edges mapping. Consequently, semantic confidence of vertices and edges are composed together to measure the semantic similarity of a subgraph, which also reflect the confidence of the answer.

**Definition 3: (Answer Confidence)**. Given a query graph  $Q$  of a question and a mapping result  $M$ . Let  $\phi(v, u)$  be the confidence between vertex  $v \in Q$  and  $u \in M$ . And  $\varphi(\overline{vw}, P(x, y))$  is the confidence between edge  $\overline{vw}$  of  $Q$  and path  $P(x, y)$  of  $M$ . Thus, the answer confidence,  $AC(Q, M)$ , is defined as follows:

$$AC(Q, M) = \sum_{v_i \in Q \ \&\& \ u_i \in M} \phi(v_i, u_i) + \sum_{\overline{v_i v_j} \in Q \ \&\& \ P(u_i, u_j) \in M} \varphi(\overline{v_i v_j}, P(u_i, u_j)) \quad (6)$$

where

$$\varphi(\overline{v_i v_j}, P(u_i, u_j)) = \alpha(P(u_i, u_j)) \cdot \beta(\overline{v_i v_j}) \quad (7)$$

## VI. EVALUATION

In this section, we evaluate our system Svega against some existing popular natural language question answering systems using the QALD series benchmarks. Here, we do not choose squall2sparql [7] as a competitor because the input of squall2sparql is controlled language question rather than uncontrolled natural language.

### A. Data Sets

QALD series competitions are one of important benchmarks to evaluate natural language question answering system. Here we choose QALD-3 and QALD-4 as many research did. According to the requirements of the QALD series competitions, DBpedia series knowledge bases are also used in the experiments and managed by TripleBit [15].

### B. Effectiveness of Question Answering

We report the experimental results in Table I (QALD-3) and Table II (QALD-4). The experimental results of our competitors on QALD-3 and QALD-4 are available at the official website<sup>2</sup> of QALD, in which *Proceed* indicates the number of questions that can be return non-empty answers by these systems.

Table I shows that Svega is the best natural language question answering system on QALD-3, with highest *recall*, *precision*, and *F-measure*. Our system can answer 44 questions correctly and 9 questions partially, while CASIA [9] can only answer 29 questions all right.

In QALD-4, we can see that Svega outperforms all competitors on both *recall* and *precision* (Table II). Both the *recall* and

<sup>2</sup><https://qald.sebastianwalter.org/>

TABLE I: QALD-3 on DBpedia 3.8

	Proceed	Right	Partially	Recall	Precision	F-measure
CASIA	52	29	8	0.36	0.35	0.36
Scalewelis	70	32	1	0.33	0.33	0.33
RTV	55	30	4	0.34	0.32	0.33
Intui2	99	28	4	0.32	0.32	0.32
SWIP	21	15	2	0.16	0.17	0.17
<b>Svega</b>	96	44	9	<b>0.52</b>	<b>0.52</b>	<b>0.52</b>

TABLE II: QALD-4 on DBpedia 3.9

	Proceed	Right	Partially	Recall	Precision	F-measure
Xser	40	34	6	0.71	0.72	0.72
gAnswer	25	16	4	0.37	0.37	0.37
CASIA	26	15	4	0.40	0.32	0.36
Intui3	33	10	4	0.25	0.23	0.24
ISOFT	28	10	3	0.26	0.21	0.23
RO_FII	50	6	0	0.12	0.12	0.12
<b>Svega</b>	48	35	6	<b>0.76</b>	<b>0.76</b>	<b>0.76</b>

*precision* of Svega are 0.76, while the *recall* and *precision* of the best competitors Xser [8] is 0.71 and 0.72 respectively. In addition, Xser needs to train a KB-independent model offline before it answers questions, while Svega does not need to train any model in advance.

### C. Effectiveness of Query Graph Building and Mapping

We implement a system by replacing the semantic vector method of Svega with paraphrase dictionary method used in gAnswer. We name it as *ED+PD*.

The experimental results show that ED+PD is not better than Svega in all aspects (Table III, IV). It confirms that the similarity evaluating method of semantic vector is very effective, because the difference between ED+PD and Svega is only the similarity evaluating method. The results also show ED+PD outperforms gAnswer. It indicates that the entity-driven method of our system has more advantages on building query graph, because ED+PD and gAnswer use same dictionary, but different approach to build query graph.

TABLE III: Results on QALD-3

	Proceed	Right	Partially	Recall	Precision	F-measure
Svega	96	44	9	0.52	0.52	0.52
ED+PD	96	36	9	0.43	0.43	0.43
gAnswer	76	32	11	0.40	0.40	0.40

TABLE IV: Results on QALD-4

	Proceed	Right	Partially	Recall	Precision	F-measure
Svega	48	35	6	0.76	0.76	0.76
ED+PD	48	22	4	0.55	0.55	0.55
gAnswer	25	16	4	0.37	0.37	0.37

## VII. CONCLUSIONS

In this paper, we present Svega - an online natural language question answering system over knowledge bases. Moreover, the query intention of a natural language question is modeled by a query graph based on an entity-driven method. As a result, the problem of natural language question answering

over knowledge graph is converted to subgraph mapping. At last but not least, predicate vector and path vector are proposed to measure the semantic confidence between predicates and paths. Consequently, our approach is effective in the terms of recall and precision.

## ACKNOWLEDGMENT

The research is supported by The National Key Basic Research Program (No. 2018YFB1004000002), NSFC (No. 61672255), Science and Technology Planning Project of Guangdong Province, China (No.2016B030306003 and 2016B030305002), and the Fundamental Research Funds for the Central Universities, HUST.

## REFERENCES

- [1] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer, "Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia," *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.
- [2] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge," in *Proc. of WWW'07*. ACM, 2007, pp. 697–706.
- [3] G. M. Mazzeo and C. Zaniolo, "Answering controlled natural language questions on RDF knowledge bases," in *Proc. of EDBT'16*, 2016, pp. 608–611.
- [4] L. Zou, R. Huang, H. Wang, J. X. Yu, W. He, and D. Zhao, "Natural language question answering over RDF: a graph data driven approach," in *Proc. of SIGMOD'14*. ACM, 2014, pp. 313–324.
- [5] H. Bast and B. Buchhold, "An index for efficient semantic full-text search," in *Proc. of CIKM'13*. ACM, 2013, pp. 369–378.
- [6] L. Shao, Y. Duan, X. Sun, H. Gao, D. Zhu, and W. Miao, "Answering who/when, what, how, why through constructing data graph, information graph, knowledge graph and wisdom graph," in *Proc. of SEKE'17*. KSI, 2017, pp. 1–6.
- [7] S. Ferré, "squall2sparql: a translator from controlled english to full sparql 1.1," in *Proc. of Working Notes for CLEF'13*. Springer, 2013.
- [8] K. Xu, Y. Feng, S. Huang, and D. Zhao, "Question answering via phrasal semantic parsing," in *Proc. of CLEF'15*. Springer, 2015, pp. 414–426.
- [9] S. He, Y. Zhang, K. Liu, and J. Zhao, "Casia@v2: A mln-based question answering system over linked data," in *Proc. of CLEF'14*. Springer, 2014, pp. 1249–1259.
- [10] J. Daiber, M. Jakob, C. Hokamp, and P. N. Mendes, "Improving efficiency and accuracy in multilingual entity extraction," in *Proc. of I-Semantics'13*. ACM, 2013, pp. 121–124.
- [11] S. Schuster and C. D. Manning, "Enhanced english universal dependencies: An improved representation for natural language understanding tasks," in *Proc. of LREC'16*, pp. 2371–2378.
- [12] M. C. D. Marnee and C. D. Manning, "Stanford typed dependencies manual," Stanford University, Tech. Rep., 2008.
- [13] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, R. Cyganiak, and S. Hellmann, "Dbpedia - a crystallization point for the web of data," *J. Web Sem.*, vol. 7, no. 3, pp. 154–165, 2009.
- [14] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proc. of EMNLP'14*. ACL, 2014, pp. 1532–1543.
- [15] P. Yuan, P. Liu, B. Wu, H. Jin, W. Zhang, and L. Liu, "Triplebit: a fast and compact system for large scale RDF data," *PVLDB*, vol. 6, no. 7, pp. 517–528, 2013.

# Software Process Selection based upon Abstract Machines for Slow Intelligence Systems

Shi-Kuo Chang, Jinpeng Zhou, Akhil Yendluri and Kadie Clancy  
Department of Computer Science  
University of Pittsburgh, Pittsburgh, PA 15238, USA  
{schang, jiz150, aky13, kdc42}@pitt.edu

**Abstract**— The Abstract Machine Model was developed by Chang to formalize the decision cycles of slow intelligence systems. It turns out the selection of software process can also be regarded as a slow intelligence system. In this paper we formalize abstract-machine prototypes for different software process models such as waterfall model, incremental model, spiral model, extreme programming model and scrum model. A Software Process Generator SPG was implemented to generate software process models based upon design considerations. Initial evaluation by undergraduate students using SPG to learn software processes suggests further improvements to make it a useful learning tool.

**Keywords**—slow intelligence system, software process models, abstract machine model, component-based software engineering, software process learning tool.

## 1 Introduction

The slow intelligence system is an approach to design human-centric psycho-physical systems. A slow intelligence system (SIS) is a system that (i) solves problems by trying different solutions, (ii) is context-aware to adapt to different situations and to propagate knowledge, and (iii) may not perform well in the short run but continuously learns to improve its performance over time. The general characteristics of a slow intelligence system include enumeration, propagation, adaptation, elimination, concentration and multiple decision cycles [1]. In our previous work, an experimental test bed was implemented that allows designers to specify interacting components for slow intelligence systems [2].

In this paper, we formalize abstract-machine prototypes for different software process models such as waterfall model, incremental model, spiral model, extreme programming (XP) model and scrum model. Inspired by recent research on design spaces [3, 4, 5], a software process design space characterized by eleven parameters can be used to assist the designer in finding an appropriate software process model.

The paper is organized as follows. Section 2 presents an abstract machine model for the computation cycles. Section 3 shows some preliminary work based on building finite state machine (FSM) for each model. Based on the observations of

preliminary work and the abstract machine of slow intelligence system (SIS), we further describe our abstract machines in Section 4. In Section 5 we present five prototypes to show how to use our abstract machine definition for different models. Once the abstract machine model is provided, a compiler can be constructed to generate the components. In Section 6 we describe the major steps of the generic Abstract Machine Compiler (AMC). Section 7 describes the Software Process Generator SPG we implemented to construct different process models based upon design parameters. Initial experimental results, discussion and conclusion are presented in Section 8.

## 2 The Abstract Machine Model for Computation Cycles

An SIS typically possesses at least two decision cycles. The first one, the *quick decision cycle*, provides an instantaneous response to environmental changes. The second one, the *slow decision cycle*, tries to follow the gradual changes in the environment and analyze the information acquired from the environments or peers or past experiences. The slow/quick decision cycles enable the SIS to both cope with the environment and meet long-term goals.

Complex SISs may possess multiple slow decision cycles and quick decision cycles. Most importantly, 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.

To model such decision cycles we introduce an abstract machine model of multiple computation cycles.

The Abstract Machine Model is specified by:  $(P, S, P_0, \text{Cycle}^1, \dots, \text{Cycle}^n)$ , where

$P$  is the non-empty problem set,

$S$  is the non-empty solution set, which is a subset of  $P_0$ ,

$P_0$  is the initial problem set, which is a subset of  $P$ ,

$\text{Cycle}^1, \dots, \text{Cycle}^n$  are the computation cycles.

Each computation cycle will start from an initial problem set and apply different operators such as  $+adap_{Aij-}$ ,  $-enum<$ ,  $>elim-$ ,  $=prop_{Aij+}$  and  $>conc=$  successively to generate new problem sets from old problem sets until a non-empty solution set is found. If a non-empty solution set is found, the cycle is completed and later the same computation cycle can be

repeated. If on the other hand no solution set is found, a different computation cycle is entered.

As an example the problem set  $P$  consists of problem elements  $p_1, p_2, p_3, \dots, p_n$ , and each problem element  $p_j$  is specified by a vector consisting of attributes  $A_{ij}$ . A computation cycle  $x$  will attempt to find a solution set by first adapting based upon input from the environment:  $P^{x0} + \text{adap}_{A_{ij}} = P^{x1}$  is to adapt based on attribute  $A_{ij}$ , for example, by appending  $A_{ij}$  to each element in  $P^{x0}$  to form  $P^{x1}$ . Then it may try to find related problem elements:  $P^{x1} - \text{enum} < P^{x2}$  where  $P^{x2} = \{y: y \text{ is related to some } x \text{ in } P^{x1}, \text{ e.g. } d(x,y) < D\}$ .

Next it may try to eliminate the non-solution elements:  $P^{x2} > \text{elim} - P^{x3}$  where  $P^{x3} = \{x: x \text{ is in } P^{x2} \text{ and } x \text{ is in } S\}$

Finally the solution elements (or alert messages if there are nosolutions) may be propagated to peers:  $P^{x3} = \text{prop}_{A_{ij}} + P^{x4}$  is to export/propagate attribute  $A_{ij}$  to peers.

Therefore this computation cycle can be specified succinctly as follows:  $\text{Cycle}^x [\text{guard } x,y]: P^{x0} + \text{adap}_{A_{ij}} = P^{x1} - \text{enum} < P^{x2} > \text{elim} - P^{x3} = \text{prop}_{A_{ij}} + P^{x4}$ .

The above expression is a specification of the computation cycle, not a mathematical equation. This expression should be read and interpreted from left to right.

If  $P^{x4}$  is non-empty, the Abstract Machine will complete this cycle of computation and terminate at the end of  $\text{Cycle}^x$ , and it may later resume at the beginning of  $\text{Cycle}^x$ . Otherwise  $P^{x4}$  is empty and the Abstract Machine will jump to a different  $\text{Cycle}^y$ . This is specified by  $[\text{guard } x,y]$  where  $x$  is the current computation cycle if a solution set is found ( $P^{x4}$  is non-empty), and  $y$  is the computation cycle to enter if no solution set is found ( $P^{x4}$  is empty). Before an Abstract Machine completes its current computation cycle, it will propagate the solution set (or alert messages) to its peers.

In the above, the elimination operator can be replaced by the concentration operator, whenever the solution set is not known apriori. The concentration operator applies a predefined threshold to filter out problem elements below the threshold:  $P^{x1} > \text{conc} = P^{x2}$  where  $P^{x2} = \{x: x \text{ is in } P^{x1} \text{ and } \text{th}(x) \text{ above a predefined threshold } t\}$ .

### 3 Software Process Models

Software process models provide certain workflows for software development. Intuitively, we can use finite state machine (FSM) to illustrate these models. Each step in the software process can be represented as a state in FSM. The inputs and outputs of each state are corresponding to the requirements and products of each process step, which may include documents, program codes, user communications, test datasets, prototypes, and timings.

### 3.1 FSM for different models

Based on the state transition tables, we drew the sketches for five software process models. These are meant to illustrate the concept, and the specific details of each software process model may vary.

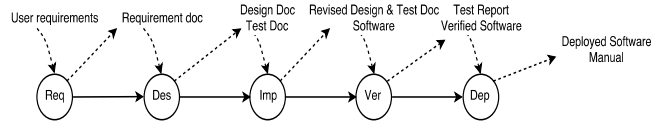


Fig 1. Waterfall model

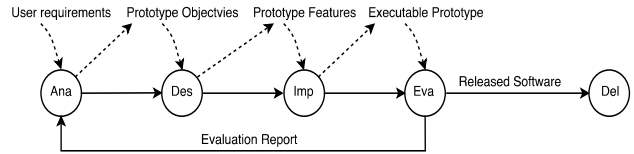


Fig 2. Incremental model

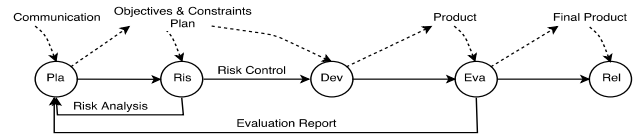


Fig 3. Spiral model

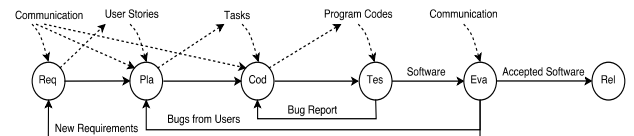


Fig 4. Extreme programming model

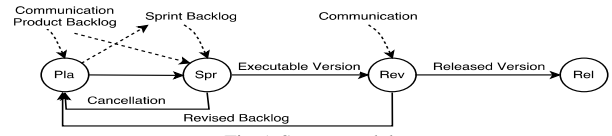


Fig 5. Scrum model

### 3.2 Observations

There are some key observations that inspire the definition of our abstract machines:

- All process models are based on a major workflow, starting from the user requirements to the final release of software. Thus, we can use operation cycles to represent the process flows. Furthermore, we need to bind to a starting point so that the machine starts from the first user requirements.
- The final purpose of a software process is to build a production software, which typically consists of different features, or objectives. These objectives derive from the original user requirements and are abstracted, implemented, and verified during the process. Thus, each state in the FSM actually can be presented as a set of

objectives, which is similar as the problems set in SIS abstract machine.

- An objective has a lifetime starting from user requirements to verification. Each step in the process will update or mark the objective with a new state. Thus, we can assign a color for each objective to represent its states during the entire process.
- The software engineering or project managing operations during the process can be represent by the operators. These operations can perform add/delete/update on each objective, including coloring.
- At some points, a step may have different successors based on certain situation. Thus, we also need a guard function to provide process control. Furthermore, we need to know these specific steps and situations.
- Different Models may have different behaviors in similar step. e.g., agile models do not need explicit and complete user requirements and system design.

## 4 Abstract Machines for Software Process Models

### 4.1 Machine definition

Based on the observations in Section 3.2, we can define the abstract machine, by modifying the abstract machine for SIS:

$$M_{spm} = [P, S, P_0, Cycles], \text{ starting from cycle}_1$$

Where  $P$  is the objective set,  $P_0$  is the initial set.  $S$  is the solution set which includes all objectives that have been implemented and verified.  $Cycles$  are sequences of different operators. As mentioned above, this abstract machine should start from a certain starting point. By default, we set the starting point to be the beginning of  $cycle_1$ .

### 4.2 Objectives

An objective is corresponding to a certain user requirement or feature for the target software. We define four colors for each objective.

- White: it's abstracted or included into the software process.
- Yellow: it's implemented and ready for verification.
- Red: failed in verification, such as failed in testing or user acceptance.
- Green: it's verified and ready to be used.

### 4.3 Guard function

In order to support process control, we also define a guard function by extending the SIS abstract machine's guard function:

$$guard [ a, b, P\_checkpoint, constraint, P\_newInit ]$$

Where  $a$  is the current cycle,  $b$  is the target cycle. When it reaches the  $P\_checkpoint$  of cycle  $a$ , it will check whether  $a$  certain constraint is satisfied. If so, it will jump to start cycle  $b$  with  $P\_newInit$ .

## 4.4 Operators

To provide a general definition, we defined six basic operators for software process models:

- Abstract (**abst**): Translate user-described requirements to software-engineering requirements. This operator will initialize objectives with color white. We further divide this operator into two types: (1) *Enumerative Abstract* (**-abst<**). This type will process every objective; (2) *Selective Abstract* (**=abst=**). This type will process selected objectives only. It does not guarantee that all objectives will be processed.
- Design (**desi**): Functionalize non-green objectives to module-level or function-level objectives. It also has two types: enumerative (**-desi<**) and selective (**=desi=**).
- Implement (**=impl=**): Implement white/red objectives to real product-level objectives and color them as yellow. We assume that implementation is strictly bound to the objectives. E.g., each objective will be implemented as a module. Thus, only selective is required here.
- Test (**=test=**): Validate yellow objectives and color them as green or red. Similarly, only selective is required.
- Adjust (**=adju=**): Modify objectives based on (external) non-engineering issues, such as user communications, funding issues. This operator is essential for agile models.
- Release (**=rele+**): release all green objectives. This operator is similar as the propagator in SIS abstract machine, which generates some outputs to environment.

## 5 Abstract Machine Prototypes

Based upon the observations in Section 3.2 and the formal specification of abstract machines in Section 4, the software process models can now be formally specified. Again, these are meant to illustrate the concept, and the specific details of each software process model may vary.

### 5.1 Waterfall

Prototype:

- Cycle\_1: guard[1, 2, P2, NULL, P2]
  - P0 **-abst**< P1 **-desi**< P2
- Cycle\_2: guard[2, 2, P2, has\_non-green, P2]
  - P0 **=impl**= P1 **=test**= P2 **=rele**+ P3

In cycle\_1: it requires a complete abstraction and design.

In cycle\_2: it will go through implementation, test, and final release. Whenever there's a failed objective after test, it should go back to the implementation and redo the following process again.

The machine halts at P3 in cycle\_2.

## 5.2 Incremental

Prototype:

- Cycle\_1: guard[1, 2, P2, NULL, P2]
  - P0 **=abst**= P1 **-desi**< P2
- Cycle\_2: guard[2, 2, P2, one\_non-green, P2], guard[2, 1, P3, all\_green, NULL]
  - P0 **=impl**= P1 **=test**= P2 **=rele**+ P3

In cycle\_1: the abstraction can be incomplete. But the design should take care of all current objectives.

In cycle\_2: different from waterfall model, here it will go back to cycle\_1 for next increment when the current increment is finished.

The machine halts when no more increment is required, which means P0 in cycle\_1 is empty.

## 5.3 Spiral

Prototype:

- Cycle\_1: guard[1, 1, P3, one\_red, NULL], guard[1, 2, P3, no\_red, P1]
  - P0 **=abst**= P1 **=impl**= P2 **>+adju**= P3
- Cycle\_2: guard[2, 1, P5, all\_green, NULL]
  - P0 **-abst**< P1 **-desi**< P2 **=impl**= P3 **=test**= P4 **=rele**+ P5

In cycle\_1: it required to build a simple prototype to evaluate the risk. Thus, we need an implement and adjust operator here. If the risk evaluation says good, then it will transfer to the cycle\_2 for further process.

In cycle\_2: similarly, it will go back to cycle\_1 until there's no more work to do.

The machine halts when P0 in cycle\_1 is empty.

## 5.4 Extreme programming

Prototype:

- Cycle\_1: guard[1, 2, P2, NULL, P2]
  - P0 **=abst**= P1 **=desi**= P2
- Cycle\_2: guard[2, 2, P2, hours, P2], guard[2, 3, P3, days, P3]
  - P0 **>+adju**= P1 **=impl**= P2 **=test**= P3
- Cycle\_3: guard[3, 1, P2, non-empty, P2]
  - P0 **=rele**+ P1 **>+adju**= P2

In cycle\_1: it does not require complete requirements or design. It selects a certain user story to implement.

In cycle\_2: The process is controlled based on timing. Thus, the major constraint here should be related to the specific time deadline. Furthermore, we need the adjust operator to make sure the implementation and testing are sensitive to user communications.

In cycle\_3: after few days, it's supposed to generate a version for quick release. Then it can keep finishing rest or new objectives based on feedbacks.

The machine halts when P2 in cycle\_3 is empty, which means feedbacks confirm that no more objectives.

## 5.5 Scrum

Prototype:

- Cycle\_1: guard[1, 2, P2, NULL, P2]
  - P0 **=abst**= P1 **=desi**= P2
- Cycle\_2: guard[2, 2, P2, day, P2], guard[2, 3, P2, Weeks, P2]
  - P0 **=impl**= P1 **=test**= P2
- Cycle\_3: guard[3, 1, P3, non-empty, P3]
  - P0 **>+adju**= P1 **=rele**+ P2 **>+adju**= P3

In cycle\_1: it does not require complete requirements or design. It selects a certain backlog and launch it as a sprint.

In cycle\_2: it starts a sprint. Inside this sprint, no requirement modification is allowed.

In cycle\_3: in the end of a sprint, the developing team will review the sprint. Then, based on user communications, the set of objectives (backlogs) will be updated.

The machine halts when P3 in cycle\_3 is empty, which means all backlogs are finished.

## 6 The Software Process Model Generator

The Abstract Machine Model is a formal specification of the computation cycles of a slow intelligence system. Once the abstract machine model is provided, a Software Process Generator SPG can be used to generate software process model based upon design considerations.



The input to the SPG are the various software process models SPM specified by Abstract Machines. The user/designer can interact with SPG to select the appropriate design choices. After a software process model is selected, the output in the form of a web page is generated by SPG. This web page describes the software process and can be used by the user/designer to track a project according to the selected process model.

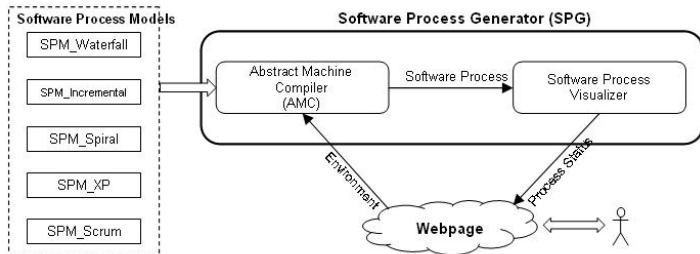


Figure 6. The software process model generator.

User/designer interacts with SPG through the webpage, such as tracking process status, updating environment and so on. The user/designer can make design choices, update requirements if possible, modify environment, etc. The SPG updates the current software process status based on the the selected SPM, and user interactions.

In what follows we describe the major steps of how the AMC and visualizer work in SPG.

**Step 1: Adapt inputs from the user**

The AMC will first invoke an interface to receive a set of features from the user/designer. This set may include necessary information of the project for simulating different process models. It is submitted by the user/designer through a webpage (see box below).

In practice, the user/designer makes choices to specify the desired parameters (see Figure 7 in Section 7).

```

Interactor:
    // adapt current data with user inputs
    while(user inputs are not required) {
        sleep();
    }
    while(currentData is not updated) {
        adaptCurrentDataWithInputs();
    }
    send DATA_READY msg to Controller;
    
```

AMC Controller manages the process by controlling the state machine (see box below).

```

Controller:
    // manage the process of one process model
    while(true) {
        while (stateMachine.precheck()) {
            trigger Interface,
            // wait for user inputs
            stateMachine.wait(DATA_READY);
        }
        stateMachine.perform(currentData);
        send msg to visualizer if necessary;
    }
    
```

The State Machine will determine the action and the output. It may give several tries. For example, two solutions can be applied to one certain state when given certain input (see box below).

```

State Machine:
    // manage the states
    enum Status {
        State0,
        State1,
        ...;
    }

    State currentState = State0;//initial state

    bool precheck();// return true if current state requires inputs

    int wait(msg); //wait until certain msg is received

    void perform(Data currentData) {
        // based on certain state and certain input
        switch (currentState) {
            case 'State0':
                perform accordingly;
                break;
            case 'State1':
                perform accordingly;
                break;
            ....
        }
    }
    
```

**Step 2: Simulate process models**

The AMC is responsible to simulate every model.

To answer the question “Which process model is the best?”: (1) each state will be measured to make sure the current project status is in a “safe-zone”. Depending on the results of the measurement, either enumeration operator or elimination operator can be applied; (2) a specific function, which takes certain parameters into consideration, will be applied to evaluate the performance of each model.

If a certain model is simulated successfully, the evaluation results and simulation logs will be presented to the user on demand. If a certain state of model A violates project’s configuration, the AMC will terminate A’s simulation and start the next un-simulated model.

**Step 3: Model selection**

After all models have been simulated, the AMC will choose the model with the best evaluation result to the user, and present it as the optimal solution to the user.

**Step 4: Model visualization**

A process visualizer is built to show the simulation to the user. There are three cases that the visualizer is invoked: (1) the running simulation requires dynamic or runtime inputs from users; (2) the user requests to check current simulation status; (3) the AMC has decided the optimal solution (see box below).

```

Visualizer:
// present the AMC results
void showCurrent(); // invoked by the AMC or the user

void showOptimal(); // invoked by the AMC

```

## 7 An Experimental SPG System

An experimental SPG was implemented. Software processes are characterized by the following eleven design parameters:

- Early Functionality (iteratively introduce features, only produce final product),
- Feature Adaptation (impossible, flexible),
- User Involvement (C only initially, at requests, frequent feedback),
- Documentation (not produced, produced),
- Experienced Team (requested, not required),
- Model Type (C linear, iterative),
- Planning and Scheduling (upfront, continuous),
- Risk Mitigation (yes, no),
- Project Size (C small, medium, large),
- Prototypes (used, not used).
- Cross-platform development (no, yes)

Parameter	Value
Early Functionality	<input checked="" type="radio"/> Iteratively introduce features <input type="radio"/> Only produce final product
Feature Adaptation	<input type="radio"/> Impossible <input checked="" type="radio"/> Flexible
User Involvement	(Initially) <input type="radio"/> 0% <input type="radio"/> 10% <input type="radio"/> 20% <input type="radio"/> 30% <input type="radio"/> 40% <input type="radio"/> 50% <input type="radio"/> 60% <input type="radio"/> 70% <input type="radio"/> 80% <input type="radio"/> 90% <input type="radio"/> 100% (Frequent feedback)
Documentation	<input checked="" type="radio"/> Not produced <input type="radio"/> Produced
Experienced Team	<input type="radio"/> Requested <input type="radio"/> Not Required
Model Type	(Linear) <input type="radio"/> 0% <input type="radio"/> 10% <input type="radio"/> 20% <input type="radio"/> 30% <input type="radio"/> 40% <input type="radio"/> 50% <input type="radio"/> 60% <input type="radio"/> 70% <input type="radio"/> 80% <input type="radio"/> 90% <input type="radio"/> 100% (Iterative)
Planning and Scheduling	<input type="radio"/> Upfront <input type="radio"/> Continuous
Risk Mitigation	<input type="radio"/> Yes <input type="radio"/> No
Project Size	(Small) <input type="radio"/> 0% <input type="radio"/> 10% <input type="radio"/> 20% <input type="radio"/> 30% <input type="radio"/> 40% <input type="radio"/> 50% <input type="radio"/> 60% <input type="radio"/> 70% <input type="radio"/> 80% <input type="radio"/> 90% <input type="radio"/> 100% (Large)
Prototype	<input type="radio"/> Used <input type="radio"/> Not Used
CrossPlatform	<input type="radio"/> No <input type="radio"/> Yes

SubmitParameters

Figure 7. The designer specifies the needed parameters.

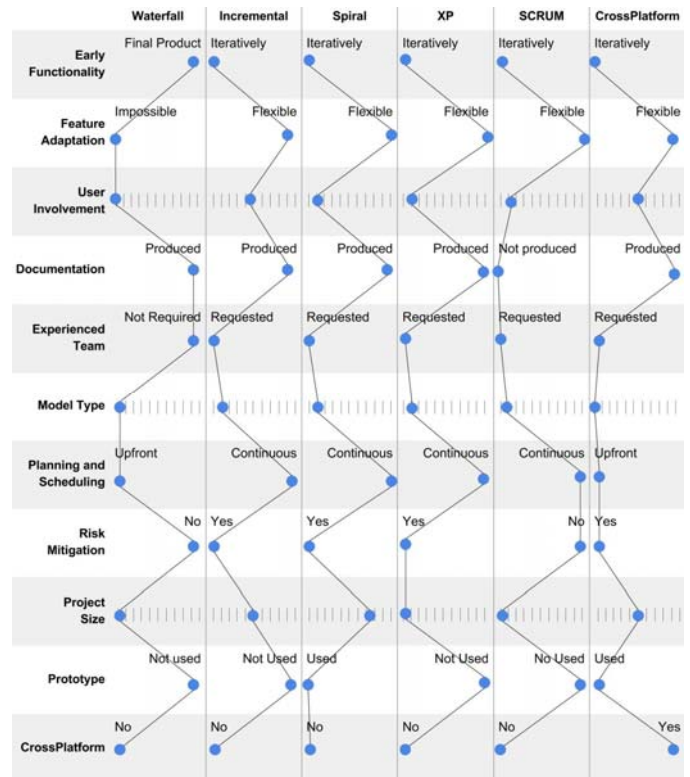


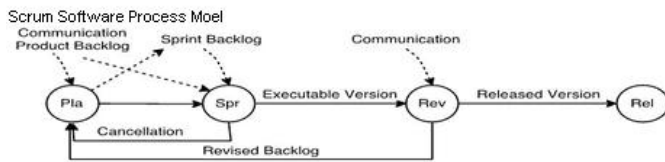
Figure 8. Stored profiles of software processes.

A parameter with continuous value is indicated by the letter 'C'. In practice the designer can specify a continuous parameter from 0% to 100% in 10% increments (see Figure 7). Parameters not specified by the designer are not used in calculating the optimum. The designer-specified profile (Figure 7) is compared with the pre-specified profiles of software process models (Figure 8) and the one with minimum distance is the SPG-recommended software process model (Figure 9).

**waterfall : 3**  
**incremental : 1**  
**spiral : 1**  
**xp : 1**  
**scrum : 0 (optimal)**  
**crossplatform : 1**

Figure 9. Scrum software process recommended by SPG.

When the designer clicks on the link for Scrum, a Scrum software process simulator is provided. As shown in Figure 10 the designer can simulate the execution of the software process by clicking on the actions associated with its current state. In addition a tutorial on Zoho is provided as the recommended tool for Scrum software development.



**Sprint**  
 Go back  
 Deliverables for each Sprint  
 If there is backlog go to next Sprint  
 If no backlog go to end

### Zoho Sprints Tutorial

1. Go to "<https://www.zoho.com/sprints/>" and create account.
2. Login to your account.
3. Select "+project" in home page.

Figure 10. Scrum software process simulator.

## 8 Discussion

The Abstract Machine Model is a formal specification of the software processes, based upon which the SPG tool was implemented: <http://ksiresearchorg.ipage.com/spg/>. The SPG tool was used by 32 undergraduate student project groups in two software engineering classes at the Univ. of Pittsburgh to experiment with software processes. The students were then asked to evaluate the SPG. In response to the question whether the SPG tool enhanced understanding of the software processes, the average rating is 0.35, between “a lot” (0.5) and “a little” (0.25). There are comments that the individual model pages are the most helpful, and percentage as a parameter value is a little vague.

The current SPG was implemented with pre-defined web pages representing the states of different software process models. We are implementing a better version by dynamically generating customized web pages (the process states). Both pre-defined software processes as well as hybrid software processes can then be generated, thus making the SPG a more powerful learning tool. More information is added to the individual model pages, and parameters are better explained. With more improvements the SPG tool can become a valuable learning tool.

### Acknowledgement:

This research was supported in part by KSI Research, USA.

### References:

[1] Shi-Kuo Chang, "A General Framework for Slow Intelligence Systems", *International Journal of Software Engineering and Knowledge Engineering*, Volume 20, Number 1, February 2010, 1-16.

[2] Shi-Kuo Chang, Sen-Hua Chang, Jun-Hui Chen, Xiao-Yu Ge, Nikolaos Katsipoulakis, Daniel Petrov and Anatoli Shein, "A Slow Intelligence System Test Bed Enhanced with Super-Components", *Proceedings of 2015 International Conference on Software Engineering and Knowledge Engineering*, Pittsburgh, USA, July 6-8, 2015, 51-63.

[3] E. Murphy-Hill, T. Zimmermann, C. Bird, and N. Nagappan, The design of bug fixes, in *Proc. Int. Conf. Softw. Eng.*, 2013, pp. 332341.

[4] Murphy-Hill, E., Zimmermann, T., Bird, C., and Nagappan, N., 2015. The Design Space of Bug Fixes and How Developers Navigate It. *IEEE Transactions on Software Engineering* 41, 1, 65-81.

[5] S. CARD, J. MACKINLAY, AND G. ROBERTSON. A MORPHOLOGICAL ANALYSIS OF THE DESIGN SPACE OF INPUT DEVICES. *ACM TRANSACTIONS ON INFORMATION SYSTEMS*, 9:9122, 1991.

# Evolutionary propositionalization of multi-relational data

Valentin Kassarnig  
Institute of Software Technology  
Graz University of Technology  
Graz, Austria  
kassarnig@ist.tugraz.at

Franz Wotawa  
Institute of Software Technology  
Graz University of Technology  
Graz, Austria  
wotawa@ist.tugraz.at

**Abstract**—Propositionalization has been proven to be a very effective solution for multi-relational data mining tasks. Traditional propositionalization approaches follow a two-step principle: transforming the relational data into a single, flat table and applying a propositional learning algorithm. During the transformation the target table gets expanded by adding many new features summarizing the information of the non-target tables. Based on the used feature construction strategy, this leads to a table of very high dimensionality with a lot of irrelevant and/or redundant features that has a negative effect on the predictive performance. In this paper, we propose an alternative propositionalization approach that evaluates the features already during the construction phase and reports only a subset of highly predictive features to the propositional learner. We present an implementation of this approach that adapts a state-of-the-art propositionalization technique and combines it with a genetic algorithm to search for an optimal feature subset. Our experiments on a number of benchmark datasets reveal superior predictive performance of the approach compared to traditional two-step methods making it a considerable extension for any propositionalization algorithm.

## I. INTRODUCTION

The rapid advance of data mining techniques during the last decades has lead to countless real-world applications, such as forecasting stock prices [1]–[3], predicting customer behavior [4], [5], or detecting credit card fraud [6]–[8]. However, mining relational data is still problematic since conventional data mining algorithms can be only applied to propositional data. A common approach to solve this problem is *Propositionalization* which typically follows a two-step principle: First, transform the relational data into a single, flat table and second, apply a propositional learning algorithm on the transformed data. This principle is also referred to as *Polka* (named after the two-step dance) and is illustrated in Fig. 1a. Such two-step propositionalization methods have been successfully applied on numerous ILP benchmark tasks as well as real-world applications such as Kaggle competitions [9], [10].

The separation of the two steps in Polka has the downside that the feature construction process is completely isolated from the learning task. Due to the lack of any evaluation of the feature construction process all possible features need to be constructed. Consequently, this results in a table of unnecessarily high dimensionality with a lot of irrelevant

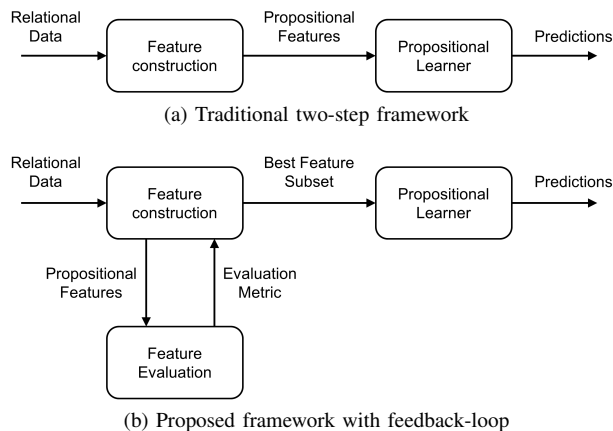


Fig. 1: Traditional and proposed propositionalization framework. The proposed framework has a feature evaluation step with a feedback loop to the feature construction that allows evaluating partial solutions and adapt the feature construction accordingly.

(and often redundant) features. Furthermore, in the case of very complex databases and sophisticated feature construction strategies, the exhaustive feature construction may be even intractable [11].

We propose to adapt the traditional approach by adding a feature evaluation step with a feedback loop to the feature construction, as illustrated in Fig. 1b. This change allows us to evaluate the predictive power of feature subsets and adapt the feature construction accordingly. So, in addition to the actual transformation, propositionalization performs a feature subset selection. This class of problems is proven to be NP-complete [12] because only the exhaustive evaluation of all possible subsets would guarantee an optimal solution. In order to tackle this problem we utilize a genetic algorithm (GA) which has a very high rate of convergence to find a near-optimal solution [13]. The GA searches through the space of all possible feature subsets and evaluates the predictive power of the candidate solutions. By only constructing a constant number of features per generation we are capable of propositionalizing complex databases where exhaustive feature construction would be intractable.

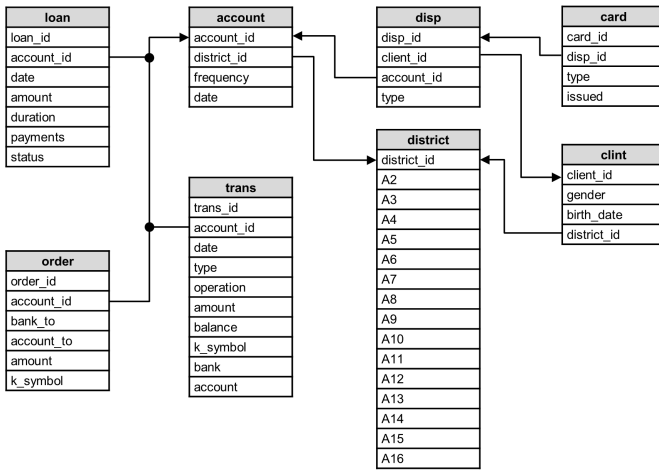


Fig. 2: Data model of the financial database from the PKDD 1999 Discovery Challenge.

In this paper, we use the financial database from the PKDD 1999 Discovery Challenge [14] as a running example to explain different concepts. This database captures information about a bank offering services to private clients. The goal is to find out, what clients need to be watched carefully to minimize the bank losses. Fig. 2 shows the corresponding data model with the table *loan* containing the target attribute *status* that indicates whether there were any repayment problems with a given loan or not.

Our main contributions through this paper are as follows:

- We propose an extension of the traditional two-step propositionalization framework
- We introduce a genetic-based algorithm to propositionalize relational data for multi-relational classification tasks
- We conduct an experimental evaluation of our method on a number of well-known benchmark tasks and compare its performance with those of state-of-the-art methods

## II. RELATED WORK

The problem of mining relational data has been extensively studied in the past. The two main approaches for this problem are Inductive Logic Programming (ILP) and Propositionalization [15]. Although there are cases where ILP methods have been successfully applied [16], propositionalization approaches generally outperform them in terms of speed [17], scalability [17]–[20], and predictive performance [20]–[22]. Furthermore, ILP-based systems perform poorly on a noisy domain compared to numerical propositionalization [15], [17], [21].

Recent works have successfully used aggregation-based propositionalization approaches to automatically mine big databases (up to 100 GB of raw data) [9], [10]. Their experiments showed that with exhaustive feature construction methods an enormous amount of computing power is required to process such massive databases. Even when the workload was distributed among 60 CPUs it took their best scaling

method nearly 13 hours to process a database of about one GB raw data [23].

Different approaches have attempted to overcome such shortcomings of traditional two-step propositionalization. The aggregation-based algorithm PRORED [24] avoids exhaustive feature construction by using stochastic optimization. Based on heuristically determined probabilities only a subset of attributes and aggregate functions is chosen to construct features. The used heuristic function makes attributes of tables further away from the target table less likely to be chosen. However, their gain in scalability comes at the cost of reduced accuracy.

Genetic Algorithms (GA) have been demonstrated to be useful tools for propositionalization. Braud and Vrain [25] propose a logic-oriented propositionalization approach with a GA-based feature construction. Their GA optimizes individual features, represented as Horn clause-patterns, through the operations union, intersection, variable isolation, variable move, split, and merge. Similarly, Alfred [26] utilizes a GA as extension of his propositionalization framework DARA [27] to find the most predictive patterns of combined attributes. Results indicate that this extension improves the efficiency as well as the performance. Furthermore, in the field of propositional data mining, GAs have been found to be robust and powerful means to find near-optimal subsets of features [28]–[31]. While the mentioned GA-based propositionalization approaches utilize GAs to optimize the predictive power of individual features, our framework optimizes the predictive power of the final feature subset. That is, a feature is either selected or not but it is not changed in any way.

## III. METHOD

### A. Genetic algorithm

Our proposed approach adapts the traditional two-step framework by adding a feature evaluation step that evaluates partial solutions (see Fig. 1). For this purpose we utilize a standard Genetic Algorithm (GA) [32] with a rank-based selection strategy. Individuals are encoded as binary strings of length  $N$  where  $N$  is defined by the total number of possible features under the current feature construction strategy. Every position in the binary string indicates either the presence or absence of a particular feature. A feature can be either an attribute of the target table or a construct based on the attributes of other tables. See Subsection III-C for more details about the feature construction process.

We will refer to this implementation as *GenPro* (GENetic PROpositionalization) throughout the remaining paper. In our experiments we parameterized the GA with the following values:

- Population size: 20
- Max. Number of generations: 150
- Probability of initial selection: 0.01
- Probability of crossover: 0.85
- Probability of mutation:  $\frac{1}{N}$

Every population consists of 20 individuals where each individual represents a candidate solution encoded as a binary

string. Over the course of 150 generations those individuals are evaluated, selected, combined and mutated in order to maximize their fitness. The probability of initial selection determines how likely a feature gets chosen to be part of an individual in the initial population. When creating an offspring for the next generation, the probability of crossover defines whether the offspring is derived by combining two individuals or just mutating one. The probability of mutation specifies how likely a bit in the binary string is flipped during the mutation operation. Because  $N$  corresponds to the length of the binary string, on average only one feature per individual is either added or removed.

### B. Fitness function

The goal of the fitness function is to evaluate the predictive power of a given feature subset. For this we perform a stratified 10-fold cross-validation with a Classification and Regression Tree (CART) [33] and determine the predictive accuracy, defined as

$$accuracy(x) = \frac{\text{Correct predictions}}{\text{Total number of examples}} \quad (1)$$

where  $x$  is a bit-string encoded individual representing a subset of features.

As fitness measure for the GA we consider two different metrics. For the first one,  $Fit_1$ , we use simply the resulting accuracy from the cross-validation:

$$Fit_1(x) = accuracy(x) \quad (2)$$

The second metric,  $Fit_2$ , takes additionally the cost of creating the feature subset into account, as suggested by Yang et al. [29]. In our case, this corresponds to the number of features in the subset. That is, the fitness function favors smaller subsets in order to improve generalizability [34] and reduce computational costs. We define this fitness metric for an individual  $x$  as

$$Fit_2(x) = \frac{accuracy(x)}{|x| + \sum_{i=1}^{|x|} x_i} \quad (3)$$

where  $x_i$  is the feature at position  $i$ . The sum of  $x$  corresponds to the number of active features in the subset and  $|x|$  is the total number of possible features.

### C. Feature construction

The feature construction strategy of a propositionalization algorithm determines the total number of possible features. While exhaustive approaches create all of them up front, we construct them on-the-fly as needed. For our GenPro implementation we adopt the RELAGGS algorithm [35] as feature construction strategy. Note that we could have used here any propositionalization technique that is based on the Polka scheme.

We reimplemented the basic RELAGGS version as presented in its original paper [35]. RELAGGS propagates the identifiers of the target instances to the non-target tables and summarizes then their attributes through numeric aggregation.

While for numeric attributes the standard SQL aggregate functions  $MIN$ ,  $MAX$ ,  $SUM$ , and  $AVG$  are used, nominal attributes are summarized by counting the occurrences of every distinct value. Additionally, a feature representing the group size of a summarized table is created for every summarized table. The RELAGGS paper gives no indication of how to treat *Date* attributes. Thus, we have extracted from every date its year, month, week number, day of the year, and weekday and treat each of them as a numeric attribute. As described in [36] we set in our experiments the *maximum cardinality* parameter for nominal attributes to 100. It is not specified what upper limit of literals was used in the experiments and we just presume a value of 6 which gives us a sufficiently large number of possible features. In contrast to the original version we allow tables to appear twice in a clause in order to capture additional information about the past [37].

The following example illustrates the basic principle of the feature construction. At first, the target identifier *loan\_id* is propagated to the associated table *account* as illustrated in Fig. 3a. This association has a  $N:1$  multiplicity and thus, every target instance can be directly linked to a particular *account* instance. Consequently there is no summarization needed and the attributes can be simply added to the target table. Next, the target identifiers are further propagated to the table *trans*. This association has a multiplicity of  $1:N$  which implies that every target instance can belong to multiple *trans* instances. Therefore, the table needs to be summarized so that every target instance corresponds to exactly one row. This is done by applying the previously discussed aggregate functions to each attribute according to its data type. The example in Fig. 3b shows how the numeric attribute *amount* is summarized by applying four different aggregate functions. Each of those four new attributes represents a feature that is eventually added to the target table. While this example illustrates the general idea of the feature construction, in GenPro we do not summarize entire tables but only the attributes needed to create the active features (the ones with a 1 in the binary string) of the current individual.

## IV. EVALUATION

### A. Setup

In order to find out how propositionalization benefits from the proposed framework we applied our method GenPro, including both fitness functions, on a number of benchmark tasks. For direct comparison, we performed the same tasks with our implementation of the RELAGGS algorithm, which uses the same feature construction strategy as GenPro but creates all features exhaustively. In addition to the basic version, we also tested RELAGGS with a follow-up feature selection (FS) and dimensionality reduction (DR) step. For FS we used the top 10% features based on an ANOVA F-test. DR was performed through a Principal Component Analysis (PCA) where the feature space was reduced to 10% of its original size.

Since the actual performance depends very much on the used learner we used three different models, namely CART,

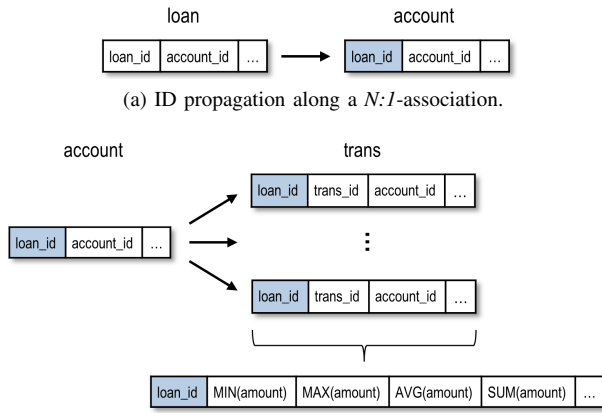


Fig. 3: Simplified illustration of the RELAGGS feature construction strategy. The colored attribute fields indicate the propagated target identifiers.

Random Forest (RF), and SVM. We used their respective Scikit-learn [38] implementations with 100 estimators for the RF and default parameters other than that. All reported results are based on ten independent and stratified 10-fold cross-validations. For the sake of a fair comparison we stopped the GA in all cases only after 150 generations regardless of whether the fitness score has already converged or not.

We performed the experiments on a PC with Windows 10 Professional, an Intel Core i7 CPU with 2x 1.70 GHz, and 8 GB RAM. The code was written in Python 2.7 and no optimizations techniques, such as parallel or GPU computing, were used. However, constructed features were cached and reused when needed in order to avoid redundant computations.

## B. Datasets

We used the financial database [14] from the PKDD 1999 discovery challenge as primary benchmark task to evaluate different GenPro variants. The dataset consists of eight tables (see Fig. 2) and more than a million records. The goal is to predict for a given loan whether there will be any repayment problems. The target table *loan* has 682 instances of which 606 did not cause any problems and only 76 had repayment issues. As suggested by Frank et al. [39] we only used transactions dated before the loan was granted in order to avoid peeking at retrospective data.

Further experiments were performed on the Mutagenesis database [40], Medical database [41], Hepatitis database [39], and the two Musk datasets [42] to cover a wide spectrum of different problem types.

Table I provides an overview of the used datasets and their properties. The last column *# features* describes the total number of features that can be constructed using our RELAGGS implementation.

Dataset	# tables	# target rows	# attributes	# features
Financial	8	682	55	675
Hepatitis	7	500	26	57
Mutagenesis	3	188	14	43
Medical	3	806	64	232
Musk large	2	102	170	665
Musk small	2	92	170	665

TABLE I: Overview of the benchmark datasets

Method	CART	RF	SVM
RELAGGS	0.904	0.918	0.889
RELAGGS + DR	0.819	0.896	0.889
RELAGGS + FS	0.906	0.932	0.889
GenPro with $Fit_1$	0.964	0.950	0.885
GenPro with $Fit_2$	<b>0.971</b>	0.959	0.891

TABLE II: Predictive accuracies on the financial task. Evaluation of different RELAGGS and GenPro variants for propositionalization in combination with Classification and Regression Trees (CART), Random Forests (RF) and Support Vector Machines (SVM) as predictive models.

## V. RESULTS

### A. Financial task

This section discusses the results on the financial task. It is divided into three parts: The first paragraph compares the predictive performance of GenPro and RELAGGS, and discusses the impact of different machine learning models. The second paragraph discusses the differences between the two GenPro variants before the last paragraph presents the results of a meta-comparison with other propositionalization methods and multi-view approaches.

*Predictive performance:* Table II shows the achieved accuracies of different GenPro and RELAGGS variants on the financial task. With an accuracy of 97.1% the best result was achieved by GenPro with the  $Fit_2$  fitness function and the CART model. This configuration outperformed the best RELAGGS variant by +4%. Moreover, both GenPro variants with either CART or RF models achieved superior results over any RELAGGS configuration. We can also observe that RELAGGS benefits from feature selection (FS) while dimensionality reduction (DR) has rather a negative effect. Furthermore, the RF classifier seems to better handle the data's high dimensionality leading to better results for RELAGGS. On the other side, GenPro works best with the CART model which was also used to determine the fitness scores. SVM performs poorly on this task and seems completely inapplicable here. Note that both RELAGGS and GenPro produce the same features but in contrast to RELAGGS, GenPro uses only a subset of it for the learning task.

GenPro's superior performance over RELAGGS comes at the cost of decreased computational efficiency. In our experiment a single 10-fold cross-validation with RELAGGS took nearly one minute. On the other side, it took GenPro about three minutes to complete the same task. However, note that GenPro was stopped after 150 generations but converged

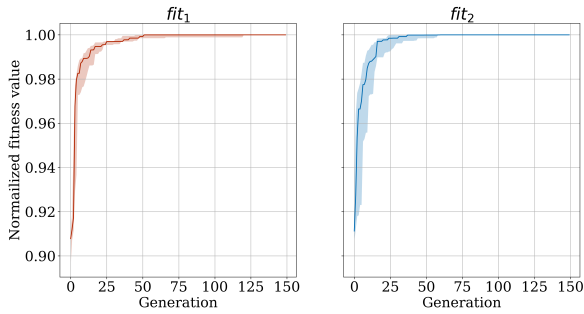


Fig. 4: GenPro’s convergence behavior of the fitness value with the two fitness functions  $fit_1$  and  $fit_2$ .

already way earlier (see Fig. 4).

*GenPro variants:* Fig. 4 illustrates GenPro’s convergence behavior of the fitness value with either of the two fitness functions. The solid lines indicate the median of the normalized fitness values of ten test runs while the transparent area around it is bound by the respective 25% and 75% percentile. We can see that both fitness functions result in a very similar convergence behavior and moreover, in most cases they converge within 75 generations.

In terms of selected feature subset, the two fitness functions produce very different results. On the financial task,  $fit_1$  and  $fit_2$  lead to average subset sizes of 34.3 and 7.3 features, respectively. This indicates that for this task only very few features are necessary to achieve outstanding results.

*Meta-comparison:* In a follow-up meta-comparison we compared our achieved results on the financial task with those of some prominent techniques for multi-relational classification problems. We considered here the propositionalization methods DARA [27], RELAGGS [35], and CrossMine [19], as well as the multi-view approaches MVC [43] and MRC [20]. Table III shows the predictive accuracies of each representative. With an accuracy of 97.1% GenPro achieved the highest score; 2% better than the second best approach DARA. Note that the here reported result of the RELAGGS algorithm origins from the original paper and is slightly better than the ones of our implementation (c.f. Table II). This is because not sufficient information about the actual implementation or experimental setup was available to us in order to fully reproduce their results. Note that the entire comparison here underlies the limitations of meta-analysis approaches and thus, the results should be interpreted with caution [44]. Nevertheless, the outstanding performance of our approach still indicates a high potential.

### B. Further benchmark tasks

Table IV shows the empirical results on further benchmark tasks. For reasons of clarity we only report the results of the best RELAGGS and the best GenPro variant, respectively. On four out of five tasks GenPro clearly outperformed RELAGGS.

Method	Accuracy	Source
GenPro	0.971	
DARA	0.951	[45]
RELAGGS	0.941	[35]
MVC	0.941	[43]
MRC	0.934	[20]
CrossMine	0.895	[19]

TABLE III: Meta-comparison of different approaches on the financial task.

Dataset	RELAGGS	GenPro	$\Delta$
Musk small	0.823	0.923	+10.0%
Musk large	0.794	0.846	+5.2%
Hepatitis	0.857	0.906	+4.9%
Mutagenesis	0.900	0.919	+1.9%
Medical	0.903	0.903	$\pm 0.0\%$

TABLE IV: Predictive accuracies on further benchmark tasks. Results indicate the highest accuracies achieved by any RELAGGS or GenPro variant, respectively. The  $\Delta$  column indicates the difference between the results of the two methods.

It achieved predictive accuracies of up to 10% higher than RELAGGS. In the one remaining case, on the Medical dataset, both methods achieved the same result. Surprisingly, even at the very small Hepatitis and Mutagenesis datasets (<60 features), GenPro could score a respectable improvement over RELAGGS which suggests a high degree of versatility of the underlying framework.

## VI. DISCUSSION AND CONCLUSION

In this paper, we have presented a modified propositionalization approach, that overcomes disadvantages of the traditional two-step propositionalization framework. By combining feature construction and feature evaluation we are capable of avoiding exhaustive feature construction and produce only a subset of highly predictive features. Furthermore, we have demonstrated how to use a Genetic Algorithm (GA) to implement the proposed approach. Our empirical results suggest competitive performance as well as great versatility of our approach. In a direct comparison, it outperformed a state-of-the-art propositionalization method on numerous benchmark tasks. Furthermore, it achieved superior results in a meta-comparison with other propositionalization methods and multi-view approaches. Thus, we conclude that this approach represents a considerable extension for any propositionalization technique to improve the predictive performance.

Despite the promising results, our approach has also some limitations which are discussed hereafter. First, we have tested our approach only with a single feature construction strategy. Although it achieved outstanding results, the framework’s performance should be also evaluated with other strategies in order to strengthen the conclusions. Furthermore, all our experiments were performed on relatively small databases making it precarious to make assumptions about the scaling behavior. Compared to exhaustive feature construction, GenPro’s properties as anytime-algorithm support undeniably the ability to handle bigger and more complex databases. Also, the increased computational cost can be reduced to a minimum



through parallel computation and further optimizations [46]. However, to fully evaluate the scalability of our approach experiments on big real-world datasets need to be performed and is therefore, topic of future research.

## REFERENCES

- [1] K.-j. Kim and I. Han, "Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index," *Expert systems with Applications*, vol. 19, no. 2, pp. 125–132, 2000.
- [2] D. Enke and S. Thawornwong, "The use of data mining and neural networks for forecasting stock market returns," *Expert Systems with Applications*, vol. 29, no. 4, pp. 927–940, 2005.
- [3] P.-F. Pai and C.-S. Lin, "A hybrid arima and support vector machines model in stock price forecasting," *Omega*, vol. 33, no. 6, pp. 497–505, 2005.
- [4] C. Apte, B. Liu, E. P. Pednault, and P. Smyth, "Business applications of data mining," *Communications of the ACM*, vol. 45, no. 8, pp. 49–53, 2002.
- [5] J. Bennett, S. Lanning *et al.*, "The netflix prize," in *Proceedings of KDD cup and workshop*, vol. 2007. New York, NY, USA, 2007, p. 35.
- [6] E. Aleskerov, B. Freisleben, and B. Rao, "Cardwatch: A neural network based database mining system for credit card fraud detection," in *Computational Intelligence for Financial Engineering (CIFER), 1997., Proceedings of the IEEE/IAFE 1997*. IEEE, 1997, pp. 220–226.
- [7] P. K. Chan, W. Fan, A. L. Prodromidis, and S. J. Stolfo, "Distributed data mining in credit card fraud detection," *IEEE Intelligent Systems and Their Applications*, vol. 14, no. 6, pp. 67–74, 1999.
- [8] S. Bhattacharyya, S. Jha, K. Tharakunnel, and J. C. Westland, "Data mining for credit card fraud: A comparative study," *Decision Support Systems*, vol. 50, no. 3, pp. 602–613, 2011.
- [9] J. M. Kanter and K. Veeramachaneni, "Deep feature synthesis: Towards automating data science endeavors," in *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*. IEEE, 2015, pp. 1–10.
- [10] H. T. Lam, J.-M. Thiebaut, M. Sinn, B. Chen, T. Mai, and O. Alkan, "One button machine for automating feature engineering in relational databases," *arXiv preprint arXiv:1706.00327*, 2017.
- [11] C. A. Ferreira, J. Gama, and V. S. Costa, "Exploring multi-relational temporal databases with a propositional sequence miner," *Progress in Artificial Intelligence*, vol. 4, no. 1-2, pp. 11–20, 2015.
- [12] A. A. Albrecht, "Stochastic local search for the feature set problem, with applications to microarray data," *Applied Mathematics and Computation*, vol. 183, no. 2, pp. 1148–1164, 2006.
- [13] I. A. Gheyas and L. S. Smith, "Feature subset selection in large dimensionality domains," *Pattern recognition*, vol. 43, no. 1, pp. 5–13, 2010.
- [14] P. Berka *et al.*, "Guide to the financial data set," *PKDD2000 discovery challenge*, 2000.
- [15] S. Kramer, "Relational learning vs. propositionalization: Investigations in inductive logic programming and propositional machine learning," *AI communications*, vol. 13, no. 4, pp. 275–276, 2000.
- [16] S. Džeroski, "Relational data mining," in *Relational Data Mining*, S. Džeroski, Ed. New York, NY, USA: Springer-Verlag New York, Inc., 2000, ch. Relational Data Mining Applications: An Overview, pp. 339–360. [Online]. Available: <http://dl.acm.org/citation.cfm?id=567222.567240>
- [17] R. Alfred and D. Kazakov, "Pattern-based transformation approach to relational domain learning using dynamic aggregation for relational attributes," in *DMIN*, 2006, pp. 118–124.
- [18] H. Blockeel and M. Sebag, "Scalability and efficiency in multi-relational data mining," *ACM SIGKDD Explorations Newsletter*, vol. 5, no. 1, pp. 17–30, 2003.
- [19] X. Yin, J. Han, J. Yang, and S. Y. Philip, "Crossmine: Efficient classification across multiple database relations," in *Constraint-Based mining and inductive databases*. Springer, 2006, pp. 172–195.
- [20] A. Thakkar and Y. Kosta, "Survey of multi relational classification (mrc) approaches & current research challenges in the field of mrc based on multi-view learning," *International Journal of Soft Computing and Engineering (I)*, vol. 247, p. 252, 2012.
- [21] C. Perlich and F. Provost, "Distribution-based aggregation for relational learning with identifier attributes," *Machine Learning*, vol. 62, no. 1-2, pp. 65–105, 2006.
- [22] A. J. Knobbe, M. De Haas, and A. Siebes, "Propositionalisation and aggregates," in *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2001, pp. 277–288.
- [23] H. T. Lam, T. N. Minh, M. Sinn, B. Buesser, and M. Wistuba, "Learning features for relational data," *arXiv preprint arXiv:1801.05372*, 2018.
- [24] V. Gjorgjioski and S. Dzeroski, "Stochastic propositionalization of relational data using aggregates," 2008.
- [25] A. Braud and C. Vrain, "A genetic algorithm for propositionalization," in *International Conference on Inductive Logic Programming*. Springer, 2001, pp. 27–40.
- [26] R. Alfred, "Feature transformation: A genetic-based feature construction method for data summarization," *Computational Intelligence*, vol. 26, no. 3, pp. 337–357, 2010.
- [27] R. Alfred and D. Kazakov, "Data summarization approach to relational domain learning based on frequent pattern to support the development of decision making," in *International Conference on Advanced Data Mining and Applications*. Springer, 2006, pp. 889–898.
- [28] W. Siedlecki and J. Sklansky, "A note on genetic algorithms for large-scale feature selection," in *Handbook Of Pattern Recognition And Computer Vision*. World Scientific, 1993, pp. 88–107.
- [29] J. Yang and V. Honavar, "Feature subset selection using a genetic algorithm," in *Feature extraction, construction and selection*. Springer, 1998, pp. 117–136.
- [30] H. Vafaie and K. De Jong, "Robust feature selection algorithms," in *Tools with Artificial Intelligence, 1993. TAI'93. Proceedings., Fifth International Conference on*. IEEE, 1993, pp. 356–363.
- [31] F. Z. Brill, D. E. Brown, and W. N. Martin, "Fast generic selection of features for neural network classifiers," *IEEE Transactions on Neural Networks*, vol. 3, no. 2, pp. 324–328, 1992.
- [32] M. Mitchell, *An introduction to genetic algorithms*. MIT press, 1998.
- [33] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.
- [34] M. Verleysen and D. François, "The curse of dimensionality in data mining and time series prediction," in *International Work-Conference on Artificial Neural Networks*. Springer, 2005, pp. 758–770.
- [35] M.-A. Krogel and S. Wrobel, "Transformation-based learning using multirelational aggregation," in *International Conference on Inductive Logic Programming*. Springer, 2001, pp. 142–155.
- [36] M.-A. Krogel, "On propositionalization for knowledge discovery in relational databases," Ph.D. dissertation, Otto-von-Guericke-Universität Magdeburg, Universitätsbibliothek, 2005.
- [37] M. Samorani, F. Ahmed, and O. R. Zaiane, "Automatic generation of relational attributes: An application to product returns," in *Big Data (Big Data), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1454–1463.
- [38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.
- [39] R. Frank, F. Moser, and M. Ester, "A method for multi-relational classification using single and multi-feature aggregation functions," in *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2007, pp. 430–437.
- [40] A. K. Debnath, d. C. R. Lopez, G. Debnath, A. J. Shusterman, and C. Hansch, "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity," *Journal of medicinal chemistry*, vol. 34, no. 2, pp. 786–797, 1991.
- [41] M. Jan, S. Tsumoto, and K. Takabayashi, "Medical thrombosis data description,"
- [42] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez, "Solving the multiple instance problem with axis-parallel rectangles," *Artificial intelligence*, vol. 89, no. 1, pp. 31–71, 1997.
- [43] S. Modi, "Relational classification using multiple view approach with voting," *International Journal of Computer Applications*, vol. 70, no. 16, 2013.
- [44] T. D. Spector and S. G. Thompson, "The potential and limitations of meta-analysis," *Journal of Epidemiology and Community Health*, vol. 45, no. 2, p. 89, 1991.
- [45] R. Alfred, *A Data Summarisation Approach to Knowledge Discovery*. Citeseer, 2008.
- [46] E. Cantú-Paz, "A survey of parallel genetic algorithms," *Calculateurs paralleles, reseaux et systems repartis*, vol. 10, no. 2, pp. 141–171, 1998.

# On A Simpler and Faster Derivation of Single Use Reliability Mean and Variance for Model-Based Statistical Testing

Yufeng Xue, Lan Lin, Xin Sun  
Department of Computer Science  
Ball State University  
Robert Bell Building, Room 455  
Muncie, IN 47306, USA  
{yxue2, llin4, xsun6}@bsu.edu

Fengguang Song  
Department of Computer and Information Science  
Indiana University-Purdue University Indianapolis  
723 W. Michigan St., SL 280  
Indianapolis, IN 46202, USA  
fgsong@cs.iupui.edu

## Abstract

*Markov chain usage-based statistical testing has proved sound and effective in providing audit trails of evidence in certifying software-intensive systems. The system end-to-end reliability is derived analytically in closed form, following an arc-based Bayesian model. System reliability is represented by an important statistic called single use reliability, and defined as the probability of a randomly selected use being successful. This paper continues our earlier work on a simpler and faster derivation of the single use reliability mean, and proposes a new derivation of the single use reliability variance by applying a well-known theorem and eliminating the need to compute the second moments of arc failure probabilities. Our new results complete a new analysis that could be shown to be simpler, faster, and more direct while also rendering a more intuitive explanation. Our new theory is illustrated with three simple Markov chain usage models with manual derivations and experimental results.*

## 1 Introduction

This paper re-examines the underlying reliability analysis for *statistical testing* based on a *Markov chain usage model*. This form of statistical testing, developed by the University of Tennessee Software Quality Research Laboratory (UTK SQRL), has been around for more than two decades [5, 4, 8, 6, 9, 12, 11]. With the software use being modeled as a finite-state, discrete parameter, time-homogeneous, and irreducible Markov chain, where the states represent “states of system use” and the arcs represent possible transitions between states of use, the method allows for quantitative certification of software using empirical test data by a statistical protocol. A public domain tool supporting statistical testing called the *JUMBL: J Us-*

*age Model Builder Library*, also developed by UTK SQRL, is freely available [7, 1].

In this paper we focus on the derivation of a system end-to-end reliability estimate, called *single use reliability* (both mean and variance), driven solely by the test data without any mathematical growth assumptions, given the Markov chain usage model, and improve on an earlier analytical solution described in [8]. Our new derivation of the mean was inspired earlier and published in [2], however, it was not until recently that we figured out a new derivation of the variance that is similarly simpler, faster, more direct, and more intuitive, which arrived through a rather convoluted path. The derivation in [8] follows from the definitions of mean and variance and only first principles, which could be a little counter-intuitive to understand. We demonstrate a new derivation and complete a new analysis. Through three examples we show the new derivation agrees with the old derivation (by implementation and experiments), as well as a direct application of the definition. The new theory is fully implemented in the latest version of the JUMBL.

## 2 Single Use Reliability Mean and Variance: The Old Derivation

Current reliability analysis underlying statistical testing follows the arc-based Bayesian model [8, 10]. Here one applies Miller’s Bayesian model [3] to individual arcs of the Markov chain, and compute for each arc a transition reliability (both mean and variance) from a posterior beta distribution. System end-to-end reliability is computed through the *single use reliability* estimate, defined as “the probability of a randomly selected use executing correctly relative to a specification of correct behavior,” [5, 8] either analytically [8] or through simulation [10]. The analytical solution in closed form [8], both faster and more precise than simulation, was implemented in the JUMBL. In this section we

summarize the major steps and results of this derivation.

Let  $P = [p_{ij}]$  be the  $n \times n$  transition matrix of a Markov chain usage model. The  $(i, j)$ -th entry  $p_{ij}$  of  $P$  is the conditional probability of the next state being state  $j$  given the current state being state  $i$ . State 1 is the source. State  $n$  is the sink and the only absorbing state (assuming a reasonable error recovery scheme). Given  $P_{n \times n}$ ,  $Q_{(n-1) \times n}$  denotes the submatrix of  $P$  omitting the last row, and  $\dot{Q}_{(n-1) \times (n-1)}$  denotes the submatrix of  $P$  omitting the last row and the last column.  $\dot{Q}$  is the transition matrix of the Markov chain restricted to the transient states.

Let  $r_{i,j}$  be a random variable for “transition reliability,” that is, the fraction of successful transitions from state  $i$  to state  $j$ . Let  $f_{i,j}$  be another random variable for “transition failure probability,” that is, the fraction of unsuccessful transitions from state  $i$  to state  $j$ . Notice that  $f_{i,j} = 1 - r_{i,j}$ .

With the arc-based Bayesian model [10], each arc (transition) reliability  $r_{i,j}$  has a standard beta distribution  $B(\alpha_{i,j}, \beta_{i,j})$  with two parameters  $\alpha_{i,j}$  (for total successes on transitions from state  $i$  to state  $j$ ) and  $\beta_{i,j}$  (for total failures on transitions from state  $i$  to state  $j$ ), where  $\alpha_{i,j} = a_{i,j} + s_{i,j}$  and  $\beta_{i,j} = b_{i,j} + f_{i,j}$  with  $a_{i,j}$ ,  $s_{i,j}$ ,  $b_{i,j}$ ,  $f_{i,j}$  representing prior successes, observed successes, prior failures, and observed failures, respectively, on transitions from state  $i$  to state  $j$ . In case no prior information is available,  $a_{i,j} = b_{i,j} = 1$ . Each executed test case is mapped to the usage model and each executed step is marked as successful or failing. The observed success and failure counts are summed for each individual arc in the usage model.

From the posterior (beta) distribution  $B(\alpha_{i,j}, \beta_{i,j})$  for  $r_{i,j}$  we may compute the mean and variance of  $r_{i,j}$ :

$$E[r_{i,j}] = \frac{\alpha_{i,j}}{\alpha_{i,j} + \beta_{i,j}} = \frac{a_{i,j} + s_{i,j}}{a_{i,j} + s_{i,j} + b_{i,j} + f_{i,j}}, \quad (1)$$

$$\begin{aligned} Var[r_{i,j}] &= \frac{\alpha_{i,j} \beta_{i,j}}{(\alpha_{i,j} + \beta_{i,j})^2 (\alpha_{i,j} + \beta_{i,j} + 1)} \\ &= \frac{(a_{i,j} + s_{i,j})(b_{i,j} + f_{i,j})}{(a_{i,j} + s_{i,j} + b_{i,j} + f_{i,j})^2 (a_{i,j} + s_{i,j} + b_{i,j} + f_{i,j} + 1)}. \end{aligned} \quad (2)$$

Since  $Var[r_{i,j}] = E[r_{i,j}^2] - E^2[r_{i,j}]$ , we have  $E[r_{i,j}^2] = E^2[r_{i,j}] + Var[r_{i,j}]$ .

Given  $f_{i,j} = 1 - r_{i,j}$ , we can compute the mean and variance of  $f_{i,j}$  as  $E[f_{i,j}] = E[1 - r_{i,j}] = 1 - E[r_{i,j}]$  and  $Var[f_{i,j}] = Var[1 - r_{i,j}] = Var[r_{i,j}]$ . Similarly we have  $E[f_{i,j}^2] = E^2[f_{i,j}] + Var[f_{i,j}]$ .

By our assumption state  $n$  (the sink) is the only absorbing state of the Markov chain. A test case ends when the sink is first encountered, therefore, we are only interested in transitions from any state other than the sink (any transient state). In the matrices defined below ( $A$ ,  $B$ ,  $S$ ,  $F$ ,  $R_1$ ,  $R_2$ ,  $F_1$ , and  $F_2$ ),  $i$  is any integer from 1 to  $n - 1$  inclusive, and  $j$  is any integer from 1 to  $n$  inclusive.

Let  $A = [a_{i,j}]$  and  $B = [b_{i,j}]$  be two matrices of size  $(n - 1) \times n$  whose entries are prior arc successes and fail-

ures, respectively, obtained from prior testing experience. Let  $S = [s_{i,j}]$  and  $F = [f_{i,j}]$  be two matrices of size  $(n - 1) \times n$  whose entries are observed arc successes and failures, respectively, obtained through testing.

Let  $R_1 = [E[r_{i,j}]]$  be an  $(n - 1) \times n$  matrix whose  $(i, j)$ -th entry is the expected arc reliability of going from state  $i$  to state  $j$ , and  $R_2 = [E[r_{i,j}^2]]$  be an  $(n - 1) \times n$  matrix whose  $(i, j)$ -th entry is the expected value of  $r_{i,j}^2$ . Let  $\dot{R}_1$  and  $\dot{R}_2$  denote respectively the submatrices of  $R_1$  and  $R_2$  omitting the last columns.

Similarly we define  $F_1 = [E[f_{i,j}]]$  as an  $(n - 1) \times n$  matrix whose  $(i, j)$ -th entry is the expected arc failure probability of going from state  $i$  to state  $j$ , and  $F_2 = [E[f_{i,j}^2]]$  as an  $(n - 1) \times n$  matrix whose  $(i, j)$ -th entry is the expected value of  $f_{i,j}^2$ .

Given two matrices  $X$  and  $Y$  of the same size (dimension),  $X \otimes Y$  denotes the entry-wise (or component-wise) product of  $X$  and  $Y$ .  $X \otimes Y$  has the same size as  $X$  and  $Y$ .

We define four entry-wise products as follows. Two of them are of size  $(n - 1) \times n$ :  $\mathcal{F}_1 = Q \otimes F_1$  and  $\mathcal{F}_2 = Q \otimes F_2$ . The other two are square matrices of order  $n - 1$ :  $\mathcal{R}_1 = \dot{Q} \otimes \dot{R}_1$  and  $\mathcal{R}_2 = \dot{Q} \otimes \dot{R}_2$ .

Let  $I$  be an  $(n - 1) \times (n - 1)$  identity matrix, and  $U$  be a column vector of ones of size  $n$ . It is established in [8] that  $F^*$  in (3) computes the expected *single use failure probability* (or *single use unreliability*) from any starting state.

$$F^* = (I - \mathcal{R}_1)^{-1} \mathcal{F}_1 U \quad (3)$$

Observe that  $F^*$  is a column vector of size  $n - 1$ . The  $i$ -th component of  $F^*$  is the computed probability of failure (the expected value) for an arbitrary use of the system from a particular usage state, state  $i$ , to the sink ( $i$  runs from 1 to  $n - 1$  inclusive; the starting state could be any transient state).

Therefore, the expected single use reliability of the system (starting from the source) is one minus the first component of  $F^*$  computed by (3).

For an intuitive understanding of (3), consider all the paths in the usage model that originate from state  $i$  and have all but the last step successful; the last step on the path is the only failure step. The probability of taking one of such paths gives the failure probability from state  $i$ , and is computed in three steps. First, it is shown in [8] that  $(I - \mathcal{R}_1)^{-1} = \mathcal{R}_1^0 + \mathcal{R}_1^1 + \mathcal{R}_1^2 + \dots$ , hence the  $(i, j)$ -th entry in the inverse matrix computes the probability of successfully moving from state  $i$  to state  $j$  in any finite and arbitrary number of steps (starting from 0 step). State  $j$  must be transient because only the last failure step could lead to the sink. Second, the inverse matrix is multiplied by the single-step failure matrix  $\mathcal{F}_1$  to give the probability of moving from any transient state to any state in the model with all but the last step successful. Here the last transition is made to either a transient state or the sink. And last, the

product is multiplied by the vector of ones of appropriate size to sum up the probabilities of taking paths with a fixed starting state, all successful prior steps before encountering the last failure step, and an arbitrary ending state. The sum is the failure probability from the particular starting state.

An equation is also given in [8] for computing the variance associated with the single use reliability (or equivalently, the variance associated with the single use failure probability) from any starting state.

$$V^* = (I - \hat{\mathcal{R}}_2)^{-1} \mathcal{F}_2 U + 2(I - \hat{\mathcal{R}}_2)^{-1} (\hat{\mathcal{R}}_1 - \hat{\mathcal{R}}_2) F^* - F^* \otimes F^* \quad (4)$$

In (4)  $I$  is an  $(n - 1) \times (n - 1)$  identity matrix, and  $U$  is a column vector of ones of size  $n$ .  $V^*$  as computed is a column vector of size  $n - 1$ . The  $i$ -th component of  $V^*$  is the computed variance associated with the single use reliability (or with the single use failure probability) starting from state  $i$  ( $i$  runs from 1 to  $n - 1$  inclusive).

Therefore, the single use reliability variance (when starting from the source) is the first component of  $V^*$  computed by (4).

### 3 A Simpler, Faster, and More Intuitive Derivation

In this section we illustrate a new derivation of single use reliability mean and variance that is simpler, faster, and more intuitive than the old derivation. The new derivation of the mean was published in [2], however, back then it was unclear if there existed an alternative and new derivation of the variance that is similarly simple and intuitive. This is the major contribution of this paper. The solution was found through a rather convoluted path. What prompted us to look for an alternative derivation was the observation that the old derivation of the variance follows its definition and first principles, and therefore could be counter-intuitive to understand. The new derivation presented here completes a new analytical solution to compute the system reliability (both mean and variance) based on testing experience observed at the arc level taking into account the usage model structure.

We are able to compute the single use reliability mean (expected value) directly, and not through the single use failure probability (or single use unreliability) as follows.

We define another entry-wise product of size  $(n - 1) \times n$ :  $\mathcal{R}'_1 = Q \otimes R_1$ . Let  $W$  be  $\mathcal{R}'_1$  restricted to the last column.  $W$  is a column vector of size  $n - 1$ .

We define  $R^*$  as follows:

$$R^* = (I - \hat{\mathcal{R}}_1)^{-1} W \quad (5)$$

(5) has an intuitive explanation. As explained above for (3), the  $(i, j)$ -th entry in the inverse matrix computes the probability of successfully moving from the transient state

$i$  to the transient state  $j$  in any finite and arbitrary number of steps (starting from 0 step). When multiplied by the single-step success matrix  $\mathcal{R}_1$  restricted to the last column (i.e.,  $W$ ), the last steps are successful steps leading to the sink, hence  $R^*$  gives the probability of successfully moving from any transient state to the sink in any finite and arbitrary number of steps (starting from 0 step).

Observe that  $R^*$  is a column vector of size  $n - 1$ . The  $i$ -th component of  $R^*$  is the expected single use reliability starting from state  $i$  ( $i$  runs from 1 to  $n - 1$  inclusive).

Therefore, the expected single use reliability of the system (starting from the source) is the first component of  $R^*$  computed by (5).

We propose an alternative way to compute the single use reliability variance. Let  $r$  be a random variable denoting the single use reliability. Let  $p_i$  and  $r_i$  denote the probability and the reliability of the  $i$ -th distinct path starting with the source ending with the sink (representing a distinct arbitrary use), respectively. Note that  $r$  is a discrete random variable that takes the value  $E(r_i)$  with probability  $p_i$ , hence  $E(r) = \sum_i p_i E(r_i)$ . The variance can be computed by  $Var(r) = E(r^2) - E^2(r)$ . The problem boils down to how to compute  $E(r^2)$ .

Note that  $r^2$  is also a discrete random variable that takes the value  $E(r_i^2)$  with probability  $p_i$ , hence  $E(r^2) = \sum_i p_i E(r_i^2)$ . We have shown how to compute  $E(r)$  using (5). With the same Markov chain we are able to compute  $E(r^2)$  similarly. Now the  $(i, j)$ -th arc is associated with a new random variable (i.e.,  $E[r_{i,j}^2]$ ) instead of  $E[r_{i,j}]$ . We can substitute  $R_1$  for  $R_2$  and compute  $E(r^2)$  similarly as follows, with the reasonable assumption that all  $r_{i,j}^2$ s are independent random variables.

We define an entry-wise product of size  $(n - 1) \times n$ :  $\mathcal{R}'_1 = Q \otimes R_2$ . Let  $W'$  be  $\mathcal{R}'_1$  restricted to the last column.  $W'$  is a column vector of size  $n - 1$ .

We define  $R'^*$  as follows:

$$R'^* = (I - \hat{\mathcal{R}}'_1)^{-1} W' \quad (6)$$

We define  $V^*$  as:

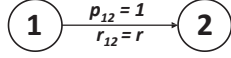
$$V^* = R'^* - R^* \otimes R^* \quad (7)$$

$V^*$  computes the single use reliability variance with each state being the starting state. The  $i$ -th component of  $V^*$  is the single use reliability variance starting from state  $i$  ( $i$  runs from 1 to  $n - 1$  inclusive).

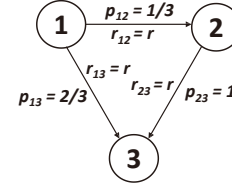
Therefore, the single use reliability variance starting from the source is the first component of  $V^*$  computed by (7).

To sum up, the following steps are needed to compute single use reliability mean and variance by our new derivation:

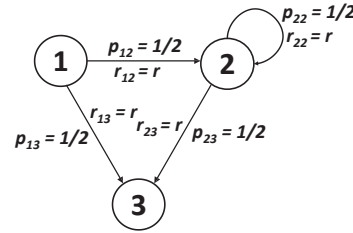
1. Determine  $Q$  and  $\hat{Q}$  from the usage model.
2. Determine  $A$  and  $B$  from prior success and failure counts for each arc in the usage model.



**Figure 1. Example 1 of a Markov chain usage model. Arcs are annotated with transitional probabilities and arc reliabilities.**



**Figure 2. Example 2 of a Markov chain usage model. Arcs are annotated with transitional probabilities and arc reliabilities.**



**Figure 3. Example 3 of a Markov chain usage model. Arcs are annotated with transitional probabilities and arc reliabilities.**

3. Determine  $S$  and  $F$  from observed success and failure counts for each arc in the usage model.
4. Compute  $R_1$  and  $R_2$  from  $A, B, S,$  and  $F$ .
5. Compute  $\mathcal{R}_1$  and  $\dot{\mathcal{R}}_1$ , and  $W$ .
6. Compute  $R^*$  by (5).
7. Compute  $\mathcal{R}'_1$  and  $\dot{\mathcal{R}}'_1$ , and  $W'$ .
8. Compute  $R'^*$  by (6).
9. Compute  $V^*$  by (7).
10. The expected value of the single use reliability is the first component of  $R^*$ .
11. The variance of the single use reliability is the first component of  $V^*$ .

Note that (1) the new derivation of the mean is simpler, faster, and more direct without the need to first compute the single use failure probability; (2) the new derivation of the variance is simpler and faster without the need to compute the second moments of arc failure probabilities; and (3) the new derivation of the variance is simpler, faster, and more intuitive by applying a well-known theorem to compute the variance (i.e.,  $Var(r) = E(r^2) - E^2(r)$ ), and by reusing the existing Markov chain and reusing with adaptation the formula we have derived for  $E(r)$  to compute  $E(r^2)$ . We have implemented the new formulae in the JUMBL under a new analysis engine.

## 4 Examples

In all the three examples (Figures 1 – 3) below the arcs are annotated with transitional probabilities (i.e., the  $p_{ijs}$ ) and arc reliabilities (i.e., the  $r_{ijs}$ ). For simplicity we assume all arc reliabilities have a uniform distribution with the means as  $r_{ijs}$  and the variances as 0s for the derivations in Sections 4.1 – 4.3. SUR is for single use reliability.

### 4.1 Example 1

By the new formula:

To compute the mean:

$$P = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \quad Q = [0 \quad 1] \quad R_1 = [r_{11} \quad r]$$

$$\mathcal{R}_1 = Q \otimes R_1 = [0 \quad r] \quad \dot{\mathcal{R}}_1 = [0] \quad I = [1]$$

$$I - \dot{\mathcal{R}}_1 = [1] \quad (I - \dot{\mathcal{R}}_1)^{-1} = [1] \quad W = [r]$$

$$R^* = (I - \dot{\mathcal{R}}_1)^{-1}W = [r] \quad E(SUR) = r$$

To compute the variance:

$$P = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \quad Q = [0 \quad 1] \quad R_1 = [r_{11} \quad r]$$

$$\mathcal{R}'_1 = Q \otimes R_2 = [0 \quad r^2] \quad \dot{\mathcal{R}}'_1 = [0] \quad I = [1]$$

$$I - \dot{\mathcal{R}}'_1 = [1] \quad (I - \dot{\mathcal{R}}'_1)^{-1} = [1] \quad W' = [r^2]$$

$$R'^* = (I - \dot{\mathcal{R}}'_1)^{-1}W' = [r^2] \quad E(SUR^2) = r^2$$

$$Var(SUR) = E(SUR^2) - (E(SUR))^2 = 0$$

We may also compute the single use reliability mean and variance directly based on its definition, given the Markov chain usage model. To compute the probability of a randomly chosen use (path) being successful, we compute the weighted sum of path reliabilities, with weights being the path probabilities.

By the definition of single use reliability:

To compute the mean:

$$E(SUR) = r * 1 = r$$

To compute the variance:

$$Var(SUR) = (r - r)^2 * 1 = 0$$

## 4.2 Example 2

By the new formula:

To compute the mean:

$$P = \begin{bmatrix} 0 & \frac{1}{3} & \frac{2}{3} \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad Q = \begin{bmatrix} 0 & \frac{1}{3} & \frac{2}{3} \\ 0 & 0 & 1 \end{bmatrix} \quad R_1 = \begin{bmatrix} r_{11} & r & r \\ r_{21} & r_{22} & r \end{bmatrix}$$

$$\mathcal{R}_1 = Q \otimes R_1 = \begin{bmatrix} 0 & \frac{r}{3} & \frac{2r}{3} \\ 0 & 0 & r \end{bmatrix} \quad \dot{\mathcal{R}}_1 = \begin{bmatrix} 0 & \frac{r}{3} \\ 0 & 0 \end{bmatrix} \quad I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$I - \dot{\mathcal{R}}_1 = \begin{bmatrix} 1 & -\frac{r}{3} \\ 0 & 1 \end{bmatrix} \quad (I - \dot{\mathcal{R}}_1)^{-1} = \begin{bmatrix} 1 & \frac{r}{3} \\ 0 & 1 \end{bmatrix} \quad W = \begin{bmatrix} \frac{2r}{3} \\ r \end{bmatrix}$$

$$R^* = (I - \dot{\mathcal{R}}_1)^{-1}W = \begin{bmatrix} \frac{2r+r^2}{3} \\ r \end{bmatrix} \quad E(SUR) = \frac{2r+r^2}{3}$$

To compute the variance:

$$P = \begin{bmatrix} 0 & \frac{1}{3} & \frac{2}{3} \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad Q = \begin{bmatrix} 0 & \frac{1}{3} & \frac{2}{3} \\ 0 & 0 & 1 \end{bmatrix} \quad R_1 = \begin{bmatrix} r_{11} & r & r \\ r_{21} & r_{22} & r \end{bmatrix}$$

$$\mathcal{R}'_1 = Q \otimes R_2 = \begin{bmatrix} 0 & \frac{r^2}{3} & \frac{2r^2}{3} \\ 0 & 0 & r^2 \end{bmatrix} \quad \dot{\mathcal{R}}'_1 = \begin{bmatrix} 0 & \frac{r^2}{3} \\ 0 & 0 \end{bmatrix}$$

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad I - \dot{\mathcal{R}}'_1 = \begin{bmatrix} 1 & -\frac{r^2}{3} \\ 0 & 1 \end{bmatrix}$$

$$(I - \dot{\mathcal{R}}'_1)^{-1} = \begin{bmatrix} 1 & \frac{r^2}{3} \\ 0 & 1 \end{bmatrix} \quad W' = \begin{bmatrix} \frac{2r^2}{3} \\ r^2 \end{bmatrix}$$

$$R'^* = (I - \dot{\mathcal{R}}'_1)^{-1}W' = \begin{bmatrix} \frac{2r^2+r^4}{3} \\ r^2 \end{bmatrix} \quad E(SUR^2) = \frac{2r^2+r^4}{3}$$

$$\text{Var}(SUR) = E(SUR^2) - (E(SUR))^2 = \frac{2r^2+r^4}{3} - \left(\frac{2r+r^2}{3}\right)^2 = \frac{2r^2(r-1)^2}{9}$$

By the definition of single use reliability:

To compute the mean:

$$E(SUR) = r^2 * \frac{1}{3} + r * \frac{2}{3} = \frac{2r+r^2}{3}$$

To compute the variance:

$$\text{Var}(SUR) = (r - \frac{2r+r^2}{3})^2 * \frac{2}{3} + (r^2 - \frac{2r+r^2}{3})^2 * \frac{1}{3} = \frac{r^2(r-1)^2}{9} * \frac{2}{3} + \frac{4r^2(r-1)^2}{9} * \frac{1}{3} = \frac{2r^2(r-1)^2}{9}$$

## 4.3 Example 3

By the new formula:

To compute the mean:

$$P = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 \end{bmatrix} \quad Q = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \quad R_1 = \begin{bmatrix} r_{11} & r & r \\ r_{21} & r & r \end{bmatrix}$$

$$\mathcal{R}_1 = Q \otimes R_1 = \begin{bmatrix} 0 & \frac{r}{2} & \frac{r}{2} \\ 0 & \frac{r}{2} & \frac{r}{2} \end{bmatrix} \quad \dot{\mathcal{R}}_1 = \begin{bmatrix} 0 & \frac{r}{2} \\ 0 & \frac{r}{2} \end{bmatrix} \quad I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$I - \dot{\mathcal{R}}_1 = \begin{bmatrix} 1 & -\frac{r}{2} \\ 0 & 1 - \frac{r}{2} \end{bmatrix}$$

$$(I - \dot{\mathcal{R}}_1)^{-1} = \frac{1}{1-\frac{r}{2}} \begin{bmatrix} 1 - \frac{r}{2} & \frac{r}{2} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & \frac{r}{2-r} \\ 0 & \frac{2}{2-r} \end{bmatrix} \quad W = \begin{bmatrix} \frac{r}{2} \\ \frac{r}{2} \end{bmatrix}$$

$$R^* = (I - \dot{\mathcal{R}}_1)^{-1}W = \begin{bmatrix} \frac{r}{2} + \frac{r^2}{4-2r} \\ \frac{r}{2-r} \end{bmatrix} = \begin{bmatrix} \frac{r}{2-r} \\ \frac{r}{2-r} \end{bmatrix}$$

$$E(SUR) = \frac{r}{2-r}$$

To compute the variance:

$$P = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 \end{bmatrix} \quad Q = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \quad R_1 = \begin{bmatrix} r_{11} & r & r \\ r_{21} & r & r \end{bmatrix}$$

$$\mathcal{R}'_1 = Q \otimes R_2 = \begin{bmatrix} 0 & \frac{r^2}{2} & \frac{r^2}{2} \\ 0 & \frac{r^2}{2} & \frac{r^2}{2} \end{bmatrix} \quad \dot{\mathcal{R}}'_1 = \begin{bmatrix} 0 & \frac{r^2}{2} \\ 0 & \frac{r^2}{2} \end{bmatrix}$$

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad I - \dot{\mathcal{R}}'_1 = \begin{bmatrix} 1 & -\frac{r^2}{2} \\ 0 & 1 - \frac{r^2}{2} \end{bmatrix}$$

$$(I - \dot{\mathcal{R}}'_1)^{-1} = \frac{1}{1-\frac{r^2}{2}} \begin{bmatrix} 1 - \frac{r^2}{2} & \frac{r^2}{2} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & \frac{r^2}{2-r^2} \\ 0 & \frac{2}{2-r^2} \end{bmatrix}$$

$$W' = \begin{bmatrix} \frac{r^2}{2} \\ \frac{r^2}{2} \end{bmatrix}$$

$$R'^* = (I - \dot{\mathcal{R}}'_1)^{-1}W' = \begin{bmatrix} \frac{r^2}{2} + \frac{r^4}{4-2r^2} \\ \frac{r^2}{2-r^2} \end{bmatrix} = \begin{bmatrix} \frac{r^2}{2-r^2} \\ \frac{r^2}{2-r^2} \end{bmatrix}$$

$$E(SUR^2) = \frac{r^2}{2-r^2} \quad \text{Var}(SUR) = E(SUR^2) - (E(SUR))^2 = \frac{r^2}{2-r^2} - \left(\frac{r}{2-r}\right)^2 = \frac{2r^2(r-1)^2}{(2-r^2)(2-r)^2}$$

By the definition of single use reliability:

To compute the mean:

$$E(SUR) = \sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i \cdot r^i \cdot \frac{r}{2} \cdot \frac{r}{2} + \frac{r}{2} = \frac{r^2}{4-2r} + \frac{r}{2} = \frac{r}{2-r}$$

To compute the variance:

$$\begin{aligned} \text{Var}(SUR) &= (r - \frac{r}{2-r})^2 \cdot \frac{1}{2} + \sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot (r^i \cdot r \cdot r - \frac{r}{2-r})^2 \\ &= \frac{r^2}{2} \cdot \frac{(1-r)^2}{(2-r)^2} + \frac{r^2}{4} \cdot \sum_{i=0}^{\infty} (r^2 \cdot \left(\frac{r^2}{2}\right)^i + \frac{1}{(2-r)^2} \cdot \left(\frac{1}{2}\right)^i - \frac{2r}{2-r} \cdot \left(\frac{r}{2}\right)^i) \\ &= \frac{r^2}{2} \cdot \frac{(1-r)^2}{(2-r)^2} + \frac{r^2}{4} \cdot \left(\frac{2r^2}{(2-r)^2} - \frac{4r}{(2-r)^2} + \frac{2}{(2-r)^2}\right) \\ &= \frac{r^2}{2} \cdot \frac{(r^2-4r+2)(2-r)^2 + r^2(2-r)^2}{(2-r)^2(2-r)^2} = \frac{r^2}{2} \cdot \frac{4r^2-8r+4}{(2-r)^2(2-r)^2} \\ &= \frac{2r^2(r-1)^2}{(2-r^2)(2-r)^2} \end{aligned}$$

## 4.4 Experiments

We input the three examples in the JUMBL, and computed the single use reliability (SUR) means and variances using the old analysis as well as our new analysis. The results are summarized in Table 1. For each Markov chain usage model, we carried out the following steps for the experiments:

1. Generate a test suite that consists of minimum coverage test cases that cover every arc and every node of the model. The generated test suite happened to cover each arc exactly once (see Table 1).

**Table 1. Single use reliabilities (means and variances) by the old and the new analyses using the JUMBL for Examples 1 – 3**

	Example 1	Example 2	Example 3
Test Cases	1, 2	1, 3 1, 2, 3	1, 3 1, 2, 2, 3
SUR Mean (Old Derivation)	0.666666667	0.592592593	0.5
SUR Variance (Old Derivation)	$55.5555556E - 3$	$65.5006859E - 3$	$83.3333333E - 3$
SUR Mean (New Derivation)	0.666666667	0.592592593	0.5
SUR Variance (New Derivation)	$55.5555556E - 3$	$65.5006859E - 3$	$83.3333333E - 3$

- Record all tests as successful in the test suite, and run a test case analysis using the old engine to get the single use reliability mean and variance by the old derivation.
- With the same recorded test results run a test case analysis using the new engine to get the single use reliability mean and variance by the new derivation.

For each example, since each arc happened to be covered exactly once in the test suite, we have  $s_{i,j} = 1, f_{ij} = 0$ . Assuming no prior information  $a_{i,j} = b_{i,j} = 1$ . By (1) and (2) each arc reliability has a mean of  $\frac{2}{3}$  and a variance of  $\frac{1}{18}$ . One can easily verify that if we plug in  $r = \frac{2}{3}$  in the formulae we derived above for Examples 1 – 3, we get the same single use reliability means as shown in Table 1 (see the two rows for SUR mean). One can also verify for Example 1 that the single use reliability variance degenerates to the arc reliability variance (as there is only one arc in the path), i.e.,  $\frac{1}{18} = 55.5555556E - 3$ .

We observe that for all the three examples (assuming an arc reliability mean of  $\frac{2}{3}$  and an arc reliability variance of  $\frac{1}{18}$  for every arc), our new derivation produces the same single use reliability mean and variance as the old derivation.

## 5 Conclusion

Statistical testing based on a Markov chain usage model has been well established in theory and proved sound and effective in practice [5, 4, 8, 6, 9, 12, 11], with tools available to support all the stages of testing and to automate the testing process [1, 7]. This paper presents a simpler, faster, more direct, and more intuitive derivation of the single use reliability mean and variance, following the arc-based Bayesian model [8, 10]. With our new theory single use reliability mean is obtained more directly without the need to first compute the single use failure probability. Single use reliability variance is obtained in a faster and simpler way applying a well-known theorem, without the need to compute the second moments of arc failure probabilities. We illustrate our new theory with three small Markov chain

usage models with manual derivations and experimental results.

## Acknowledgements

This work was generously funded by Ontario Systems through the NSF Security and Software Engineering Research Center (S<sup>2</sup>ERC).

## References

- [1] 2018. J Usage Model Builder Library (JUMBL). Software Quality Research Laboratory, The University of Tennessee. <http://jumbl.sourceforge.net/jumblTop.html>.
- [2] L. Lin, Y. Xue, and F. Song. A simpler and more direct derivation of system reliability using markov chain usage models. In *Proceedings of the 29th International Conference on Software Engineering and Knowledge Engineering*, pages 462–466, Pittsburgh, PA, 2017.
- [3] K. W. Miller, L. J. Morell, R. E. Noonan, S. K. Park, D. M. Nicol, B. W. Murrill, and J. M. Voas. Estimating the probability of failure when testing reveals no failures. *IEEE Transactions on Software Engineering*, 18(1):33–43, 1992.
- [4] J. H. Poore. Theory-practice-tools for automated statistical testing. *DoD Software Tech News: Model-Driven Development*, 12(4):20–24, 2010.
- [5] J. H. Poore, L. Lin, R. Eschbach, and T. Bauer. Automated statistical testing for embedded systems. In J. Zander, I. Schieferdecker, and P. J. Mosterman, editors, *Model-Based Testing for Embedded Systems in the Series on Computational Analysis and Synthesis, and Design of Dynamic Systems*. CRC Press-Taylor & Francis, 2011.
- [6] J. H. Poore and C. J. Trammell. Application of statistical science to testing and evaluating software intensive systems. In M. L. Cohen, D. L. Steffey, and J. E. Rolph, editors, *Statistics, Testing, and Defense Acquisition: Background Papers*. National Academies Press, 1999.
- [7] S. J. Prowell. JUMBL: A tool for model-based statistical testing. In *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, page 337c, Big Island, HI, 2003.
- [8] S. J. Prowell and J. H. Poore. Computing system reliability using Markov chain usage models. *Journal of Systems and Software*, 40(4):199–222, 2004.
- [9] S. J. Prowell, C. J. Trammell, R. C. Linger, and J. H. Poore. *Cleanroom Software Engineering: Technology and Process*. Addison-Wesley, Reading, MA, 1999.
- [10] K. Sayre and J. H. Poore. A reliability estimator for model based software testing. In *Proceedings of the 13th International Symposium on Software Reliability Engineering*, pages 53–63, Annapolis, MD, 2002.
- [11] J. A. Whittaker and J. H. Poore. Markov analysis of software specifications. *ACM Transactions on Software Engineering and Methodology*, 2(1):93–106, 1993.
- [12] J. A. Whittaker and M. G. Thomason. A Markov chain model for statistical software testing. *IEEE Transactions on Software Engineering*, 30(10):812–824, 1994.

# A Document-based Parameter Correlation Metric for Test Design

Hiroyuki Nakagawa  
Graduate School of Information  
Science and Technology  
Osaka University  
Osaka, Japan  
Email: nakagawa@ist.osaka-u.ac.jp

Nobukazu Ishii  
Information and Computer Sciences  
School of Engineering Science  
Osaka University  
Osaka, Japan  
Email: n-ishii@ist.osaka-u.ac.jp

Tatsuhiko Tsuchiya  
Graduate School of Information  
Science and Technology  
Osaka University  
Osaka, Japan  
Email: t-tutiya@osaka-u.ac.jp

**Abstract**—Efficient software testing requires precise test space definition. To determine the test space, constraint elicitation is one of the important processes in a test design; however, the process usually requires manual capturing and precise definition of constraints. We have developed a constraint elicitation process that helps to define constraints from documents relevant to the test model. In this paper, we propose a refined metric that finds parameter combinations to be extracted more precisely. This metric determines the parameter correlation on the basis of word co-occurrences in the specification document. We conduct experiments on some test models and demonstrate that our metric allows us to find parameter combinations that form constraints with a high recall rate.

## I. INTRODUCTION

Software testing is an essential activity in the software development process. In order to conduct the activity correctly and efficiently, the testing technique requires precise test space definition to perform testing. We consider a test space that is modeled by a set of parameters, their values, and constraints on the value combinations [1] [2]. The constraints define the value combinations that are prohibited and should be excluded from test cases. Constraint elicitation and handling is crucial in test design [3] [4]; however, the constraint elicitation process has not been well studied.

Our objective is to construct a constraint elicitation process that helps us define constraints. In general, constraint elicitation processes have to solve two problems, that is, “*How do we find which parameter combinations form constraints?*”, and “*How do we find which value combinations on the given parameters define constraints?*”.

We have designed the overview of a constraint elicitation process [5]. To solve the first problem, we proposed a metric for calculating the correlation between parameters, which uses the distances between words in a document of a System Under Test (SUT) to estimate parameter correlations, in the previous work [5]. In this paper, we introduce an enhanced metric to estimate the correlation between parameters more precisely. Since constraints are defined on the relationships between relevant parameters, such a metric helps to find which parameter combinations should be considered to define constraints. We previously proposed a metric for calculating

the correlation between parameters in [5], which sums up the distances between words of two parameters in a specification document of a System Under Test (SUT). We conduct experiments on two real world applications, a web application and a Unix command, to evaluate the validity of our metric. This empirical evaluation indicates that our new metric allows us to find parameter combinations that form constraints from a specification document of SUT with a higher precision and recall rate than the previous metric.

The rest of the paper is organized as follows: Section 2 gives the background of this study by providing the explanation of constraints; Section 3 gives the overview of our approach; Section 4 describes how we find parameter combinations that probably cause constraints using our metric; Section 5 presents the results of two experiments on real world testing examples, and Section 6 explains how we evaluate our approach with the experimental results; Section 7 discusses related work, and Section 8 concludes the paper.

## II. CONSTRAINTS

As an example of a System Under Test (SUT), consider a web application which may be influenced by various factors including operating systems, browsers and memory size. Suppose that the three factors are chosen as test parameters. We list the possible values of parameters in Table I. This test model has three parameters, i.e., OS, Browser, and RAM (memory), with their parameter values. A test case is a vector of parameter values, such as (Windows, Chrome, 4GB).

Formally the test space is modeled by a set of parameters, their values, and constraints on the value combinations. In particular, constraints define the combinations that never happen and must be excluded from test cases. For example, when we choose Mac for the parameter OS, we should choose as the

TABLE I: A cross-browser test model.

Parameter	Values
OS	Windows, Mac, Linux
Browser	IE, Safari, Chrome
RAM	512MB, 1GB, 2GB, 4GB



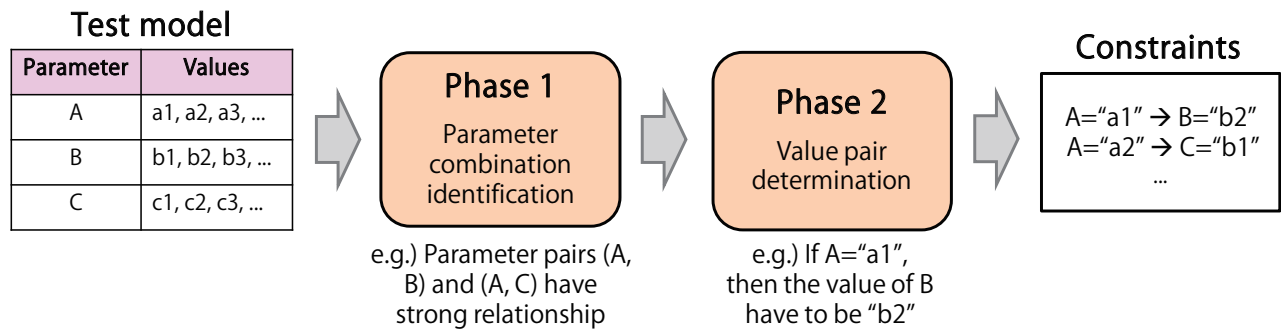


Fig. 1: An overview of the constraint elicitation process. This paper focuses on Phase 1. We have proposed a method [5] for the activity in Phase 2.

browser Safari or Chrome, because Mac OS does not support IE (Internet Explorer). This constraint is defined as follows: OS = "Mac" ⇒ Browser = "Safari" || "Chrome".

In the presence of constraints it is necessary to design a test suite such that all test cases satisfy the constraints. Otherwise, some test cases would not be executable because of constraint violation, resulting in redoing the test design process. A test suite constructed *with* considering constraints is different with one constructed *without* considering constraints because these test spaces are not the same.

### III. CONSTRAINT ELICITATION PROCESS

The objective of our study is to support identifying which combinations of values define constraints. There are two difficulties in the constraint elicitation: one exists in *parameter combination identification* and the other exists in *value combination identification*. In our example explained in Section II, we first have to identify which parameter combinations, such as parameters "OS" and "Browser", form constraints. Next, after identifying parameter combinations that cause constraints, we also have to determine which value combinations cause constraints. In the above example, it corresponds to the identification of the value combinations, such as "Mac" and "Safari" on the parameter combination "OS" and "Browser".

Figure 1 illustrates an overview of our constraint elicitation process. This process consists of the following two steps:

- **Phase 1: Parameter combination identification.** We identify which parameter combinations form constraints. We find such combinations by analyzing a specification document for the SUT. For this analysis, we use a metric that we propose in this paper.
- **Phase 2: Value pair determination.** We determine which value combinations cause constraints.

The goal of this paper is to help find which parameter combinations have strong relationships in Phase 1. Although Phase 2 is beyond the scope of our paper, we have proposed a method of determining value combinations that define constraints using a web search engine [5]. We use hits of search results as a metric. We support that excessively higher/lower hits indicate that the corresponding value pairs are bound/uncommon pairs and therefore they probably define constraints.

### IV. PARAMETER COMBINATION IDENTIFICATION

The objective of Phase 1 is to identify which parameter combinations have strong relationships. Most constraints are caused by strong relationships between parameters. In order to find such strong relationships, we use a specification document as a definitive source and a metric for calculating correlation between parameters. The correlation  $\rho(f, g)$  between parameters  $f$  and  $g$  represents how strong the relationship between two parameters is. The value of the metric is calculated by using a  $diff(f, g)$  value, which represents how differently relevant words of these two parameters appear in the document.

Our parameter combination identification process consists of the following steps. A specification document for the test model is given to the process as an input data. We assume that the document is given as a sequence of English words.

- **Step 1:** select a set of words for each parameter. The members of the set are the parameter name and values of the parameter. If a value of the parameter is a common word, such as "on" and "off", the value is excluded from the set. Instead, representative words that can explain the parameter, such as the full name of the parameter, can also be the members, if they exist. We call this set *word group*.
- **Step 2:** split the given document into multiple small parts (bins). In this paper, we construct bins all of which have the same size, that is, all of the bins contain the same number of words.
- **Step 3:** count the occurrences of words in every word group within each bin.
- **Step 4:** construct relative frequency tables for each word group using the results of Step 3.
- **Step 5:** sum up the difference of bin values between every two word groups. These values provide the *diff* values.
- **Step 6:** output the reciprocals of the *diff* values as *correlation*  $\rho$  values ( $\rho(f, g) \propto 1/diff(f, g)$ ).

We briefly explain our identification process using our cross-browser testing example described in Section II. First, we define word groups (Step 1). The word group is defined as a set whose members are the parameter, the values of the parameter, and relevant words. In our example, the word group of OS contains the words such as "OS" and "Windows". Next,

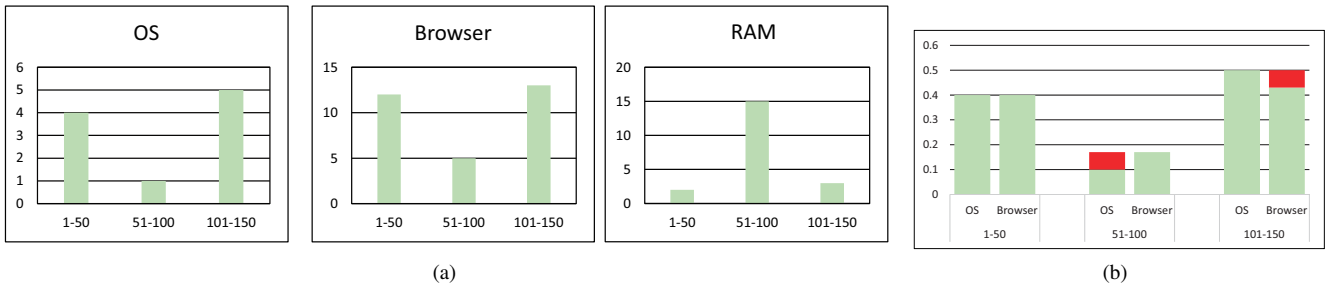


Fig. 2: (a) Histograms for word groups in the cross-browser testing example. (b) The difference of bin values between relative frequency tables for parameters OS and Browser. The sum of red bars corresponds to the  $diff(OS, Browser)$  value.

TABLE II: Frequency and relative frequency tables. The table unifies three frequency tables (denoted by “Freq.”) and three relative frequency tables (“Relative”).

Group	Bin	0-50	51-100	101-150	Total
OS	Freq.	4	1	5	10
	Relative	0.4	0.1	0.5	1
Browser	Freq.	12	5	13	30
	Relative	0.4	0.17	0.43	1
RAM	Freq.	2	15	3	20
	Relative	0.1	0.75	0.15	1

we split the document into small parts (Step 2). We assume that a given document for the system consists of 150 words. In this example, we split it into three parts, i.e., words from the first to the 50th word as the first part, words from the 51st to the 100th word as the second part, and so on. Third, we count the occurrences of the words in every word group within each bin (Step 3). Figure 2(a) illustrates the counting result of the example. After counting the words within each bin for every word group, we construct relative frequency tables (Step 4). Algorithm 1 explains the details of Step 4. Table II lists the results of Step 4 in our example. Next, we sum up the difference of bin values between every pair of two word groups to calculate the  $diff(f, g)$  values (Step 5). Figure 2(b) represents the difference of bin values between relative frequency tables for parameters OS and Browser. We sum up these differences (red bars) and regard the total value as the  $diff$  value. This value becomes low when the occurrences of a word group is similar to those of the other word group. Finally we calculate parameter correlations as the reciprocals of the results of Step 5 (Step 6). Algorithm 2 explains the details of Steps 5 and 6. Table III lists the correlation  $\rho$  values of our example. This result indicates that the parameter combination (OS, Browser) has strong relationship, and therefore, the combination may cause constraints, such as OS = “Mac”  $\Rightarrow$  Browser = “Safari” || “Chrome”.

## V. EXPERIMENTS

In order to evaluate the validity of our metric, we conducted experiments on some applications. We applied our process to two applications: a real world web application (Exp 1) and a Unix command (Exp 2).

**Algorithm 1** Construct relative frequency tables from a document.

```

[Input] doc: given specification document
[Input] wordgroup[][]: word groups
// wordgroup[n]: word group for the n th parameter
[Input] BINS: the number of bins
[Output] relFreqDist[][]: relative frequency tables
// relFreqDist[n]: a relative frequency table for the n th parameter
(word group)
// relFreqDist[n][i]: the relative frequency of i th bin for the n
th parameter (word group)
1: // split given document into a word list
2: words  $\leftarrow$  split(doc);
3:
4: // count frequency
5: freqDist[][]  $\leftarrow$  initialized by 0;
6: currentBin  $\leftarrow$  0;
7: binSize  $\leftarrow$  words.size() / BINS; // the number of words for each
bin
8: for i = 0 to words.size() - 1 do
9:   for j = 0 to wordgroup.size() - 1 do
10:    if wordgroup[j].contains(words[i]) then
11:      freqDist[j][currentBin]++;
12:    end if
13:  end for
14:  if (i + 1) % binSize == 0 then
15:    currentBin++; // move to the next bin
16:  end if
17: end for
18:
19: // construct relative frequency tables
20: for i = 0 to wordgroup.size() - 1 do
21:   for j = 0 to BINS - 1 do
22:     relFreqDist[i][j]
        $\leftarrow$  freqDist[i][j] / sum(freqDist[i])
23:   end for
24: end for

```

### A. Exp 1: Testing of web application

First, we conducted an experiment on a web application testing. The target application is Confluence [6], which is a web application designed for team collaboration. The SUT model for the Confluence testing has seven parameters, i.e., database, server, client, browser, add-ons, and attachment file type. We derived seven word groups corresponding to the parameters for this experiment as illustrated in Table IV. We

TABLE III: Parameter Correlations ( $\rho$  values).

	OS	Browser	RAM
OS	-	7.14	0.77
Browser	-	-	0.86
RAM	-	-	-

**Algorithm 2** Calculate parameter correlation  $\rho(f, g)$  using relative frequency tables.

---

**[Input]**  $f[]$ ,  $g[]$ : relative frequency tables for word groups  $f$  and  $g$ , respectively  
 //  $f[i]$ : relative frequency of  $i$  th bin  
**[Input]**  $BINS$ : the number of bins  
**[Output]** correlation between parameters (word groups)  $f$  and  $g$   
 1: **for**  $i = 0$  to  $BINS-1$  **do**  
 2:    $diff \leftarrow diff + |f[i] - g[i]|$   
 3: **end for**  
 4:  
 5: **return**  $1/diff$ ; //parameter correlation

---

used the document [7] as an input document. We split the document into 960 parts (330 words per bin) based on the number of pages in this document. We determined the bin size to let each bin roughly correspond to a page. In reality, constraints such as the following ones exist in this test model:

- Constraint 1-a: Database = “Microsoft SQL Server”  
 $\Rightarrow$  Server = “Windows Server”.
- Constraint 1-b: Client = “Mac”  $\Rightarrow$  Browser  $\neq$  “IE”.
- Constraint 1-c: Macro = “Multimedia Macro”  
 $\Rightarrow$  File Type = “audio” || “video” || “animation”.

We regarded the following six combinations, (Database, Server), (Browser, Client), (Browser, File type), (Browser, Add-on), (File type, Add-on), and (File Type, Macro), as the combinations to be extracted.

Table V lists the results of the correlation calculation. We extracted the top 30% combinations, whose values are shaded in Table V, from the calculation results. Observing the results of this experiment, although we extracted the combination (Server, Client), which is not involved in any constraints, the metric allowed us to extract most of the correct combinations that should be extracted.

We also observed the correctness of the proposed method. We evaluated the proposed metric by comparing it with the previous metric proposed in our preliminary work [5] as the baseline metric. Briefly explained, our previous study determines parameter correlation using the distance metric, which sums up the distances between words of two parameters in the document.

We use the following definition of the precision and recall:  $Precision = |Correct \cap Extracted|/|Extracted|$ ; and  $Recall = |Correct \cap Extracted|/|Correct|$ , where  $Correct$  is the set of correct pairs to be extracted and  $Extracted$  is the set of pairs that the baseline or proposed method actually extracts from the document. Figure 3 shows the precision and recall rates of the two methods. From the figure, both precision and recall rates of the proposed method are higher than the rates of the baseline method.

## B. Exp 2: Testing of a Unix command

Next, in order to evaluate the scalability of our approach, we applied our metric to a Unix command, `mount` (Exp 2). The SUT model for the testing on mount command has parameters for options, such as `-a`, `-o`, and `-t`, and arguments, such as ones for specifying devices, directories, and volume labels. We constructed 29 word groups corresponding to the parameters for this experiment. We used a manual of `mount` command [8] as an input document<sup>1</sup>. We split the document into 400 parts based on the number of lines in this document (32 words per a bin). In reality, some of constraints exist in the test model as follows:

- Constraint 2-a: “-o” = “ON”  $\Rightarrow$  “-a” = “ON”.
- Constraint 2-b: “-r” = “ON”  $\Rightarrow$  “-w” = “OFF”.
- Constraint 2-c: “-t” = “ON”  $\Rightarrow$  “vfstype”  $\neq$  “NULL”.

In this experiment, we changed the extraction rate (top 5%, 10%, or 15%) and observed each precision and recall rates. From the results listed in Table VII, it was found that we were able to extract most of the parameter combinations to be extracted even if we extracted only top 5% combinations.

## VI. DISCUSSION

We now discuss our correlation metric in the light of our experiments.

The experimental results in Exp 1 and Exp 2 demonstrate that our metric could extract most of the parameter combinations that form constraints. In Exp 1, both precision and recall rates of the proposed method are higher than the rates of our previous metric. While our previous metric evaluate the strength of parameter relationship using the distance between relevant words, the new metric uses co-occurrence of relevant words. If the word frequency is largely different among word groups, the latter metric works more properly than the previous one.

The fact that recall rates in Exp 2 are better than precision rates indicates that our metric extracts parameters to be extracted with low false negative rates. It means that the process using the metric allows developers to find almost all of the parameter combinations to be extracted by continuously relaxing the threshold, i.e., extraction rate in Exp 2. This process is equivalent to acquiring parameter combinations one by one from the parameter combination list sorted by the correlation metric in descending order. We can define the end condition of extraction using the number of a series of wrong extractions, which mean the extracted combinations are not involved any constraints.

Precision, on the other hand, was not high. The main reason is that the strong relationships do not always cause constraints. While this fact decreases the precision rate, we can still improve the precision rate. For example, we can improve the construction of bins. The current process constructs bins by dividing an input document equally so that the bins contain the same number of words. The mapping of bins to

<sup>1</sup>This document can be shown by specifying the keyword “mount” with “8- Maintenance Commands” option.

TABLE IV: Word groups for the Confluence testing (Exp 1). Group names correspond to the parameter names.

Group	Words in word group
Database	database, PostgreSQL, MySQL, Oracle, Microsoft SQL Server, H2
Browser	browser, IE, Internet Explorer, Firefox, Chrome, Safari, Mobile Safari
Server	server, Windows Server, Linux, Unix, Mac
Client	client, Windows, Linux, iOS, Android, Mac
File type	file type, file extension, upload file, image, Office, PDF, video, audio, animation, docx, doc, pptx, ppt, xlsx, xls, txt
Add-on	add-on, plugin, Scroll versions, copy space, scroll PDF Exporter, Gliffy, Lucidchart, Balsamiq
Macro	macro, Multimedia Macro, Space Attachments Macro, Office PowerPoint Macro, Gallery Macro, PDF Macro, View File Macro

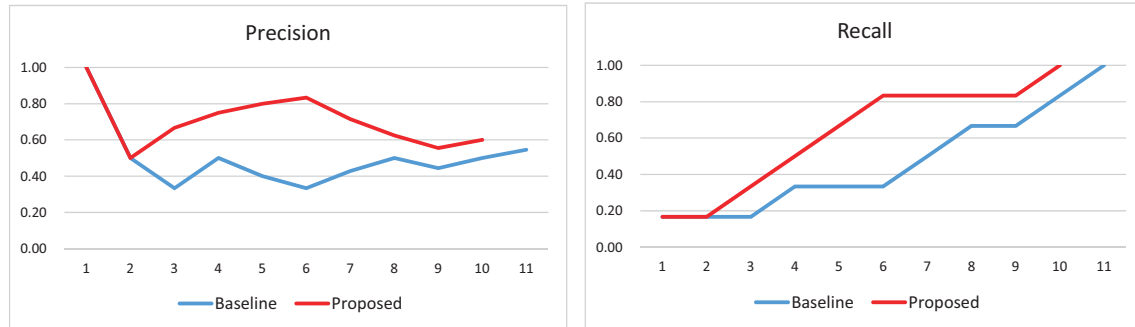


Fig. 3: The transition of precision and recall rates in Exp 1. X axis represents the number of extracted pairs in descending order.

TABLE V: Correlation values in Exp 1. Bold values represent correct combinations to be extracted. Shaded values represents combinations that our method extracted.

	Browser	Server	Client	File type	Add-on	Macro
Database	0.575	<b>0.716</b>	0.568	0.512	0.548	0.508
Browser		0.578	<b>0.608</b>	<b>0.602</b>	<b>0.588</b>	0.525
Server			0.698	0.528	0.58	0.522
Client				0.561	0.572	0.58
File type					<b>0.575</b>	<b>0.603</b>
Add-on						0.567

sentimental blocks, such as paragraphs or sentences, will make the extraction more precise. The main objective of this paper is to elicit constraints; however, we can use the metric for other activities in the test design, such as test suite reduction [9] or the identification of a subset of parameters that require higher coverage strength in variable strength interaction testing [10].

The effectiveness of our parameter combination identification also depends on how we select relative words of parameters in their word groups. As demonstrated in Exp 1 and Exp 2, we added some characteristic words related to the parameters, such as synonyms and full names of options, into word groups. In order to choose more adequate word groups, we need a mechanism of collecting relevant words. We could use a query augmentation method, such as [11], or word2vec [12][13] for this purpose.

## VII. RELATED WORK

Currently there are few studies on the constraint elicitation. Blue et al. [14] presents a test suite reduction method, which excludes test cases that contain prohibited value combinations.

Their method uses existing test cases and minimizes the test suite with covering all combinations that are included in the existing test cases. Their method is useful to define a small set of test cases without violating constraints; however, the main focus is on the extraction of minimized test cases from the existing test cases.

Our approach, including our previous metric [5] explained in Section V, uses word frequencies in a document to identify the correlation between parameters. TF-IDF [15] is also known as a method of reasoning the relationship between words. Gabrilovich et al. [16] proposed a method called *Explicit Semantic Analysis (ESA)*, in which a word is represented as a vector whose attributes represent the relevance of individual concepts calculated using TF-IDF by using Wikipedia as an input document. While these co-occurrence-based approaches require a large number of document, our approach uses only one document. This feature is useful because documents related to the target systems are often limited.

The literature in the requirements engineering field has dealt with linguistic techniques. Falessi et al. [17] evaluate the performance of a large number of natural language processing techniques. They define seven principles for evaluating the performance of these techniques. Some of them could be useful to improve the correlation metric. Query augmentation techniques, such as one in [11], may improve the correctness of extracted parameter combinations by enhancing word groups.

## VIII. CONCLUSIONS

We defined a metric for estimating a parameter correlation for the test design. This metric allows us to identify

TABLE VI: Correlation values in the mount command testing (Exp2). Bold values represent correct combinations to be extracted. Shaded values represent the top 5% combinations that our method extracted.

	-V	-v	-a	-F	...	-r	-w	-L	-U	-t	-O	-o	-B	-R	-M	vfstype	device	dir	uuid	label	num	dirs	opts	
-h	1000	1.5	0.583	0.5	...	0.5	0.5	0.5	0.5	0.528	0.5	0.5	0.5	0.5	0.5	0.5	0.503	0.5	0.5	0.5	0.5	0.5	0.503	
-V		1.5	0.583	0.5	...	0.5	0.5	0.5	0.5	0.528	0.5	0.5	0.5	0.5	0.5	0.5	0.503	0.5	0.5	0.5	0.5	0.5	0.503	
-v			0.583	0.5	...	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.503	0.5	0.5	0.5	0.5	0.5	0.505	
-a				0.538	...	0.5	0.5	0.5	0.5	0.679	1	0.519	0.5	0.5	0.5	0.538	0.524	0.536	0.5	0.5	0.5	0.5	0.526	
-F					...	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.517	0.5	0.5	0.5	0.5	0.5	0.505	
...					...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
-r						0.667	0.5	0.5	0.5	0.5	0.538	0.5	0.5	0.5	0.5	0.5	0.503	0.5	0.5	0.5	0.5	0.5	0.511	
-w							0.5	0.5	0.5	0.5	0.519	0.5	0.5	0.5	0.5	0.5	0.51	0.5	0.5	0.5	0.5	0.5	0.509	
-L								0.5	0.5	0.519	0.5	0.5	0.5	0.5	0.5	0.5	0.51	0.5	0.917	0.667	0.5	0.5	0.503	
-U									1000	0.5	0.5	0.519	0.5	0.5	0.5	0.5	0.51	0.5	0.917	0.667	0.5	0.5	0.503	
-t										0.633	0.583	0.5	0.5	0.5	0.679	0.547	0.594	0.5	0.5	0.5	0.5	0.5	0.537	
-O											0.519	0.5	0.5	0.5	0.583	0.51	0.536	0.5	0.5	0.5	0.5	0.5	0.517	
-o												0.538	0.538	0.5	0.538	0.556	0.605	0.519	0.547	0.5	0.5	0.56	0.57	
-B													0.625	0.5	0.503	0.5	0.5	0.5	0.5	0.5	0.5	0.857	0.509	
-R														0.833	0.5	0.503	0.5	0.5	0.5	0.5	0.5	0.706	0.507	
-M															0.5	0.503	0.5	0.5	0.5	0.5	0.6	0.501		
vfstype																0.517	0.577	0.5	0.5	0.5	0.5	0.5	0.509	
device																	0.6	0.524	0.547	0.503	0.503	0.503	0.567	
dir																		0.55	0.526	0.5	0.5	0.5	0.519	
uuid																			0.909	0.5	0.5	0.5	0.507	
label																				0.5	0.5	0.5	0.524	
num																					0.5	0.5	0.501	
dirs																						0.5	0.509	

TABLE VII: Experimental results in Exp 2.

Extraction rate	Top 5%	Top 10%	Top 15%
Extracted combinations	23	44	69
Precision	0.55	0.23	0.15
Recall	0.73	0.91	1.00

parameter combinations that probably cause constraints. Our experimental results demonstrated that our metric helps us extract valid parameter combinations when we analyze specification documents for the SUT. Such parameter combination extraction also helps other activities in the test design, such as test suite reduction or the identification of a subset of parameters that require higher coverage strength in variable strength interaction testing.

The results presented in this paper indicate some possible directions of further work and improvements. There are two major directions for improving the elicitation mechanism to identify more precise parameter combinations. First one is to develop an additional mechanism to improve the precision rate. The second direction is to define guidelines for applying the metric, which include the criteria for defining thresholds and word groups.

#### ACKNOWLEDGMENTS

This work was supported by JSPS Grants-in-Aid for Scientific Research (Grant Numbers 15K00097, 15K00098).

#### REFERENCES

- [1] S. R. Dalal, A. Jain, N. Karunanithi, J. M. Leaton, C. M. Lott, G. C. Patton, and B. M. Horowitz, "Model-based testing in practice," in *Proc. of the 21st International Conference on Software Engineering (ICSE '99)*, ser. ICSE '99. ACM, 1999, pp. 285–294.
- [2] C. Nie and H. Leung, "A survey of combinatorial testing," *ACM Computing Surveys (CSUR)*, vol. 43, no. 2, pp. 11:1–11:29, Feb. 2011.
- [3] M. Grindal, J. Offutt, and J. Mellin, "Managing conflicts when using combination strategies to test software," in *Proc. of the 18th Australian Software Engineering Conference 2007 (ASWEC'07)*, April 2007, pp. 255–264.
- [4] M. B. Cohen, M. B. Dwyer, and J. Shi, "Constructing interaction test suites for highly-configurable systems in the presence of constraints: A greedy approach," *IEEE Transactions on Software Engineering*, vol. 34, no. 5, pp. 633–650, Sep. 2008.
- [5] H. Nakagawa and T. Tsuchiya, "A search-based constraint elicitation in test design," *IEICE Transactions on Information and Systems*, vol. E99-D, no. 9, pp. 2229–2238, Sep. 2016. [Online]. Available: <https://doi.org/10.1587/transinf.2015KBP0010>
- [6] Atlassian, "Confluence," <https://www.atlassian.com/software/confluence/>.
- [7] —, "Documentation for confluence 5.9," <https://confluence.atlassian.com/all/doc/confluence-documentation-directory-12877996.html>.
- [8] T. F. Foundation, "FreeBSD Man Pages," <https://www.freebsd.org/cgi/man.cgi>.
- [9] P. J. Schroeder and B. Korel, "Black-box test reduction using input-output analysis," in *Proc. of the 2000 ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA '00)*. ACM, 2000, pp. 173–177.
- [10] M. Cohen, P. Gibbons, W. Mugridge, C. Colbourn, and J. Collofello, "A variable strength interaction testing of components," in *Proc. of the 27th Annual International Computer Software and Applications Conference (COMPSAC 2003)*, Nov 2003, pp. 413–418.
- [11] M. Gibiec, A. Czauderna, and J. Cleland-Huang, "Towards mining replacement queries for hard-to-retrieve traces," in *Proc. of the IEEE/ACM International Conference on Automated Software Engineering (ASE'10)*. ACM, 2010, pp. 245–254.
- [12] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. of the 26th International Conference on Neural Information Processing Systems (NIPS'13) Volume 2*. Curran Associates Inc., 2013, pp. 3111–3119.
- [13] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, 2013, <https://arxiv.org/abs/1301.3781>.
- [14] D. Blue, I. Segall, R. Tzoref-Brill, and A. Zlotnick, "Interaction-based test-suite minimization," in *Proc. of the 2013 International Conference on Software Engineering (ICSE 2013)*. IEEE Press, 2013, pp. 182–191.
- [15] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., 1983.
- [16] E. Gabrilovich and S. Markovitch, "Computing semantic relatedness using wikipedia-based explicit semantic analysis," in *Proc. of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*. Morgan Kaufmann Publishers Inc., 2007, pp. 1606–1611.
- [17] D. Falessi, G. Cantone, and G. Canfora, "Empirical principles and an industrial case study in retrieving equivalent requirements via natural language processing techniques," *IEEE Transactions on Software Engineering*, vol. 39, no. 1, pp. 18–44, Jan. 2013.

---

# Improving Integration Testing of Web Service by Propagating Symbolic Constraint Test Artifacts Spanning Multiple Software Projects

Andreas Fuchs <sup>♦</sup>

Vincent von Hof <sup>♦</sup>

<sup>♦</sup> University of Münster

## Abstract

*Software testing is a process involving multiple consecutive testing phases. Once software-units and -components have been tested, the integration of software-components itself must be tested. This discipline of Integration Testing involves multiple systems and resources (human and hardware) and the effort for successfully implementing an integration testing scheme is easily underestimated. Yet errors detected in this phase of the testing process, may require fixes in the individual software components. Trivially, if many such problems appear, the testing process is delayed and in the worst case, the roll-out of the system has to be rescheduled. Focusing on the widely adopted Java Enterprise Edition platform for web-services, this paper presents an approach to move the detection of possible integration errors from the integration testing phase to the earlier and simpler unit testing phase, by generating and collecting constraints relating to web-service calls on the client side, in order to improve the overall time required for the testing process.*

## 1 Introduction

Recently, it was demonstrated how to automatically generate unit tests for the business logic of Java applications that include web services [11]. More specifically, a minimal set of test cases is generated which cover the control-flow graph of that application, e.g. for the **Checkout Counter** employing a **Library Service** as depicted in Figure 1. If a path depends on the response of a web-service request, an executable web service with the desired behavior is generated as well. To achieve this, the application is executed **symbolically** in order to systematically explore all paths through the application. During this execution, a system of constraints with symbolic variables for both the application’s input is created, as well as the web service responses. Once the symbolic execution of one path is

finished, a constraint solver is used to generate concrete values for every symbolic variable constellation. In the end, those values guide the generation of executable test cases and a set of web services which correspond to the expected behavior. Thus, the generated service acts as a **stub**, which always returns the same result regardless of change in input, or as a **mock**, where the return value is based on method-internal logic.

It is important to note, that the web services generated by this approach only mimics the model based on the *client’s* knowledge of the server, since the server source code is a black box to the client during generation time. While this has its own merit, integration testing is still required to ascertain flawless component integration. In software engineering systems are often designed in a waterfall model approach. After writing the code, unit tests are written (or generated and validated). Afterwards the individual software components are tested regarding their *integration* with one another. Since more than two components may exist in a system, the integration between each and every one of them needs to be tested. This may occur for the individual component pairs, larger sets or in a ”big bang” approach. For large systems with several integrated components, there may exist a large amount of these tests. Larger software companies, e.g. Google LLC, restrict themselves to only write integration tests for ”large, high priority“ features [21]. Integration tests are often run during off-hours of development. Problems that occur, can only be fixed the next day. If problems relate to two interconnected components that both require correction, a problem with the fixes is only detected upon the next integration testrun—another round of fixes may be required.

We propose a system, that utilized the constraints generated during the web service automated test case generation phase from one (*client*) component, during the test case generation phase of a (*server*) component. The resulting *unit* test case suite is then to be inter-

preted as follow: If a unit test case of this specific test suite fails upon execution of the unit testing phase, it hints at a problem that might occur **later** during integration testing. By moving the detection time of potential problems from the integration- to the unit-testing-phase, problems that may occur due to changes that were made to the server component are uncovered earlier. Then, testers may run a small subset of the integration test only involving these involved components, thus reducing the total amount of time required to remedy the problem.

To summarize, this paper provides the following contributions:

- The detection of integration errors is *shifted to earlier* testing phases.
- A system for reusing and sharing of constraint information is presented.
- We utilize the constraint sharing system to propagate constraints to the automatic test case generation tool and generate a distinct integration test suite for the web-service software component.

The rest of the paper is organized as follows. Section 2 gives an overview of the phases of software testing and specifically about the relation. In Section 3 we describe how the generated constraints can be gathered and propose an infrastructure schema for exchanging constraint data. In Section 4, we present related work that is comparable to our approach, and we conclude this paper in Section 5.

## 2 Integration Testing in the Development Process

The V-Model is an abstract representation of the process used for software development [9]. It can be considered an extension of the waterfall model [9]. In Figure 2, the left hand side of the model represents steps necessary to define the goal software system. Based upon the requirement specification, the function and non-functional specification are developed. This model omits, that the two lowest phases occur a number of times for a given requirement specification, depending on how many systems are involved.

Afterwards, a technical specification is defined, and the individual programs are specified. After the actual development of the software artifacts, they are

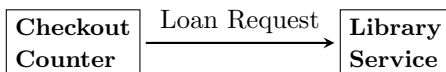


Figure 1: An overview of two systems that exchange data.

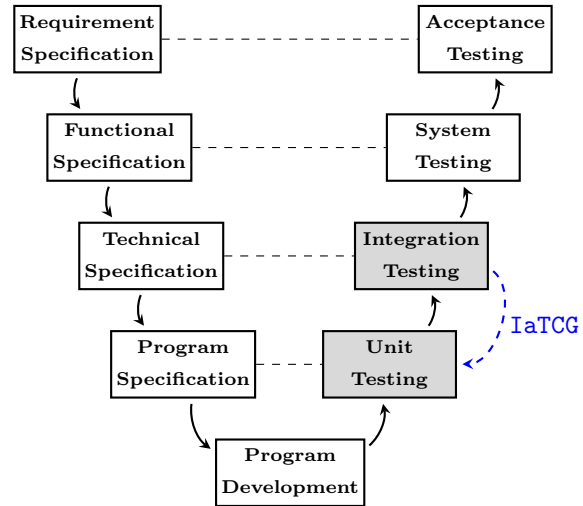


Figure 2: A Software Development V-Model with Integration aware Test Case Generation (IaTCG).

```

1
2 public class CheckoutCounter {
3     private final long maxBalance = 50L;
4
5     private Loan createLoan(Integer userId, List<
6         String> books) {
7         User user = library.getUser(userId);
8         long balance = library.getBalance(user);
9         if (balance <= maxBalance) {
10            Loan approvedLoan = library.requestLoan(
11                user, books);
12            return approvedLoan;
13        }
14        throw new InsufficientFundsException();
15        return null;
16    }
17    ...
18 }

```

Figure 3: Checkout Counter Client.

*unit tested* to ascertain whether or not they meet the program testing goals. At this step, automated test case generation may be utilized to develop the unit test cases. If the individual programs adhere to the goals, the *integration* of the programs according to the technical specification is checked. The goal of the paper, relates to this and the aforementioned step *unit-testing*.

Integration testing of the individual software components may occur in a *top-down*, *bottom-up* or *big-bang* fashion, among others [5]. Assuming, that there is some hierarchy in the way modules are integrated, we may call modules, that require none of the other modules to offer their functionality as basic. For bottom-up and for top-down testing modules will be assigned to groups. In the case of bottom-up integration testing, basic modules make up level 0. Level 1 makes up groups of modules, that require some of the level

```

1
2 public class LibraryService {
3     private final long maxBalance = 50L;
4
5     @WebMethod
6     public Loan requestLoan(User user , List<
7         String> books) {
8         long balance = getBalance(user);
9         if (balance >= maxBalance) {
10            return null;
11        }
12        Loan loan = createLoan(user , books);
13        return loan.getId();
14    }
15    ...
16 }

```

Figure 4: Initial Library Service Server.

0 modules to be present upon execution. This holds for all other levels  $i + 1$ ,  $i \in N$ . Abiding by this process, the modules and their dependencies are executed concretely during this stage.

Using the top-down approach, the lower level dependencies are not present upon integration testing. The functionality of the lower-level dependencies is provided by a mock service instead, that mimic the behavior of the lower-level dependencies. This is also a viable approach, as the mocks can be designed in such a way, that they are very close in behavior to the functionality checked by test cases used in the lower-level dependencies interfaces. A mismatch of mock behavior and lower-level interface test behavior is problematic. Of course, bottom-up and top-down approach can be combined into a mix to decrease integration testing time, depending on the tree structure of the dependency hierarchy tree.

Finally, integration testing can occur by starting all modules at the same time and testing the modules in arbitrary order or even at once. However, first, depending on the amount of modules, it may be problematic to create a runtime environment as that matches the hardware of the production system. Second, if the tests are not run at the same time, the integration test will take a large amount of time, compared to the other approaches. Third, if the tests are run at the same time, it may be hard to determine which module in the module call tree failed. This can be remedied, if error handling and propagation are well designed and allow for tracing by bubbled errors messages. Side effects of running all modules of the system at once may occur that influence the test. Coincidentally, the detection of occurrences of these kinds of problems, is one reason why a big-bang test may be employed in complement with bottom-up, top-down or mixed strategies.

Next, system testing involves testing concrete user workflows, e.g. from user creation, login, and book

```

1
2 public class LibraryServiceV2 {
3
4     @WebMethod
5     public Loan requestLoan(User user , List<
6         String> books){
7         long maxBalance = getMaxBalance(user);
8         long balance = getBalance(user);
9         if (balance >= maxBalance) {
10            return null;
11        }
12        Loan loan = createLoan(user , books);
13        return loan.getId();
14    }
15    ...
16 }

```

Figure 5: Second Version of Library Service Server.

lending in our library example illustrated in Figure 1.

Finally, the process concludes with the acceptance testing phase. Initially, based upon the initial requirements specification, project acceptance criteria were devised, that specify, what parts of the requirements need to be fulfilled, so that the software component is accepted as complete. Furthermore, to allow for some room for error in software development, it specifies how many high level, medium level and low level test failures are acceptable for the module to still past the acceptance test.

Considering the example in Figure 34, Line 2 in CHECKOUTCOUNTER and Line 2 in LIBRARYSERVICE specify a maximum balance in unpaid fees that a library may accumulate, before he is denied a book loan. If the implementation of LIBRARYSERVICE changes as illustrated in Figure 5, where the maximum balance is now an attribute defined on a per-user basis, the CHECKOUTCOUNTER would potentially not throw the desired INSUFFICIENTFUNDS EXCEPTION to display a message to the counter clerk.

### 3 Integration Aware Test Case Generation

This section, will describe the approach, constraint propagation system and test case generation of the proposed new JUnit test suites for automatically creating integration checker tests that take the real behavior of multiple modules into account. The scheme proposed here, is depicted in Figure 6. (1) The modified symbolic execution system takes as input the client under test (CUT). For each symbolically executed path  $\pi$  through the CUT, the system generates a set  $\mathcal{C}_\pi$  of constraints. (2) Next, the constraints are filtered into the set  $\mathcal{C}_I \subseteq \mathcal{C}_\pi$ , This represents only the constraints relating to variables involved in web services calls. Currently, we use a file that stores these constraints (*constraint store*).



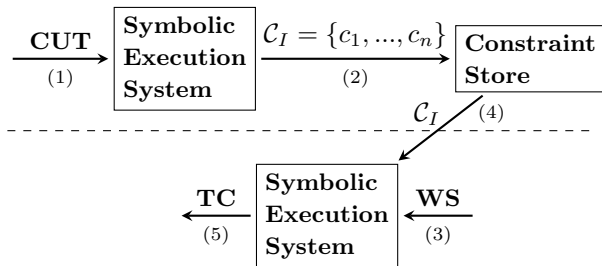


Figure 6: An overview of the constraint propagation and test case generation approach.

Additionally, a unique service identifier name  $\mathcal{N}$  that represents the called component is calculated. It contains the following information.

- I) The domain or the network address of the called web service,
- II) the number and type of parameters of the called method and
- III) the return type of the called method.

Finally,  $\mathcal{C}_I$  and  $\mathcal{N}$  are sent to the constraint store.

In a second phase, we use the *constraint store* and the service under test (including the web service) to generate a test case (TC) for  $\pi$  through the CUT.

**Using stored constraints for integration checker test suite generation.** (3) The second part of the schema occurs, once the modified symbolic execution system is started in server side generation mode. The result of the server mode execution depend on the constraint store. The modified symbolic execution system generates the service ID object for the method under test, by examining the annotations present prior to the method declaration. The constraint store is queried for the constraint set for this service ID. (4) If the constraint store offers an empty set as result, the computation is finished. Otherwise, the test case generator receives the result set  $\mathcal{C}_I$  and starts in **normal** execution mode, i.e. without taking the received constraints into account. Once this step is completed, the test case generator starts in **additional** execution mode. The gathered constraints  $\mathcal{C}_I$  relate to the parameters and return variable of the method under test. Whenever the symbolic execution passes a choice-point, we conjunct the constraints  $\mathcal{C}_I$  to the constraint stack and check the constraint solver for satisfiability as usual. This continues, until the choice-point stack is empty and all node and edges of the control-flow graph have been covered. If a branch, that was previously unreachable in normal mode, becomes reachable in additional mode, this hints at a possible integration error. (5) If a branch becomes unreachable, we keep the normal branch, but generate a failing test case. The test cases generated for these

```

1 public LibraryServiceTest {
2   @Test
3   public void test () {
4     User peter = new User ();
5     Account peterBilling = new Account ();
6     peterBilling.setUser (peter);
7     peterBilling.setBalance (51L);
8     peterBilling.setLimit (52L);
9
10    LibraryServiceV2 ws = new LibraryServiceV2 ()
11      ;
12    assertEquals (null, ws.requestLoan (peter ,
13      null));
14  }
15  @Test
16  @Category (PotentialIntegrationFault.class)
17  public void testIA () {
18    User peter = new User ();
19    Account peterBilling = new Account ();
20    peterBilling.setUser (peter);
21    peterBilling.setBalance (51L);
22    peterBilling.setLimit (52L);
23
24    LibraryServiceV2 ws = new LibraryServiceV2 ()
25      ;
26    assertEquals (null, ws.requestLoan (peter ,
27      null));
28    fail ("The additional execution revealed,
29      that this branch became unreachable");
30  }
31 }

```

Figure 7: Normal  $WS'$  Test Case and additional (integration) Web Service  $WS'$  Test Case.

branches, are saved into a dedicated test suite, marked with `POTENTIALINTEGRATIONFAULT`.

**Making use of the dedicated test suite.** The new additional automated generate test suite is used as follows. Instead of creating a component stub or mock for top-down and mixed-mode integration testing or including the whole lower-level component in the component currently under integration testing, neither is necessary to reap benefits of the new approach. Instead, the dedicated test suite is executed against the unmodified program, if information regarding the compatibility of component external use of the method is required. Should one of the dedicated test suites fail, this does not necessarily point to an error in the server code. However, it does indicate, that the client that was responsible for the creation of the relevant constraints, will fail if it runs into the parameter scenario described in the test case. If, e.g., the failure of the dedicated test suite occurs after the server code was changed, then if this change is deemed to be correct, then the corresponding client should be informed, that an impending change to the server will certainly impede the client operation. A secondary benefit of this approach is, that given the case of service usage across organizational borders, clients may voluntarily offer their server usage constraints to the constraint store, to

possibly guide development of the server component or at least be notified of impending compatibility breaking changes on the server side. Consider the simplified example: Figure 4 is the actual implementation on the server side. The developers introduce a change to the service which results in Figure 5. The client is unaware of these changes and does not adapt the implementation accordingly. However, if the constraints regarding the service usage are stored in the constraint store, upon generating test cases for the server project, the problematic usage of the client can be exposed. Figure 7 depicts the results of both the **normal** execution in Lines 1-12 and the additional generation step in Lines 14-26. The **additional** generation procedure test case will fail upon execution, giving the hint to the server, that there is a problem with the way the service is utilized.

## 4 Related Work

Automated test data and test case generation has been the subject of an extensive research effort. As a result, several techniques and tools have been proposed. *Search-based* approaches [13] use optimization algorithms to identify (near) optimal solutions. This approach can be applied to software testing [3, 6, 16, 18] by optimizing testing criteria. EVOSUITE [10] is an automated search-based unit test generation approach.

Our approach is based upon *symbolic execution*. EFFIGY [14] is one of the earliest systems that uses symbolic execution to generate test cases for programs. Recent approaches [7, 12, 19, 20] use symbolic (or *concolic* as a combination of symbolic and concrete) execution to generate test cases for more complex programs. MUGGL [17] is a symbolic-execution tool for automated unit test generation.

Based on this, several techniques exist, that aim to test web service, e.g. by generating data for requests that invoke specific operations of the service [2, 4, 22, 8, 15] and generate unit tests based upon the source code [11, 1]. However, these techniques take only either the client-, or only the server-side into consideration and are unaware of integration implications that can be observed if both sides are taken into account. To the best of our knowledge, no other approaches have used the constraints generated during symbolic execution of web service clients and reused them to generate a dedicated test suite for the purpose of detecting potential integration test failure ahead of the integration test phase.

## 5 Conclusion

Integration testing requires significant amount of time and resources to complete, which is why some

larger companies restrict the use of integration testing to high profile features [21]. To answer this, we have presented an approach to move the detection of potential errors regarding the integration of service from the integration phase to earlier phases, by enabling the detection of *potential* problems during test case generation for single component (mock-less) unit testing.

The approach extends an existing symbolic execution automated test case generation tool, capable of generating web service mocks and stubs, with the ability of reusing constraints as test artifacts for future test case generation runs. This allows us to detect if service consuming clients—even from other organizations—may be affected by changes in the source code of the service providing classes, without doing integration testing.

Due to the prototypical nature of this work, the constraint store only stores its artifacts directly as objects, so a more robust architecture for storing constraints would be preferable. Currently, the work focused on web services, as they are a popular means of integrating components across organizational boundaries, but it can be easily applied to other modes of interaction, e.g. message oriented middleware.

## References

- [1] A. Arcuri. RESTful API Automated Test Case Generation. In *Software Quality, Reliability and Security (QRS), 2017 IEEE International Conference on*, pages 9–20. IEEE, 2017.
- [2] X. Bai, W. Dong, W.-T. Tsai, and Y. Chen. WSDL-based Automatic Test Case Generation for Web Services Testing. In *Service-Oriented System Engineering, 2005. SOSE 2005. IEEE International Workshop*, pages 207–212. IEEE, 2005.
- [3] A. Baresel, D. Binkley, M. Harman, and B. Korel. Evolutionary Testing in the Presence of Loop-assigned Flags: A Testability Transformation Approach. In *ACM SIGSOFT Software Engineering Notes*, volume 29, pages 108–118. ACM, 2004.
- [4] C. Bartolini, A. Bertolino, E. Marchetti, and A. Polini. WS-TAXI: A WSDL-based Testing Tool for Web Services. In *Software Testing Verification and Validation, 2009. ICST'09. International Conference on*, pages 326–335. IEEE, 2009.
- [5] R. V. Binder. *Testing object-oriented systems: models, patterns, and tools*. Addison-Wesley Professional, 2000.
- [6] L. C. Briand, Y. Labiche, and M. Shousha. Stress testing real-time systems with genetic algorithms.

- In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 1021–1028. ACM, 2005.
- [7] C. Cadar, D. Dunbar, D. R. Engler, et al. KLEE: Unassisted and Automatic Generation of High-Coverage Tests for Complex Systems Programs. In *OSDI*, volume 8, pages 209–224, 2008.
- [8] T. Fertig and P. Braun. Model-driven Testing of RESTful APIs. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1497–1502. ACM, 2015.
- [9] K. Forsberg and H. Mooz. The Relationship of System Engineering to the Project Cycle. In *INCOSE International Symposium*, volume 1, pages 57–65. Wiley Online Library, 1991.
- [10] G. Fraser and A. Arcuri. Evosuite: automatic test suite generation for object-oriented software. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, pages 416–419. ACM, 2011.
- [11] A. Fuchs and H. Kuchen. Test-case generation for web-service clients. In *Proceedings of the The 33rd ACM/SIGAPP Symposium On Applied Computing*, Pau, France, 2018. Publication status: Accepted.
- [12] P. Godefroid, N. Klarlund, and K. Sen. DART: Directed Automated Random Testing. In *ACM Sigplan Notices*, volume 40, pages 213–223. ACM, 2005.
- [13] M. Harman and B. F. Jones. Search-based Software Engineering. *Information and Software Technology*, 43(14):833–839, 2001.
- [14] J. C. King. Symbolic Execution and Program Testing. *Communications of the ACM*, 19(7):385–394, 1976.
- [15] P. Lamela Seijas, H. Li, and S. Thompson. Towards Property-Based Testing of RESTful Web Services. In *Proceedings of the twelfth ACM SIGPLAN workshop on Erlang*, pages 77–78. ACM, 2013.
- [16] Z. Li, M. Harman, and R. M. Hierons. Search Algorithms for Regression Test Case Prioritization. *IEEE Transactions on software engineering*, 33(4), 2007.
- [17] T. A. Majchrzak and H. Kuchen. Automated test case generation based on coverage analysis. In *Theoretical Aspects of Software Engineering, 2009. TASE 2009. Third IEEE International Symposium on*, pages 259–266. IEEE, 2009.
- [18] P. McMinn, M. Harman, D. Binkley, and P. Tonella. The Species per Path Approach to Search-Based Test Data Generation. In *Proceedings of the 2006 international symposium on Software testing and analysis*, pages 13–24. ACM, 2006.
- [19] C. S. Păsăreanu and N. Rungta. Symbolic PathFinder: Symbolic Execution of Java Bytecode. In *Proceedings of the IEEE/ACM international conference on Automated software engineering*, pages 179–180. ACM, 2010.
- [20] K. Sen, D. Marinov, and G. Agha. Cute: a concolic unit testing engine for c. In *ACM SIGSOFT Software Engineering Notes*, volume 30, pages 263–272. ACM, 2005.
- [21] M. Wacker. Just say no to more end-to-end tests. <http://googletesting.blogspot.co.uk/2015/04/just-say-no-to-more-end-to-end-tests.html>.
- [22] W. Xu, J. Offutt, and J. Luo. Testing Web Services by XML Perturbation. In *Software Reliability Engineering, 2005. ISSRE 2005. 16th IEEE International Symposium on*, pages 10–pp. IEEE, 2005.

# Prioritizing Unit Testing Effort Using Software Metrics and Machine Learning Classifiers

Fadel TOURE

Department of Mathematics and Computer Science,  
University of Quebec at Trois-Rivières,  
Trois-Rivières, Québec, Canada.  
Fadel.Toure@uqtr.ca

Mourad BADRI

Department of Mathematics and Computer Science,  
University of Quebec at Trois-Rivières,  
Trois-Rivières, Québec, Canada.  
Mourad.Badri@uqtr.ca

**Abstract**— Unit testing plays a crucial role in object-oriented software quality assurance. Unfortunately, software testing is often conducted under severe pressure due to limited resources and tight time constraints. Therefore, testing efforts have to be focused, particularly on critical classes. As a consequence, testers do not usually cover all software classes. Prioritizing unit testing effort is a crucial task. We previously investigated a unit testing prioritization approach based on software information histories. We analyzed different attributes of ten open-source Java software systems tested using the JUnit framework. We used machine learning classifiers (Multivariate Logistic Regression and Naïve Bayes) to obtain, for each system, a set of classes to be tested. The obtained sets of candidate classes have been compared to the sets of classes for which JUnit test cases have been actually developed by testers. The cross system validation (CSV) technique results showed, among others, that the sets of candidate classes suggested by machine learning classifiers properly reflect the testers' selection. In this paper, we extend our previous work by investigating more classifiers and using leave one system out validation (LOSOV) technique. This LOSOV technique uses a combination of training datasets from different systems. The obtained results indicate that: (1) the new classifiers correctly suggest classes to be tested, and (2) tested classes are particularly well predicted in the case of large-size systems.

**Key words**— Tests Prioritization; Unit Tests; Source Code Metrics; Machine Learning Classifiers.

## I. INTRODUCTION

Unit testing is one of the main phases of the testing process where each software unit is individually tested using dedicated unit test cases. In object-oriented (OO) software systems, units are software classes and testers usually write a dedicated unit test class for each software class they decided to test. The main goal is to early reveal faults in software classes. In the case of large-scale OO software systems, because of resource limitations and tight time constraints, the unit testing efforts are often focused. Testers usually select a limited set of software classes for which they write dedicated unit tests. Hence, it is important to target the most critical and fault-prone ones. However, the task is not obvious and requires a deep analysis of software. In this paper, we focus on how to automatically target suitable classes, candidates to unit testing. Our approach relies on classifiers algorithms trained on different unit tests information and source code metrics collected from different software systems.

A large number of OO metrics, related to different OO internal class attributes, have been proposed in literature [1, 2]. Some of these metrics have already been used in recent years to predict unit testability of classes in OO software systems [3-9]. The authors noticed that, for each of the analyzed systems, unit test cases have been developed only for a subset of classes. In our previous work [10], we tried to understand and determine, using different source code attributes, which criteria have been considered during the classes' selection. We investigated, using Multivariate Logistic Regression (MLR) and Naive Bayesian (NB) classifiers, how to automate and improve the selection of classes on which unit testing effort has to be focussed using source code metrics. In the current study, we considered two more well-known classifiers (K-Nearest Neighbors (KNN) and Random Forest (RF) algorithms). We also used the Cross System Validation (CSV) and the Leave One System Out Validation (LOSOV) techniques. The LOSOV technique allows, in particular, combining data from various systems as training datasets. The goal was to determine, firstly, the affinities that may exist between prediction dataset systems and training dataset systems according to their characteristics (such as size, type, category, etc.) and, secondly, to investigate to what extent the combined datasets could improve or degrade the learners' prediction levels.

The rest of the paper is organized as follows. Section 2 presents some related works. Section 3 presents the OO software metrics we used in the study. Section 4 describes the data collection procedure. Section 5 presents the empirical study that we conducted. Section 6 focuses on the main threats to validity related to our empirical experimentations. Finally, Section 7 concludes the paper, summarizes the contributions of this work and outlines several directions for future investigations.

## II. RELATED WORK

Many researchers have proposed different tests prioritization techniques in the literature, particularly in the context of regression testing. The proposed techniques are based on various criteria such as fault detection, coverage rates, software history information, and risk analysis.

In fault detection based techniques, the main goal is to run test cases that target the most fault prone components. These techniques use different factors of fault exposure as proxies, which can be estimated in different ways from the software artifacts. These approaches have been proposed, among others, by Rothermel et al. [11] and Yu and Lau [12]. Results showed that these techniques improve the fault detection rates.

In coverage based techniques, the main goal is to run test suites that cover most modified software artefacts during regression testing. The authors [13-15] used Naïve Bayes, Genetic Algorithms and different levels of granularity to implement their prioritization approaches. Results showed that coverage based techniques also lead to fault detection rate improvement. Rothermel et al. [11] compared nine test case prioritization techniques based on random prioritization, coverage prioritization and fault detection prioritization. Obtained results provide insights into the trade-offs among various techniques for test cases prioritization.

The history based prioritization uses information from previous regression tests of the same software system and current modification information in order to prioritize the new given test suites. This makes the prioritization technique unsuitable for the first regression testing of software. Kim and Porter [16] used the historical execution data to prioritize test cases for regression tests, while Lin et al. [17] investigated the weight of used information between two versions of history based prioritization techniques. The different results indicated that the history based prioritization provides a better fault detection rate.

Carlson et al. [18] mixed history and coverage based techniques using a clustering based prioritization technique. They improved the effectiveness of test cases prioritization techniques. Elbaum et al. [19] analyzed the conditions under which techniques are relevant. The obtained results provide insights and conditions into which types of prioritization techniques are or are not appropriate under specific testing scenarios.

Some other techniques allow, upstream, the prioritization of components to be tested. The goal is to optimize the testing efforts distribution by targeting the most fault prone components. Boehm and Basili [20] proposed a Pareto distribution in which 80% of all defects within software are found in 20% of the modules. Ray and Mohapatra [21] rely on that Pareto distribution to address the question of components prioritization. Shihab et al. [22] explored the prioritization for unit testing phase in the context of legacy systems.

Ray and Mohapatra' approach [21] ignores the history of the software, whereas the approach of Shihab et al. [22] is not suitable for new software. Moreover, neither approach takes advantage of the large amount of information available in the public open source repositories. In [10], we proposed the prioritization of unit test candidate classes for OO software systems. We conjectured that testers generally rely on classes' characteristics captured by source code metrics, in order to select the components to test. Thus, we proposed an approach that takes advantage of different software testers experiences and software class attributes, in order to prioritize classes (candidates) to be tested. With the same systems, the same software metrics, more learning algorithms and LOSOV technique, the current study focusses on the systems dataset affinities and the effect of merged training datasets on learner prediction performances. The long-term objective is to build a collaborative IDE plugin, based on tests information history and some specific metrics that support the tests prioritization decisions with suitable machine learning techniques.

### III. SOFTWARE METRICS

We present, in this section, the OO source code metrics we selected for the empirical study. These metrics have received considerable attention from researchers and are

also being increasingly adopted by practitioners as testability [3-9,23], maintainability [24-26], and fault proneness [26-31] indicators. These metrics have been computed using Borland Together IDE (<http://www.borland.com>). We also included the well-known source lines of code metric.

- *Coupling Between Objects*: The CBO metric counts for a given class, the number of other classes to which it is coupled and vice versa.
- *Weighted Methods per Class*: The WMC metric gives the sum of the complexities of the methods of a given class, where each method is weighted by its cyclomatic complexity [27]. Only methods specified in the class are considered.
- *Lines Of Code per class*: The LOC metric counts for a given class its number of source lines of code.

## IV. DATA COLLECTION

### A. Data collection procedure

The selected systems have been developed by different teams in Java language and tested using the JUnit framework. JUnit (<http://www.junit.org/>) is a simple framework for writing and running automated unit tests for Java classes. A typical usage of JUnit is to test each class  $C_s$  of the software by means of a dedicated test class  $C_t$ . To actually test a class  $C_s$ , we need to execute its test class  $C_t$  by calling JUnit's test runner tool. JUnit will report how many of the test methods in  $C_t$  succeeded, and how many failed.

We used the prefix/suffix linking approach, as other authors [4, 23, 32-33], to match each software class to its JUnit test class (es). Indeed, developers usually name the JUnit dedicated test classes by prefixing or suffixing the name of software class under the test by "Test" or "TestCase". We assign the modality 1 to the set of *tested classes* and the modality 0 to the remaining classes, referred as *untested classes*.

### B. Selected Systems

We extracted information from the repositories of 10 open source OO software systems that were developed in Java. For each system, only a subset of classes has been tested using the JUnit framework. We present in the following the selected systems, from small size to large-size systems.

- IO (<https://commons.apache.org/proper/commons-io/>): Commons IO is a library of utilities for developing Input/Output functionalities. It is developed by Apache Software Foundation (ASF).
- MATH (<http://commons.apache.org/proper/commons-math/>): Commons MATH is a library of lightweight, self-contained mathematics and statistics.
- JODA (<http://joda-time.sourceforge.net/>): JODA-Time is the de facto standard library for advanced date and time in Java.
- DBU (<http://dbunit.sourceforge.net/>): DbUnit is a JUnit extension (also usable with Ant) used in database-driven projects that, among others, put a database into a known state between test runs.
- LOG4J (<http://wiki.apache.org/logging-log4j/>): Log4j is a fast and flexible framework for logging applications debugging messages.
- JFC (<http://www.jfree.org/jfreechart/>): JFreechart is a free chart library for Java platform.

TABLE I: DESCRIPTIVE STATISTICS

	MATH			JFC		
	<i>CBO</i>	<i>LOC</i>	<i>WMC</i>	<i>CBO</i>	<i>LOC</i>	<i>WMC</i>
Obs.	94	94	94	411	411	411
Min.	0	2	0	0	4	0
Max.	18	660	174	101	2041	470
Sum	306	7779	1824	4861	67481	13428
$\mu$	3.255	82.755	19.404	11.827	164.187	32.672
$\sigma$	3.716	97.601	25.121	14.066	228.056	46.73
Cv	1.141	1.179	1.295	1.189	1.389	1.43
	IO			IVY		
	<i>CBO</i>	<i>LOC</i>	<i>WMC</i>	<i>CBO</i>	<i>LOC</i>	<i>WMC</i>
Obs.	100	100	100	610	610	610
Min.	0	7	1	0	2	0
Max.	39	968	250	92	1039	231
Sum	405	7604	1817	5205	50080	9664
$\mu$	4.05	76.04	18.17	8.533	82.098	15.843
$\sigma$	5.702	121.565	31.751	11.743	141.801	27.38
Cv	1.408	1.599	1.747	1.376	1.727	1.728
	JODA			LUCENE		
	<i>CBO</i>	<i>LOC</i>	<i>WMC</i>	<i>CBO</i>	<i>LOC</i>	<i>WMC</i>
Obs.	201	201	201	615	615	615
Min.	0	5	1	0	1	0
Max.	36	1760	176	55	2644	557
Sum	1596	31339	6269	3793	56108	10803
$\mu$	7.94	155.915	31.189	6.167	91.233	17.566
$\sigma$	6.443	210.974	30.553	7.243	192.874	35.704
Cv	0.811	1.353	0.98	1.174	2.114	2.033
	DBU			ANT		
	<i>CBO</i>	<i>LOC</i>	<i>WMC</i>	<i>CBO</i>	<i>LOC</i>	<i>WMC</i>
Obs.	213	213	213	663	663	663
Min.	0	4	1	0	1	0
Max.	24	488	61	41	1252	245
Sum	1316	12187	1989	4613	63548	12034
$\mu$	6.178	57.216	9.338	6.958	95.849	18.151
$\sigma$	5.319	60.546	9.451	7.25	132.915	24.168
Cv	0.861	1.058	1.012	1.042	1.387	1.332
	LOG4J			POI		
	<i>CBO</i>	<i>LOC</i>	<i>WMC</i>	<i>CBO</i>	<i>LOC</i>	<i>WMC</i>
Obs.	231	231	231	1382	1382	1382
Min.	0	5	1	0	2	0
Max.	107	1103	207	168	1686	374
Sum	1698	20150	3694	9660	130185	23810
$\mu$	7.351	87.229	15.991	6.99	94.2	17.229
$\sigma$	10.119	130.419	25.7	10.782	154.282	28.319
Cv	1.377	1.495	1.607	1.543	1.638	1.644

- IVY (<http://ant.apache.org/ivy/>): IVY is a simple and flexible agile dependency manager tightly integrated with Apache Ant.
- LUCENE (<http://lucene.apache.org/>): LUCENE is a high-performance, full-featured text search engine library suitable for applications requiring full-text search.
- ANT (<http://www.apache.org/>): ANT is a Java library and command-line tool that drives processes described in build files as target.
- POI (<http://poi.apache.org/>): POI is a Java APIs for manipulating various file formats based upon the Office Open XML standards and Microsoft's OLE2.

### C. Descriptive Statistics

Table I summarizes the statistics of selected metrics of all systems. It shows that the considered systems are of different sizes. The number of lines of code varies from 7,600 lines spread over 100 software classes (IO), to more than 130,185 lines of code over 1,382 software classes (POI). Table I also suggests 4 groups of systems according to their size: (1) the small-size systems, about 100 classes (IO and MATH), (2) the medium-size systems around 200 classes (LOG4J, DBU and JODA), (3) the large-size systems, between 400 and 600 classes (LUCENE, IVY, ANT and JFC), and (4) the very large-size systems over than 1,000 software classes (POI).

The average cyclomatic complexity varies widely between systems with similar sizes. Indeed, the medium-size systems, JODA and DBU, have a quite different average of cyclomatic complexity (9.34 vs 31.18). Similar trend is observed for LUCENE and JFC systems. In the dataset, each row has a binary attribute TESTED taking modalities 1 or 0 indicating whether it is a *tested class* or *untested class*.

## V. EMPIRICAL ANALYSIS

### A. Research questions

The current study focusses on the systems' dataset affinities and the effect of mixing datasets on classifiers' performance. We tried to respond to the following research questions:

- RQ1- Can other well-known machine learning algorithms correctly predict the testers' selections?
- RQ2- To what extent can we mix dataset histories of different systems to predict testers' selections for a given new system.

### B. Goals

The goal of our first research question (RQ1) is to validate our previous results using other machine learning algorithms, to compare their prediction performances, and to determine whether they depend (or not) on the type or size

TABLE II: CROSS SYSTEM VALIDATIONS

		MATH	IO	JODA	DBU	LOG4J	JFC	IVY	LUCENE	ANT	POI
MATH	LR	<b>0.745</b>	0.590	0.338	0.404	0.160	0.414	0.188	0.184	0.193	0.295
	NB	<b>0.723</b>	0.620	0.478	0.521	0.411	0.606	0.400	0.405	0.363	0.418
	KNN	<b>0.734</b>	0.64	0.527	0.563	0.416	0.431	0.53	0.48	0.431	0.446
	RF	<b>0.926</b>	0.58	0.338	0.469	0.32	0.397	0.382	0.358	0.398	0.391
IO	LR	0.617	<b>0.740</b>	0.458	0.624	0.329	0.623	0.418	0.403	0.359	0.395
	NB	0.628	<b>0.710</b>	0.413	0.667	0.494	0.652	0.454	0.433	0.403	0.505
	KNN	0.585	<b>0.79</b>	0.418	0.592	0.364	0.275	0.334	0.304	0.275	0.378
	RF	0.649	<b>0.94</b>	0.388	0.493	0.394	0.305	0.301	0.366	0.276	0.373
JODA	LR	0.415	0.390	<b>0.711</b>	0.620	<b>0.779</b>	0.620	<b>0.814</b>	<b>0.807</b>	<b>0.796</b>	<b>0.721</b>
	NB	0.457	0.420	0.692	0.620	<b>0.771</b>	0.681	<b>0.799</b>	<b>0.800</b>	<b>0.742</b>	<b>0.721</b>
	KNN	0.468	0.39	<b>0.841</b>	0.446	0.632	0.698	0.692	<b>0.706</b>	0.698	0.633
	RF	0.447	0.43	<b>0.95</b>	0.582	0.684	0.656	<b>0.729</b>	0.62	0.649	0.656
DBU	LR	0.457	0.490	0.622	0.671	0.645	0.664	<b>0.743</b>	<b>0.725</b>	0.677	0.681
	NB	0.489	0.600	0.448	<b>0.756</b>	0.589	0.684	0.617	0.571	0.511	0.627
	KNN	0.543	0.57	0.453	<b>0.85</b>	0.563	0.538	0.671	0.558	0.538	0.567
	RF	0.489	0.49	0.453	<b>0.967</b>	0.563	0.561	0.666	0.58	0.555	0.588
LOG4J	LR	0.436	0.440	0.647	0.554	<b>0.840</b>	0.577	<b>0.818</b>	<b>0.816</b>	<b>0.804</b>	<b>0.728</b>
	NB	0.468	0.490	0.637	0.629	<b>0.758</b>	0.672	<b>0.783</b>	<b>0.774</b>	0.692	<b>0.744</b>
	KNN	0.394	0.44	0.612	0.577	<b>0.883</b>	<b>0.768</b>	<b>0.803</b>	<b>0.78</b>	<b>0.768</b>	<b>0.719</b>
	RF	0.415	0.47	0.602	0.559	<b>0.961</b>	<b>0.724</b>	<b>0.78</b>	<b>0.735</b>	<b>0.719</b>	<b>0.716</b>
JFC	LR	0.511	0.500	0.602	0.601	<b>0.736</b>	0.698	<b>0.766</b>	<b>0.771</b>	0.697	<b>0.724</b>
	NB	0.511	0.530	0.597	0.685	<b>0.701</b>	0.689	<b>0.730</b>	<b>0.736</b>	0.632	<b>0.707</b>
	KNN	0.404	0.38	0.632	0.592	<b>0.775</b>	<b>0.855</b>	<b>0.794</b>	<b>0.782</b>	<b>0.855</b>	0.69
	RF	0.383	0.37	0.667	0.559	<b>0.749</b>	<b>0.952</b>	<b>0.77</b>	<b>0.766</b>	<b>0.958</b>	0.686
IVY	LR	0.383	0.340	0.622	0.596	<b>0.801</b>	0.511	<b>0.822</b>	<b>0.820</b>	<b>0.828</b>	<b>0.726</b>
	NB	0.479	0.480	0.667	0.634	<b>0.749</b>	0.696	<b>0.786</b>	<b>0.772</b>	<b>0.707</b>	<b>0.728</b>
	KNN	0.404	0.4	0.657	0.592	<b>0.775</b>	<b>0.786</b>	<b>0.893</b>	<b>0.784</b>	<b>0.786</b>	<b>0.726</b>
	RF	0.415	0.42	0.642	0.629	<b>0.766</b>	<b>0.736</b>	<b>0.964</b>	<b>0.777</b>	<b>0.741</b>	<b>0.708</b>
LUCENE	LR	0.394	0.360	0.657	0.596	<b>0.814</b>	0.533	<b>0.823</b>	<b>0.820</b>	<b>0.706</b>	<b>0.730</b>
	NB	0.468	0.500	0.662	0.615	<b>0.753</b>	0.689	<b>0.784</b>	<b>0.777</b>	<b>0.706</b>	<b>0.736</b>
	KNN	0.394	0.45	0.612	0.592	<b>0.805</b>	<b>0.789</b>	<b>0.803</b>	<b>0.863</b>	<b>0.789</b>	<b>0.711</b>
	RF	0.426	0.42	0.592	0.554	<b>0.753</b>	<b>0.753</b>	<b>0.798</b>	<b>0.964</b>	<b>0.765</b>	<b>0.705</b>
ANT	LR	0.404	0.360	0.692	0.596	<b>0.823</b>	0.513	<b>0.820</b>	<b>0.816</b>	<b>0.833</b>	<b>0.728</b>
	NB	0.394	0.390	0.692	0.629	<b>0.779</b>	0.628	<b>0.814</b>	<b>0.813</b>	<b>0.778</b>	<b>0.735</b>
	KNN	0.404	0.38	0.632	0.592	<b>0.775</b>	<b>0.855</b>	<b>0.794</b>	<b>0.782</b>	<b>0.855</b>	0.69
	RF	0.404	0.4	0.647	0.573	<b>0.762</b>	<b>0.964</b>	<b>0.788</b>	<b>0.753</b>	<b>0.965</b>	0.696
POI	LR	0.415	0.370	0.682	0.596	<b>0.823</b>	0.582	<b>0.822</b>	<b>0.816</b>	<b>0.837</b>	<b>0.728</b>
	NB	0.511	0.540	0.527	<b>0.709</b>	<b>0.714</b>	0.681	<b>0.725</b>	<b>0.712</b>	0.605	<b>0.718</b>
	KNN	0.489	0.46	0.572	0.568	<b>0.745</b>	<b>0.7</b>	<b>0.78</b>	<b>0.746</b>	<b>0.7</b>	<b>0.846</b>
	RF	0.5	0.49	0.617	0.592	<b>0.736</b>	0.698	<b>0.76</b>	<b>0.756</b>	0.697	<b>0.94</b>

of considered applications (affinities). In the experiment that we conducted to answer RQ1, the training datasets are composed of 1 system tests information history at the time. Thus, we could determine potential prediction affinities between different systems which could lead to introduce more affinity parameters that may be related to the systems type, or size, in order to determine the suitable kind of training dataset for a given new system. To address the second research question (RQ2), we conducted an empirical study that uses mixed training datasets from different systems, builds a classifier and tries to determine whether the datasets mixing can improve or degrade the predictions obtained in the previous experiment.

### C. Classifiers and cross-system validation

We used machine learning classifiers trained on a system's test information data to provide a set of classes to be tested for other software systems. We added to the MLR and NB classifiers previously considered, the well-known K-KNN and RF algorithms to build prediction models from the datasets. KNN is an intuitive and fast algorithm that classifies observations according to their similarity. KNN is particularly suitable for pattern recognition. RF behaves well when irrelevant features are present or these features have skewed distributions. The larger the training dataset the more accurate the classifier. It makes RF particularly interesting when the training dataset is a combined information of different systems.

#### 1) Cross System Validation

In the CSV technique, datasets collected from each of the 10 systems are used in turn as training set and the derived classifiers are cross-validated on each of the 9 remaining systems. In such approach, training dataset may be small depending on the considered system, hence the usefulness of NB classifiers which can perform on small datasets.

Table II presents the performances of models derived from MLR, NB, KNN and RF classifiers. In each table, the  $cell(i,j)$  shows the accuracy (1-error) of the 4 classifiers built from training dataset of the system on row  $i$ , and tested on the dataset of the system on column  $j$ . The diagonal  $cells(k,k)$  hold the adjustment (1 - optimistic error) of classifiers on the dataset of system  $k$ . We considered the models with accuracy values greater than 0.70 (error < 0.30) as good classifiers. In Table II, systems are sorted from the smallest one to the largest one in terms of number of classes.

The results show that the very small-size systems (MATH and IO) form bad training datasets and are not well predicted. All non-adjustment prediction rates related to both of the systems are smaller than 70%. Three reasons may explain these results:

- The small size of the systems. IO (with 100 classes) and MATH (with 125 classes) are the smallest systems in our datasets. The dataset they form may not be large enough to build good classifiers.
- The use of very specific criteria when selecting classes to be tested leading to over-fitting problems. This hypothesis is supported by the optimistic prediction rates observed on both systems' diagonal cells.

- The lack of a unit test strategy in small-size systems. With limited software classes, the testers could cover a higher percent of classes without any strategy. Thus, the poor rules selection makes irrelevant the information captured by software metrics. Candidate classes become unpredictable.

The medium-size systems (DBU JODA and LOG4J) results are mitigated. LR and NB models based on JODA training set well predicted all the larger systems except JFC. JODA’s testers may use common criteria that are also used when testing larger systems, while DBU testers seem to use a particular strategy. As a consequence, we obtained the same performances as for the small-size systems. The same previous reason may explain DBU’s results. LOG4J results are similar to large-size systems.

The LOG4J system, the large-size and the very large-size systems form good training sets and are well predicted by classifiers trained on their datasets. The results may be explained by the adoption of an effective tests prioritization strategy by testers in order to target the critical classes. Indeed, in large-size systems, testers may adopt an effective strategy to carefully select the software classes to be tested. The adopted strategy may suggest large, complex and highly coupled classes as unit test candidates. Complexity and coupling attributes are captured by the LOC, WMC and CBO metrics. This may explain why the associated systems are well predicted and form good training datasets.

JFC, ANT and IVY are of standalone type systems while remaining systems are libraries. The results suggest no affinity based on the application type. This is supported by the fact that many libraries are not well predicted by other libraries datasets. The good predictions obtained between standalone apps seem to be related to their size affinity.

The cross-validation results, especially for the medium, large and very large-size systems, show that it is possible, based on only a combination of metrics, to construct classifier models from existing software datasets that automatically suggest, for another software system, a set of classes to be tested. It also indicates that small systems are unpredictable and do not form good training datasets. Furthermore, the most obvious affinity between systems is related to their size. The larger the systems, the better the training datasets and the prediction levels of learners.

## 2) Leave One System Out Validation

In Leave One System Out Validation (LOSOV) technique, each system will be used in turn as testing dataset to validate the classifiers derived from the training dataset formed by the 9 remaining systems. This approach uses large training datasets and combines different selection criteria if they exist. However, if one of the systems’ testers randomly selects classes to be tested, it may impact the whole training dataset quality depending on that system size.

Table III presents the accuracy rates of the classifiers for LOSOV. The results confirm those obtained using the CSV and show that each of medium, large and very large size systems are well predicted by classifiers trained on merged dataset of 9 remaining systems. The best accuracy rates vary from 70.1% to 93.2%.

Some classifiers failed to correctly predict some systems: NB on JFC system, KNN on POI system and RF on POI system. The POI data test prediction results may be explained by its size. Indeed, leaving POI out from training datasets during the LOSOV may drastically reduce the train-

ing dataset (POI represents 30.7% of classes and 30.5% of tested classes).

TABLE III: LEAVE ONE SYSTEM OUT VALIDATIONS

	LR	NB	KNN	RF
MATH	0.404	0.457	0.447	0.468
IO	0.37	0.53	0.5	0.41
JODA	0.667	0.627	0.567	0.647
DBU	0.596	0.629	0.573	0.559
LOG4J	<b>0.818</b>	<b>0.736</b>	<b>0.71</b>	<b>0.701</b>
JFC	<b>0.834</b>	0.661	<b>0.846</b>	<b>0.92</b>
IVY	<b>0.84</b>	<b>0.796</b>	<b>0.76</b>	<b>0.747</b>
LUCENE	<b>0.82</b>	<b>0.746</b>	<b>0.732</b>	<b>0.702</b>
ANT	<b>0.834</b>	0.661	<b>0.839</b>	<b>0.932</b>
POI	<b>0.731</b>	<b>0.734</b>	0.681	0.679

The small-size systems are not well predicted since all classifiers’ prediction rates are smaller than 70%. This result suggests that their testers used very specific criteria or uncaptured (by metrics) criteria, or may be no criteria during the selection candidate classes. This result is plausible for small systems, since testers could test all classes, they also may not need any particular strategy.

Compared with CSV results, LOSOV slightly improve and degrade the prediction levels in several cases. However, LOSOV takes advantage of being usable in real conditions, in the context of a collaborative tool that supports unit tests prioritization when different information history (metrics and tests data) are provided by different teams of developers.

## VI. THREATS TO VALIDITY

The study we presented in this paper was performed on 10 open-source systems containing almost a half million lines of code (453K). The sample is large enough to allow obtaining significant results, but the measuring methods and approaches have limitations that can restrict the generalization of certain conclusions.

The external validity threats are mainly related to the application domain of considered systems. Indeed, some analyzed systems are mathematical algorithms libraries (IO), while other systems have more complex architectures and involve many OO-technology specific artifacts such as inheritance and polymorphism (JFC). Hence, when a learning algorithm is trained on some types of systems, it could be well-adjusted when tested on the datasets of similar domain systems and not able to suggest good candidate classes for other types and domain systems.

The data we collected does not provide any information on tested classes selection criteria. It may be that, for some systems, tested classes were randomly selected particularly for the small size systems.

The main threat of construct validity lies in the technique we used to match JUnit test classes to software classes during the *tested classes* identification. Indeed, the remaining unpaired software classes that are tested by transitive method invocations are ignored by our approach.

## VII. CONCLUSIONS AND FUTURE WORK

Ten open source (Java) software systems have been analyzed in this study and totalize more than 4400 software classes. The testers of each system developed dedicated unit test classes for a subset of classes using the JUnit Framework. In our previous investigations, we explored the possibility of explaining and reusing the selection criteria for different systems through three experiments using three source code metrics.

This study extends our previous work by including the



KNN and RF classifiers to the MLR and NB classifiers. In addition to the CSV, we used LOSOV validation technique, which merges training sets from different systems. The main objective was to know to what extents the combined information of different systems could be a good training set for the learners and to determine if there exist affinities between different datasets of systems' test information history. Results show that systems with more than a hundred classes have their *tested classed* generally well predicted by classifiers built from medium, large and very large systems training datasets. Furthermore, the obtained results suggest that all obtained classifiers could help to support unit tests prioritization with more than 70% of accurate predictions. The results of the experiments we conducted become particularly interesting knowing that effort prioritization is especially useful during large and complex systems testing. It demonstrates the viability of a unit tests prioritization automation technique that uses classifiers trained on merged software source code metrics with the unit tests information history. Grouping the systems according to their domains could improve our results. Since the proposed prioritization technique suggests a slightly different (30%) tested classes from those of the testers, it would be pertinent to analyze and compare their actual performance on covering faulty classes. This topic will be the next direction of our investigations.

#### REFERENCES

- [1] Chidamber S.R. and Kemerer C.F., 1994. A Metrics Suite for Object Oriented Design, IEEE Transactions on Software Engineering, vol. 20, no. 6, pp. 476–493.
- [2] Henderson-Sellers B. 1996. Object-Oriented Metrics Measures of Complexity, Prentice-Hall, Upper Saddle River.
- [3] Gupta V., Aggarwal K.K. and Singh Y., 2005, A Fuzzy Approach for Integrated Measure of Object-Oriented Software Testability, Journal of Computer Science, Vol. 1, No. 2, pp. 276-282.
- [4] Bruntink M. and Van Deursen A. 2006. An Empirical Study into Class Testability, Journal of Systems and Software, Vol. 79, No. 9, pp. 1219-1232.
- [5] Badri L., Badri M. and Toure F., 2010. Exploring Empirically the Relationship between Lack of Cohesion and Testability in Object-Oriented Systems, JSEA Eds., Advances in Software Engineering, Communications in Computer and Information Science, Vol. 117, Springer, Berlin.
- [6] Badri M. and Toure F., 2011. Empirical analysis for investigating the effect of control flow dependencies on testability of classes, in Proceedings of the 23rd International Conference on Software Engineering and Knowledge Engineering SEKE.
- [7] Badri M. and Toure F. 2012. Empirical analysis of object oriented design metrics for predicting unit testing effort of classes, Journal of Software Engineering and Applications (JSEA), Vol. 5 No. 7, pp.513-526.
- [8] Toure F., Badri M. and Lamontagne L., 2014. Towards a metrics suite for JUnit Test Cases. In Proceedings of the 26th International Conference on Software Engineering and Knowledge Engineering (SEKE Vancouver, Canada. Knowledge Systems Institute Graduate School, USA pp 115–120.
- [9] Toure F., Badri M. and Lamontagne L., 2014. A metrics suite for JUnit test code: a multiple case study on open source software, Journal of Software Engineering Research and Development, Springer, 2:14.
- [10] Toure F., Badri M. and Lamontagne L., 2017. Investigating the Prioritization of Unit Testing Effort Using Software Metrics, In Proceedings of the 12th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE'17) Volume 1: ENASE, pages 69-80.
- [11] Rothermel G., Untch R.H., Chu C. and Harrold M.J., 1999. Test case prioritization: an empirical study, International Conference on Software Maintenance, Oxford, UK, pp. 179–188.
- [12] Yu Y. T. and Lau M. F., 2012. Fault-based test suite prioritization for specification-based testing, Information and Software Technology Volume 54, Issue 2, Pages 179–202.
- [13] Wong W., Horgan J., London S., and Agrawal, H., 1997. A study of effective regression in practice, Proceedings of the 8th International Symposium on Software Reliability Engineering, November, p. 230–238.
- [14] Mirarab S. and Tahvildari L., 2007. A prioritization approach for software test cases on Bayesian networks, In FASE, LNCS 4422-0276, pages 276–290.
- [15] Walcott K.R., Soffa M.L., Kapfhammer G.M. and Roos R.S., 2006. Time aware test suite prioritization, Proceedings of the International Symposium on Software Testing and Analysis (ISSTA 2006). ACM Press, New York, 1–12.
- [16] Kim J. and Porter A., 2002. A history-based test prioritization technique for regression testing in resource constrained environments, In Proceedings of the International Conference on Software Engineering.
- [17] Lin C.T., Chen C.D., Tsai C.S. and Kapfhammer G. M., 2013. History-based Test Case Prioritization with Software Version Awareness, 18th International Conference on Engineering of Complex Computer Systems.
- [18] Carlson R., Do H., and Denton A., 2011. A clustering approach to improving test case prioritization: An industrial case study, Software Maintenance, 27th IEEE International Conference, ICSM, pp. 382-391.
- [19] Elbaum S., Rothermel G., Kanduri S. and Malishevsky A.G., 2004. Selecting a cost-effective test case prioritization technique, Software Quality Control, 12(3):185–210.
- [20] Boehm B. and Basili V. R., 2001. Software defect reduction top-10 list, Computer, vol. 34, no. 1, pp. 135–137.
- [21] Ray M. and Mohapatra D.P., 2012. Prioritizing Program elements: A pretesting effort to improve software quality, International Scholarly Research Network, ISRN Software Engineering.
- [22] Shihaby E., Jiangu Z. M., Adamsy B., Ahmed E. Hassany A. and Bowermanx R., 2010. Prioritizing the Creation of Unit Tests in Legacy Software Systems, Softw. Pract. Exper., 00:1–22.
- [23] Bruntink M., and Deursen A.V., 2004. Predicting Class Testability using Object-Oriented Metrics, 4th Int. Workshop on Source Code Analysis and Manipulation (SCAM), IEEE.
- [24] Li W., and Henry S., 1993. Object-Oriented Metrics that Predict Maintainability Journal of Systems and Software, vol. 23 no. 2 pp. 111-122.
- [25] Dagginar M., and Jahnke J., 2003. Predicting maintainability with object-oriented metrics – an empirical comparison, Proceedings of the 10th Working Conference on Reverse Engineering (WCRE), IEEE Computer Society, pp. 155–164.
- [26] Zhou Y., and Leung H., 2007. Predicting object-oriented software maintainability using multivariate adaptive regression splines, Journal of Systems and Software, Volume 80, Issue 8, August 2007, Pages 1349-1361, ISSN 0164-1212.
- [27] McCabe T. J., 1976. A Complexity Measure, IEEE Transactions on Software Engineering: 308–320.
- [28] Basili V.R., Briand L.C. and Melo W.L., 1996. A Validation of Object-Oriented Design Metrics as Quality Indicators, IEEE Transactions on Software Engineering. vol. 22, no. 10, pp. 751-761.
- [29] Aggarwal K.K., Singh Y., Kaur A., and Malhotra R., 2009. Empirical Analysis for Investigating the Effect of Object-Oriented Metrics on Fault Proneness: A Replicated Case Study, Software Process Improvement and Practice, vol. 14, no. 1, pp. 39-62.
- [30] Shatnawi R., 2010. A Quantitative Investigation of the Acceptable Risk Levels of Object-Oriented Metrics in Open-Source Systems, IEEE Transactions On Software Engineering, Vol. 36, No. 2.
- [31] Zhou Y. and Leung H., 2006. Empirical Analysis of Object-Oriented Design Metrics for Predicting High and Low Severity Faults, IEEE Transaction Software Engineering, vol. 32, no. 10, pp. 771-789.
- [32] Mockus A., Nagappan N. and Dinh-Trong T. T., 2009. Test coverage and post-verification defects: a multiple case study, in Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement (ESEM), pp. 291– 301.
- [33] Rompaey B. V. and Demeyer S., 2009. Establishing traceability links between unit test cases and units under test, in Proceedings of the 13th European Conference on Software Maintenance and Reengineering (CSMR '09), pp. 209–218.

# Security Analysis of the Access Control Solution of NDN Using BAN Logic

Yuan Fei    Huibiao Zhu\*    Huiwen Wang  
Shanghai Key Laboratory of Trustworthy Computing,  
School of Computer Science and Software Engineering,  
East China Normal University, Shanghai, China

**Abstract**—Named Data Networking (NDN) is a new promising architecture of information-centric networking. For its caching property, traditional mechanisms of access control can no longer work. Hamdane et al. propose a new access control solution for both closed and open environments. In this paper, we make the very first attempt to formally analyze this access control solution. Inspired by the basic BAN logic which is often used to describe protocols by logical formulas, we present our BAN-like logic by adding some new notions to make it suitable for the access control solution. Using the BAN-like logic, the procedures of the access control solution is idealized in the form of the beliefs of principals. Then the idealized procedures are analyzed under several security goals with a set of logical postulates. Several unsatisfied goals may lead the access control solution to be vulnerable to intruders. We give the modification in the idealized procedures to archive more goals. We also present the related modification in the implementation of the access control solution. Our study helps to improve security and protect against various attacks for the access control solution.

**Keywords**—Named Data Networking (NDN), Access Control Solution, BAN Logic

## I. INTRODUCTION

Named Data Networking (NDN) [1] is an architecture of Information-Centric Networking (ICN). ICN aims to offer solutions to problems existing in TCP/IP Internet. Nowadays users pay more attention to named content rather than its location. Though TCP/IP Internet has shown great resilience over the years, it cannot support the newly evolving content distribution model successfully. One of the promising candidates of ICN is NDN, which supports multicast of data and adopts the publish/subscribe model. The data producers mean publishers and the data consumers represent subscribers in NDN. When data consumer needs data, it sends out an *Interest* packet with a required name of the data; according to the name, routers forward the packet over the network; and a *Data* packet is returned to the consumer when a data produced by the data producer is matched. NDN routers can cache previous forwarded *Data* packets, which are able to be reused when a matching *Interest* packet comes. For this caching property, traditional mechanisms of access control, as a way of limiting access to data, can no longer work. Some access control specifications [2], [3] are proposed for NDN. However, each owns several limits. Hamdane et al. [4] put forward a new access control solution to address these limits.

\*Corresponding Author. E-mail address: hbzhu@sei.ecnu.edu.cn (H. Zhu).

We focus on this solution and analyze it step by step with our BAN-like logic.

The BAN logic [5] was first proposed by Burrows, Abadi and Needham in 1989. It provides a way to formalize the description and analysis of authentication protocols. It has been applied to analyze existing protocols and to find out their flaws [5]. Gaarder et al. [6] introduced new notions based on the basic BAN logic specially for PKCS authentication protocols. In order to reason about cryptographic protocols, Gong et al. [7] added more accurate concepts and definitions to the basic BAN logic. By adding negation, [8] presented the special BAN logic designed for monotonic protocols.

Our BAN-like logic is inspired by the basic BAN logic. We add some new notions to make it suitable for the access control solution. Adopting the BAN-like logic, the procedures of the access control solution in [4] is idealized. Then the idealized procedures are analyzed under several security goals with a set of logical postulates. The results show that some goals could not be archived. It indicates that this access control solution cannot ensure the source of critical keys and data. This may lead the access control solution to be vulnerable to intruders. We give the modification of the idealized procedures and the proofs of them. Meanwhile we also present the related modification in the implementation of the access control solution. Our study helps to improve security and protect against various attacks for the access control solution.

The rest of the paper is organized as follows: Section II briefly introduces the access control solution of NDN and explains how it works. Section III gives the BAN-like logic definitions and the assumptions of the access control solution. In Section IV and Section V, we apply BAN-like logic to analyze the access control solution of write and read operations in a closed environment. Section VI displays the analysis of write and read operation in an open environment. Finally, Section VII concludes and points out the future work.

## II. ACCESS CONTROL SOLUTION OF NDN

In this section, we introduce the access control solution [4] for NDN. We give the related entities and assumptions. The write operations and read operations in closed and open environments are proposed.

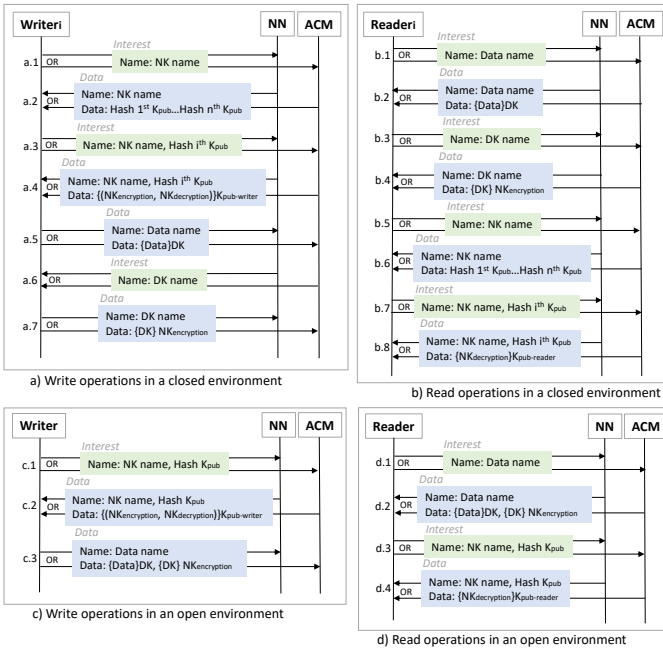


Fig. 1. Write and Read operations in closed and open environments (simplified from Hamdane et al. [4])

## A. Entities and Assumptions

Entities in NDN own two roles: readers and writers. An access control manager (ACM) is introduced to control the management of the access control policy. It creates a key pair  $(NK_{encryption}, NK_{decryption})$ , which is designed for encrypting a symmetric data key  $DK$ . These keys are similar as public and private keys, but both of them are secret.  $NK_{encryption}$  is applied to encrypt the  $DK$  and only be acquired by entities with write privilege.  $NK_{decryption}$  allows the  $DK$  decryption which can be obtained by any entities. Meanwhile, every entity owns the related public and private key pair. The public writer keys are used to encrypt key pair  $(NK_{encryption}, NK_{decryption})$ , and public reader keys are adopted to encrypt key  $NK_{encryption}$ .

## B. Write Operations and Read Operations

We simplify some steps of write and read operations in [4] and only retain the processing related with keys and data. Fig.1 illustrates four situations of write and read operations in different environments. Because the node playing reader's or writers role can be connected with a normal node or the ACM, the communication object of  $Writer_i$  and  $Reader_i$  can be ACM or NN. NN is only responsible to transit message to ACM eventually. As a result,  $Writer_i$  and  $Reader_i$  are communicating with ACM essentially.

### 1) Write operations in a closed environment:

Assuming that  $Writer_i$  knows the name of key pair  $(NK_{encryption}, NK_{decryption})$  in advance, it sends an Interest packet containing  $NK$  name to ACM (step a.1). ACM returns a Data packet with all the hash value of public keys that ACM has already known (step a.2). Then  $Writer_i$  recognizes its own hash and transmits it together with  $NK$  name as a

new Interest to ACM (step a.3). ACM recognizes which writer is communicating with it, and uses  $Writer_i$ 's public key to encrypt key pair  $(NK_{encryption}, NK_{decryption})$ . This is added to a new Data packet which is fed back to ACM (step a.4). Then  $Writer_i$  sends the encrypted data  $DataDK$  to ACM (step a.5). ACM learns  $DK$  name and sends the Interest to ACM (step a.6). ACM then uses  $NK_{encryption}$  to encrypt  $DK$  and produces a new Data packet as a reply (step a.7).

2) Read operations in a closed environment:  $Reader_i$  sends the required Data name to ACM (step b.1). ACM replies  $Reader_i$  with Data packet including Data encrypted with data key  $DK$  (step b.2).  $Reader_i$  knows the name of data key  $DK$  and sends the Interest to ACM (step b.3). ACM uses  $NK_{encryption}$  to encrypt  $DK$  and creates a Data packet to send back to  $Reader_i$  (step b.4).  $Reader_i$  sends the Interest packet carrying name of key pair  $(NK_{encryption}, NK_{decryption})$  to ACM (step b.5). ACM returns all the hash value of public keys (step b.6).  $Reader_i$  gives ACM with  $NK$  name and its hash value of public key (step b.7). ACM uses the related public key to encrypt  $NK_{decryption}$  to produce a Data packet for ACM (step b.8).

3) Writer operations in an open environment: As the environment is open,  $Writer$  is unknown to ACM. As a result,  $Writer$  needs to send ACM its hash value of public key. ACM can get the related public key to encrypt information. First,  $Writer$  sends  $NK$  name and the hash value of its public key to ACM (step c.1). As ACM has acknowledged the public key of  $Writer$ , it uses the key to encrypt key pair  $(NK_{encryption}, NK_{decryption})$  (step c.2).  $Writer$  sends the encrypted data  $\{Data\}DK$ , together with encrypted data key  $\{DK\} NK_{encryption}$  (step c.3).

4) Read operations in an open environment: Because the environment is open, ACM does not acknowledge the public key of  $Reader$ . So it is necessary for  $Reader$  to transmit the hash value of its public key to ACM. First,  $Reader$  sends an Interest with Data name to ACM (step d.1). ACM responds the encrypted data  $\{Data\}DK$  and the encrypted data key  $\{DK\} NK_{encryption}$  (step d.2).  $Reader$  transmits  $NK$  name and its hash value of public key to ACM (step d.3). Because ACM realizes the public key of  $Reader$ , it applies this key to encrypt  $NK_{decryption}$  which is built into a Data packet and conveyed to  $Reader$  (step d.4).

## III. AN INTRODUCTION OF BAN-LIKE LOGIC AND ASSUMPTIONS

In this section, we introduce our BAN-like logic inspired by the basic BAN logic. We add some new notions to make it suitable for the access control solution. Assumptions of access control solution are also presented for further analysis.

### A. Statements

There are three sorts of objects in our BAN-like logic: principals, keys, and formulas (also statements). The symbols  $P$  and  $Q$  range over principles;  $K$  ranges over keys;  $X$  ranges over formulas. The following are basic statements.

- $P \models X$ : The principal  $P$  believes the statement  $X$  is true.

- $P \triangleleft X$ :  $P$  sees  $X$ , which represents  $P$  has received a message  $X$ .
- $P \vdash X$ :  $P$  once said  $X$ , which also means  $P \models X$  when  $P$  sent it.
- $P \Rightarrow X$ :  $P$  governs  $X$ , showing that  $P$  has an authority on  $X$ .
- $\#(X)$ : The message  $X$  is fresh.

There is little difference between the keys in the basic BAN logic and the key used here. We add new notations for a better explanation. The key  $K$  represents the public key of asymmetric keys, which can be seen by other agents. Its associated private key  $K^{-1}$  will be secret to any other agents except one agent which owns it. The key pair  $(@K, K@)$  denotes an asymmetric key pair, in which  $@K$  is used for encryption and  $K@$  is devoted to decryption. An agent can only acquire this key pair from the package it received, if it is not the creator of the pair. The symmetric key  $\$K$  is applied for both encryption and decryption, which is encrypted and transmitted between agents.

- $\xrightarrow{K}$ :  $P$ : The encryption key  $K$  is the public key of  $P$ . The matching private key  $K^{-1}$  will be secret to any other principals except  $P$ .
- $\xrightarrow{@K}$ :  $P$ : The key  $@K$  is the encryption key of the asymmetric key pair  $(@K, K@)$ , which is acknowledged by  $P$ .
- $\xrightarrow{K@}$ :  $P$ :  $P$  knows the key  $K@$ , which is the decryption key of the asymmetric key pair  $(@K, K@)$ .
- $\xrightarrow{\$K}$ :  $P$ :  $P$  learns that the key  $\$K$  is a symmetric key.
- $\{X\}_K$ : The statement  $X$  is encrypted under the key  $K$ .  $K$  can also be replaced by  $@K$  or  $\$K$ .

## B. Logical Postulates

We introduce four categories of postulates and give their explanations.

(1) The *message-meaning* rules are about interpretation of encrypted messages. We postulate the *message-meaning* rule for symmetric keys as below.

$$\text{MM1} \quad \frac{P \models \xrightarrow{\$K} P, Q \models \xrightarrow{\$K} Q, P \triangleleft \{X\}_{\$K}}{P \models Q \vdash X}$$

If both  $P$  and  $Q$  believe that the key  $K$  is the symmetric key and  $P$  sees a message  $X$  encrypted under  $K$ , then  $P$  believes that  $Q$  once said  $X$ .

We also present the *message-meaning* rule for asymmetric keys as below.

$$\text{MM2} \quad \frac{P \models \xrightarrow{K} Q, P \triangleleft \{X\}_{K^{-1}}}{P \models Q \vdash X}$$

If  $P$  believes that the key  $K$  is the public key of  $Q$  and sees a message  $X$  encrypted under  $K^{-1}$ , then  $P$  believes that  $Q$  once said  $X$ .

(2) The *nonce-verification* rule states the check of the freshness of a message.

$$\text{NV} \quad \frac{P \models \#(X), P \models Q \vdash X}{P \models Q \models X}$$

If  $P$  believes a formula  $X$  is fresh and  $P$  believes that  $Q$  once said formula  $X$ , then  $P$  believes that  $Q$  believes  $X$ .

(3) The *jurisdiction* rule expresses how jurisdiction effects the belief.

$$\text{J} \quad \frac{P \models Q \Rightarrow X, P \models Q \models X}{P \models X}$$

If  $P$  believes that  $Q$  has jurisdiction over  $X$  and  $Q$  believes  $X$ , then  $P$  trusts  $X$ .

(4) The *seeing* rule describes the situation when a principal sees a formula encrypted with different kinds of keys.

$$\text{SEE1} \quad \frac{P \triangleleft \{X\}_K, P \models \xrightarrow{K} P}{P \triangleleft X}$$

If  $P$  sees  $X$  encrypted with a public key  $K$  and processes the corresponding private key  $K^{-1}$ , then  $P$  is considered to have seen  $X$ .

$$\text{SEE2} \quad \frac{P \triangleleft \{X\}_{K^{-1}}, P \models \xrightarrow{K} Q}{P \triangleleft X}$$

If  $P$  sees  $X$  encrypted with a private key  $K^{-1}$  and owns the corresponding public key  $K$ , then  $P$  is regarded as seeing  $X$ .

$$\text{SEE3} \quad \frac{P \triangleleft \{X\}_{@K}, P \models \xrightarrow{K@} P}{P \triangleleft X}$$

If  $P$  sees  $X$  encrypted with an encryption key  $@K$  of a key pair  $(@K, K@)$  and has the decryption key  $K@$ , then  $P$  is thought of as seeing  $X$ .

$$\text{SEE4} \quad \frac{P \triangleleft \{X\}_{\$K}, P \models \xrightarrow{\$K} P}{P \triangleleft X}$$

If  $P$  sees  $X$  encrypted with a known symmetric key  $\$K$ , then  $P$  sees  $X$ .

## C. Assumptions of access control solution

We refer principals to the Access Control Manager (ACM), readers and writers, which are presented by symbols  $M, R_1, \dots, R_n$  and  $W_1, \dots, W_n$ .  $KW_1, \dots, KW_n$  and  $KR_1, \dots, KR_n$  denote the public keys of writers  $W_1, \dots, W_n$  and readers  $R_1, \dots, R_n$  respectively.  $KW_1^{-1}, \dots, KW_n^{-1}$  and  $KR_1^{-1}, \dots, KR_n^{-1}$  represent the corresponding private keys.  $(@NK, NK@)$  is the asymmetric key pair produced by ACM  $M$ .  $\$DK$  and  $Data$  is the symmetric key and data created by writers.

To analyze the access control solution, we first list the following assumptions.

$$\text{A1: } M \setminus W_i \setminus R_i \models \xrightarrow{K_j} W_j \setminus R_j \quad \text{A2: } W_i \models \xrightarrow{\$DK} W_i$$

$$\text{A3: } M \models \xrightarrow{@NK} M \quad \text{A4: } M \models \xrightarrow{NK@} M$$

$$\text{A5: } W_i \setminus R_i \models M \Rightarrow (@NK, NK@) \quad \text{A6: } M \setminus R_i \models W_j \Rightarrow \$DK$$

$$\text{A7: } M \setminus R_i \models W_j \Rightarrow Data \quad \text{A8: } W_i \models \#(\$DK) \quad \text{A9: } W_i \models \#(Data)$$

$$\text{A10: } P \triangleleft NK@ \rightarrow P \models \#(NK@) \quad \text{A11: } P \triangleleft NK@ \rightarrow P \models \xrightarrow{NK@} P$$

$$\text{A12: } P \triangleleft (@NK, NK@) \rightarrow P \models \#(@NK, NK@)$$

$$\text{A13: } P \triangleleft (@NK, NK@) \rightarrow P \models \xrightarrow{@NK} P$$

$$\text{A14: } P \triangleleft (@NK, NK@) \rightarrow P \models \xrightarrow{NK@} P$$

$$\text{A15: } P \triangleleft (\$DK) \rightarrow P \models \#(\$DK) \quad \text{A16: } P \triangleleft (\$DK) \rightarrow P \models \xrightarrow{\$DK} P$$

$$\text{A17: } P \triangleleft (Data) \rightarrow P \models \#(Data)$$

Assumptions **A1-A4** are about the keys initially known to the principals. Assumptions **A5-A7** describe that  $M$  is trusted by  $W_i$  to make key pair  $(@NK, NK@)$  and  $M$  trusts that  $W_i$  can produce data key  $\$DK$  and data  $Data$ . Assumptions **A8** and **A9** indicate that  $W_i$  believes data  $Data$  and data key  $\$DK$  are fresh. Assumptions **A10-A16** present the situation when a principle sees different keys. The principle  $P$  could be replaced by  $M, R_i$  and  $W_i$ .  $P$  learns the related key which is also considered to be fresh. **A17** shows when  $P$  sees data  $Data$  it also confirms the freshness of it.

#### IV. ANALYSIS OF WRITE OPERATION IN A CLOSED ENVIRONMENT

In this section, we introduce the specific write operation in a closed environment. Then we apply our BAN-like logic to this procedure. Several security goals are listed to be proved.

##### A. Write operation in a closed environment

We give the write operation procedure in a closed environment according to Fig.1.(a) as below.

- Message 1.  $Writer_i \rightarrow ACM : name_{NK}$   
 Message 2.  $ACM \rightarrow Writer_i : name_{NK}, H(K_1)...H(K_n)$   
 Message 3.  $Writer_i \rightarrow ACM : name_{NK}, H(K_i)$   
 Message 4.  $ACM \rightarrow Writer_i : name_{NK}, H(K_i), \{NK_e, NK_d\}_{K_i}$   
 Message 5.  $Writer_i \rightarrow ACM : name_{Data}, \{Data\}_{DK}$   
 Message 6.  $ACM \rightarrow Writer_i : name_{DK}$   
 Message 7.  $Writer_i \rightarrow ACM : name_{DK}, \{DK\}_{NK_e}$

$Writer_i$  sends the name of key  $NK$  to  $ACM$ .  $ACM$  returns  $name_{NK}$  together with the hash values of  $K_1...K_n$ , denoted by  $H(K_1)...H(K_n)$ . Then  $Writer_i$  recognizes the correct hash values of  $K_i$  in the hash value sequences and feedbacks it to  $ACM$  with  $name_{NK}$ . After receiving the package,  $ACM$  returns a new message added with the corresponding key pair  $(NK_e, NK_d)$ .  $Writer_i$  uses its special data key  $DK$  to encrypt the data and sends it to  $ACM$  for storing.  $ACM$  can learn the name of data key  $DK$  and send it to  $Writer_i$ . Finally,  $Writer_i$  returns the message of  $DK$  encrypted by key  $NK_e$  with  $name_{DK}$ . This can lead  $ACM$  to acknowledge data key  $DK$ , which is used to decrypt message  $\{Data\}_{DK}$ .

##### B. Analysis of security goals of asymmetric key pair

In order to idealize the procedure, we abstract all the forwarding encrypted messages, modify the forms of keys, and change the expression of formulas in our BAN-like logic as below.

- M1:**  $W_i \triangleleft \{(@NK, NK@)\}_{KW_i}$   
**M2:**  $M \triangleleft \{Data\}_{SDK}$   
**M3:**  $M \triangleleft \{SDK\}_{@NK}$

We hope the procedure should archive several security goals when distributing  $(@NK, NK@)$ ,  $Data$  and  $SDK$ . Considering  $(@NK, NK@)$ , there are three authentication goals described in formulas:  $W_i \triangleleft (@NK, NK@)$ ,  $W_i \equiv M \equiv (@NK, NK@)$  and  $W_i \equiv (@NK, NK@)$ . These mean that  $W_i$  should see key pair  $(@NK, NK@)$ ,  $W_i$  also believes that  $M$  believes the key pair, and  $W_i$  believes the key pair respectively.

The first goal can be proved easily. Applying the seeing rule **SEE1** to message **M1** and assumption **A1** yields

$$\mathbf{S1:} W_i \triangleleft (@NK, NK@).$$

Now we hence proved that the procedure has achieved the first goal. But we cannot keep carrying forward, as the next two goals need more information. The crux of proving the two remaining goals is to achieve  $W_i \equiv M \equiv (@NK, NK@)$  using the message-meaning rule **MM2**. Hence, we need a pair of asymmetric keys for the application of this rule. We assume that  $M$  owns a public key  $K_M$  and a corresponding private key  $K_M^{-1}$  used for its signatures. As a result, we do the modification to the idealized procedure. Two new assumptions are added as below.

$$\mathbf{A'1:} W_i \equiv \xrightarrow{K_M} M$$

$$\mathbf{A'2:} M \equiv \xrightarrow{K_M} M$$

As the key pair  $(NK_e, NK_d)$  is produced by  $M$ , message **M1** is changed as below.

$$\mathbf{M1':} W_i \triangleleft \{ \{ @NK, NK@ \}_{K_M^{-1}} \}_{KW_i}$$

Again applying **SEE1** to message **M1'** and assumption **A1** yields

$$\mathbf{S2:} W_i \triangleleft \{ @NK, NK@ \}_{K_M^{-1}}.$$

Using the seeing rule **SEE2** to **S2** with assumption **A'1** also produces

$$\mathbf{S1:} W_i \triangleleft (@NK, NK@).$$

Then employing the message-meaning rule **MM2** to **S2** and assumption **A'1** gets

$$\mathbf{S3:} W_i \equiv M \sim (@NK, NK@).$$

As BAN logic defaults to using Modus Ponens (MP) rule, adopting MP to **S1** and assumption **A12** obtains

$$\mathbf{S4:} W_i \equiv \#(@NK, NK@).$$

Utilizing the nonce-verification rule **NV** to **S3** and **S4** obtains

$$\mathbf{S5:} W_i \equiv M \equiv (@NK, NK@).$$

Applying the jurisdiction rule **J** to assumption **A5** and **S5** yields

$$\mathbf{S6:} W_i \equiv (@NK, NK@).$$

We can conclude that the original formulas can only archive the first goal  $W_i \triangleleft (NK_e, NK_d)$ . After our modification, the next two goals  $W_i \equiv M \equiv (NK_e, NK_d)$  and  $W_i \equiv (NK_e, NK_d)$  can also be satisfied. Hence, in the implementation of the access control solution in Fig.1 (step a.4), the data domain of  $Data$  packet should be changed from  $\{(NK_{encryption}, NK_{decryption})\}_{K_{pub-writer}}$  to  $\{ \{ (NK_{encryption}, NK_{decryption}) \}_{K_M^{-1}} \}_{K_{pub-writer}}$ .

##### C. Analysis of security goals of data key

The first series of goals should be archived are about data key  $SDK$ :  $M \triangleleft SDK$ ,  $M \equiv W_i \equiv SDK$  and  $M \equiv SDK$ .

Applying the seeing rule **SEE4** to message **M3** and assumption **A5** yields

$$\mathbf{S7:} M \triangleleft SDK.$$

In order to push further, we need to obtain the formula  $M \equiv W_i \sim SDK$  concerning with the message-meaning rule **MM2**. As a result, we need to utilize the private key  $KW_i^{-1}$  to encrypt  $SDK$  first. We also make a little change to the idealized procedure. Message **M3** is altered as below.

$$\mathbf{M3':} M \triangleleft \{ \{ SDK \}_{KW_i^{-1}} \}_{@NK}$$

Utilizing the seeing rule **SEE3** to message **M3'** and assumption **A4** gains

$$\mathbf{S8:} M \triangleleft \{ SDK \}_{KW_i^{-1}}.$$

Using the seeing rule **SEE3** to **S8** with assumption **A1** produces

$$\mathbf{S7:} M \triangleleft SDK.$$

Applying the message-meaning rule **MM2** to **S8** and assumption **A1** yields

$$\mathbf{S9:} M \equiv W_i \sim SDK.$$

Adopting the MP rule to **S7** and assumption **A15** gets

$$\mathbf{S10:} M \equiv \#(SDK).$$

Utilizing the nonce-verification rule **NV** to **S9** and **S10** gains

$$\mathbf{S11}: M \equiv W_i \equiv \$DK.$$

Applying the jurisdiction rule **J** to **S11** and assumption **A6** yields

$$\mathbf{S12}: M \equiv \$DK.$$

By this point, we have figured out the the formal proof of three goals for data key  $DK$ . As a result, in the implementation of the access control solution in Fig.1 (step a.7), the data domain of  $Data$  packet should be changed from  $\{DK\}NK_{encryption}$  to  $\{\{DK\}K_{pri-writer}\}NK_{encryption}$ .

#### D. Analysis of security goals of data

We focus on the other three goals for  $Data$  which represents the real data the writer wants to publish. They are  $M \triangleleft Data$ ,  $M \equiv W_i \equiv Data$  and  $M \equiv Data$ . These denote that  $M$  should see  $Data$ ,  $M$  also believes that  $W_i$  believes  $Data$ , and  $M$  believes  $Data$  respectively.

Similarly, we can easily apply the MP rule to assumption **A15** and **S7** to get

$$\mathbf{S13}: M \equiv \xrightarrow{\$DK} M.$$

The seeing rule is also adopted to **S13** and message **M2** to gain

$$\mathbf{S14}: M \triangleleft Data.$$

For further proof, we are required to gain the formula  $M \equiv W_i \sim Data$  springing from the message-meaning rule **MM2**. So we choose to apply the private key  $KW_i^{-1}$  to encrypt  $\$Data$  first. Message **M2** is changed as below.

$$\mathbf{M2}': M \triangleleft \{\{Data\}_{KW_i^{-1}}\}\$DK$$

Utilizing the seeing rule **SEE4** to message **M2'** and **S13** gains

$$\mathbf{S15}: M \triangleleft \{Data\}_{KW_i^{-1}}.$$

Applying the seeing rule **SEE2** to **S15** and assumption **A1** yields

$$\mathbf{S14}: M \triangleleft Data.$$

Employing the message-meaning rule **MM2** to **S15** according to assumption **A1** gets

$$\mathbf{S16}: M \equiv W_i \sim Data.$$

Using the MP rule to **S14** with assumption **A16** produces

$$\mathbf{S17}: M \equiv \#(Data)$$

Applying the nonce-verification rule **NV** to **S16** and **S17** yields

$$\mathbf{S18}: M \equiv W_i \equiv Data.$$

Adopting the jurisdiction rule **J** to assumption **A7** and **S18** gets

$$\mathbf{S19}: M \equiv Data.$$

Now we have settled the formal proof of the three goals of data  $Data$ . As a result, in the implement of the access control solution in Fig.1 (step a.5), the data domain of  $Data$  packet should be changed from  $\{Data\}DK$  to  $\{\{Data\}K_{pri-writer}\}DK$ .

## V. ANALYSIS OF READ OPERATION IN A CLOSED ENVIRONMENT

In this section, the specific read operation in a closed environment is presented. We adopt our BAN-like logic to this procedure and analyze it with some important security goals.

### A. Read operation in a closed environment

We demonstrate the read operation procedure in a closed environment according to Fig.1.(b) as below.

Message 1.  $Reader_i \rightarrow ACM : name_{Data}$

Message 2.  $ACM \rightarrow Reader_i : name_{Data}, \{Data\}_{DK}$

Message 3.  $Reader_i \rightarrow ACM : name_{DK}$

Message 4.  $ACM \rightarrow Reader_i : name_{DK}, \{DK\}_{NK_e}$

Message 5.  $Reader_i \rightarrow ACM : name_{NK}$

Message 6.  $ACM \rightarrow Reader_i : name_{NK}, H(K_1)...H(K_n)$

Message 7.  $Reader_i \rightarrow ACM : name_{NK}, H(K_i)$

Message 8.  $ACM \rightarrow Reader_i : name_{NK}, H(K_i), \{NK_d\}_{K_i}$

$Reader_i$  sends the name of  $Data$  to  $ACM$ .  $ACM$  returns  $Data$  encrypted with data key  $DK$ . In order to decrypt it,  $Reader_i$  sends  $name_{DK}$  to  $ACM$  to request for data key  $DK$ .  $ACM$  gives  $Reader_i$  with  $DK$  encrypted with key  $NK_e$ . As a result,  $Reader_i$  still needs to deliver  $name_{NK}$  to get decryption key  $NK_d$ .  $ACM$  returns  $name_{NK}$  together with the hash values of  $K_1...K_N$ , denoted by  $H(K_1)...H(K_N)$ . Then  $Reader_i$  recognizes the correct hash values of  $K_i$  in the hash value sequences and feeds it back to  $ACM$  with  $name_{NK}$ . Finally,  $ACM$  returns a new message added with  $NK_d$  encrypted with public key  $K_i$ .  $NK_d$  is the corresponding decryption key of encryption key  $NK_e$ , which is applied to decrypt  $\{DK\}_{NK_e}$ .

### B. Analysis of security goals of data

We also idealize the read operation procedure as below.

$$\mathbf{M4}: R_i \triangleleft \{Data\}_{\$DK}$$

$$\mathbf{M5}: R_i \triangleleft \{\$DK\}_{@NK}$$

$$\mathbf{M6}: R_i \triangleleft \{NK@\}_{KR_i}$$

Three security goals described in BAN formulas need to be proved:  $R_i \triangleleft Data$ ,  $R_i \equiv W_j \equiv Data$  and  $R_i \equiv Data$ .

Applying the seeing rule **SEE2** to message **M6** and assumption **A1** yields

$$\mathbf{S20}: R_i \triangleleft NK@.$$

Using Modus Ponens (MP) rule to **S20** with assumption **A9** produces

$$\mathbf{S21}: R_i \equiv \xrightarrow{NK@} R_i.$$

Utilizing the seeing rule **SEE3** to message **M5** and **S21** gains

$$\mathbf{S22}: R_i \triangleleft \$DK.$$

Employing the Modus Ponens (MP) rule to **S22** and assumption **A15** gets

$$\mathbf{S23}: R_i \equiv \xrightarrow{\$DK} R_i.$$

Applying the seeing rule **SEE4** to message **M4** and **S23** yields

$$\mathbf{S24}: R_i \triangleleft Data.$$

Now the first goal has been proved. To prove the remaining goals, we also change message **M4** slightly. The modification reason is similar with the one in Section IV-D. Supposing that the  $Data$  is produced by writer  $W_j$ , it first is encrypted by  $W_j$ 's private key  $KW_j^{-1}$ .

$$\mathbf{M4}': R_i \triangleleft \{\{Data\}_{KW_j^{-1}}\}\$DK$$

Adopting the seeing rule **SEE4** to message **M4'** and **S23** produces

$$\mathbf{S25}: R_i \triangleleft \{Data\}_{KW_j^{-1}}.$$

Applying the seeing rule **SEE2** to **S25** and assumption **A1** yields

$$\mathbf{S24}: R_i \triangleleft Data.$$

Utilizing the Modus Ponens (MP) rule to **S24** and assumption **A16** gains

$$\mathbf{S26}: R_i \equiv \#(Data).$$

Using the message-meaning rule **MM2** to **S25** with assumption **A1** produces

$$\mathbf{S27}: R_i \equiv W_j \sim Data.$$

Employing the nonce-verification rule **NV** to **S26** and **S27** gets

$$\mathbf{S28}: R_i \equiv W_j \equiv Data.$$

Applying the jurisdiction rule **J** to **S28** and assumption **A6** yields

$$\mathbf{S29}: R_i \equiv Data.$$

So far, we have proved three goals of data *Data*. So in the implement of the access control solution in Fig.1 (step b.2), the data domain of *Data* packet should be changed from  $\{Data\}DK$  to  $\{\{Data\}K_{pri-writer}\}DK$ .

## VI. ANALYSIS OF WRITE AND READ OPERATION IN AN OPEN ENVIRONMENT

In this section, we show the specific read and write operation in an open environment. The BAN-like logic is applied to them. Then we analyze them with some important security goals.

### A. Write operation in an open environment

Similar with Section IV-A, write operation procedure in an open environment in 1.(c) can be idealized as blew.

$$\mathbf{M1}': W \triangleleft \{ @NK, NK@ \}_{KW_w}$$

$$\mathbf{M2}': M \triangleleft \{ Data \}_{SDK}$$

$$\mathbf{M3}': M \triangleleft \{ SDK \}_{@NK}$$

As *ACM* can figure out key  $K_w$  when it received the hash value of it, we can infer that

$$\mathbf{A'3}: M \equiv \xrightarrow{KW_w} W.$$

There are six goals needing to be proved:  $W \triangleleft (@NK, NK@)$ ,  $W \equiv M \equiv (@NK, NK@)$ ,  $W \equiv (@NK, NK@)$ ,  $M \triangleleft Data$ ,  $M \equiv W \equiv Data$  and  $M \equiv Data$ . With the help of assumption **A'3**, we can also prove them in almost the same way as we do in Section IV.

### B. Read operation in an open environment

Like Section V-A, read operation procedure in an open environment in 1.(d) can be idealized as blew.

$$\mathbf{M4}': R \triangleleft \{ Data \}_{SDK}$$

$$\mathbf{M5}': R \triangleleft \{ SDK \}_{@NK}$$

$$\mathbf{M6}': R \triangleleft \{ NK@ \}_{K_r}$$

We can also deduce a new assumption.

$$\mathbf{A'4}: M \equiv \xrightarrow{K_r} R$$

Three goals should be verified:  $R \triangleleft Data$ ,  $R \equiv W_j \equiv Data$  and  $R \equiv Data$ . With assumption **A'4**, we can demonstrate these goals using the similar way in Section V.

## VII. CONCLUSION AND FUTURE WORK

In this paper, we described a new approach to reasoning about an access control solution of NDN. Our work, which is inspired by the BAN logic, is a special logic to analyze different operation procedures in the access control solution. We idealized the procedures using the BAN-like logic and set several security goals to analyze them. These unsatisfied goals indicate that this access control solution cannot ensure the source of critical keys and data. This may lead the access control solution to be vulnerable to intruders. Then we did some improvement for the unsatisfied goals and made the new access control solution archive the important security goals. This method leads us in identifying mistakes and suggesting corrections. Our study helps to improve security and protect against various attacks for the access control solution. As for the future work, we would like to apply this method to other access control solutions of NDN.

**Acknowledgement.** This work was partly supported by Shanghai Collaborative Innovation Center of Trustworthy Software for Internet of Things (No. ZF1213).

## REFERENCES

- [1] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, K. Claffy, D. Krioukov, D. Massey, C. Papadopoulos, T. Abdelzaher, L. Wang, and E. Yeh, "Named data networking (NDN) project," PARC, Tech. Rep. NDN-0001, 2010.
- [2] T. Chen, K. Lei, and K. Xu, "An encryption and probability based access control model for named data networking," in *IEEE 33rd International Performance Computing and Communications Conference, IPCCC 2014, Austin, TX, USA, December 5-7, 2014*, 2014, pp. 1–8.
- [3] J. Golle and D. Smetters, "Ccnx access control specifications," Xerox Palo Alto Research Center-PARC, Tech. Rep., 2010.
- [4] B. Hamdane, R. Boussada, M. E. Elhdhili, and S. G. E. Fatmi, "Towards a secure access to content in named data networking," in *26th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE 2017, Poznan, Poland, June 21-23, 2017*, 2017, pp. 250–255.
- [5] M. Burrows, M. Abadi, and R. M. Needham, "A logic of authentication," *ACM Trans. Comput. Syst.*, vol. 8, no. 1, pp. 18–36, 1990.
- [6] K. Gaarder and E. Sneekenes, "On the formal analysis of PKCS authentication protocols," in *Advances in Cryptology - AUSCRYPT '90, International Conference on Cryptology, Sydney, Australia, January 8-11, 1990, Proceedings*, 1990, pp. 106–121.
- [7] L. Gong, R. M. Needham, and R. Yahalom, "Reasoning about belief in cryptographic protocols," in *Proceedings of the 1990 IEEE Symposium on Security and Privacy, Oakland, California, USA, May 7-9, 1990*, 1990, pp. 234–248.
- [8] A. D. Rubin and P. Honeyman, "Nonmonotonic cryptographic protocols," in *Seventh IEEE Computer Security Foundations Workshop - CSFW'94, Franconia, New Hampshire, USA, June 14-16, 1994, Proceedings*, 1994, pp. 100–116.

# Re-checking App Behavior against App Description in the Context of Third-party Libraries

Chengpeng Zhang<sup>1</sup>, Haoyu Wang<sup>1,2\*</sup>, Ran Wang<sup>1</sup>, Yao Guo<sup>3</sup>, Guoai Xu<sup>1\*</sup>

<sup>1</sup> Beijing University of Posts and Telecommunications, Beijing, China, 100876

<sup>2</sup> Beijing Key Lab of Intelligent Telecommunication Software and Multimedia

<sup>3</sup> Peking University, Beijing, China, 100871

Email: {buptkick, haoyuwang, wangran2015, xga}@bupt.edu.cn, yaoguo@pku.edu.cn

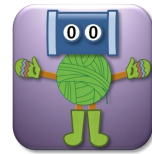
**Abstract**—Recent research suggested promising approaches that identify potential malware by checking the inconsistency between app description and actual behavior of the app. However, state-of-the-art approaches have ignored the impact of third-party libraries (TPLs) when detecting outliers, which could affect the detection results greatly in two folds. On one hand, most Android apps would not list the functionality of TPLs in app description, which could cause false positives, as many apps that use TPLs will be identified as outliers. On the other hand, it is important to separate TPLs from custom code when analyzing the sensitive behaviors, otherwise the malicious behaviors of custom code will be obscured by TPLs. In this paper, we revisit the study of checking app behavior against app description in the context of TPLs. Experiment results on more than 400K Android apps suggest that more than 54% of apps are no longer identified as outliers after filtering TPLs, and we could identify roughly 50% of new outliers. Furthermore, removing the impact of TPLs could help to identify malware and pinpoint the malicious behavior of custom code. Our results shed a light on applying the TPL analysis to enhance a variety of mobile app analysis tasks.

## I. INTRODUCTION

Mobile malware is rapidly becoming a serious threat in recent years. The number of Android malware has risen steadily. Recent report [1] shows that the number of Android malware achieves almost 3.5 million in 2017, which has affected billions of devices.

A wide range of research has thus proposed approaches to analyze and detect malware, which could be roughly categorized into static analysis methods [2]–[5] and dynamic analysis methods [6], [7]. However, these approaches are usually not help against new malware families [8], as in many cases, it is difficult to differentiate between malware and benign apps because they may share the same/similar sensitive behaviors. For example, automated tools might detect that a flashlight app and a map app both use users' location information, while the map app is more legitimate than the flashlight app from the users' perspective. As a result, the difference between the users' expectations and the actual behavior of the app should be an important indicator for detecting a malicious app.

Recent research suggested promising approaches that identify potential malware/outliers by checking whether it behaves



Knitting and Crochet Buddy

### Sensitive APIs used in TPLs:

LocationManager.getLastKnownLocation()  
TelephonyManager.getCellLocation()

### Sensitive APIs used in custom code:

None

(1) Sensitive behaviors used in TPLs would lead to outlier apps



YaYa Globe

### Sensitive APIs used in TPLs:

TelephonyManager->getLine1Number  
TelephonyManager->getSimSerialNumber  
LocationManager->getLastKnownLocation

### Sensitive APIs used in custom code:

TelephonyManager->getLine1Number  
TelephonyManager->getSimSerialNumber  
LocationManager->getLastKnownLocation

(2) Sensitive behaviors used in TPLs would hide the behaviors of custom code

Fig. 1. Motivating Examples

as advertised [9]–[11]. For example, WHYPER [9] and AutoCog [10] use natural language processing (NLP) techniques to infer the apps expected behaviors from app descriptions, and compare with the actual behavior extracted from the requested permissions. CHABADA [11] uses Latent Dirichlet Allocation (LDA) on app descriptions to identify the main topics of each app, and then clusters apps based on related topics. By extracting sensitive APIs used for each app, it can identify outliers which use APIs that are uncommon for that cluster. For example, for a group of 20 wallpaper apps, the app that has uncommon behaviors (e.g., access location and contacts) is probably up to malware.

Our work is motivated by CHABADA [11]. Although their experiment results are promising, they have ignored an important issue: *the impact of third-party libraries (TPLs)*. We argue that TPLs should be considered separately when checking app behavior against app description for two reasons.

First, TPLs (e.g., ad library, third-party analytics, social networking libraries, etc.) are widely used in Android apps. Previous work [12] suggested that more than 60% of the code in Android apps belongs to third-party libraries on average. However, **the use of TPLs could affect the outliers detection due to the reason that most Android apps would not list the functionalities of TPLs in their descriptions.** In our initial experiment on 100 popular Android apps, we found that

\*Co-corresponding authors

DOI reference number: 10.18293/SEKE2018-180



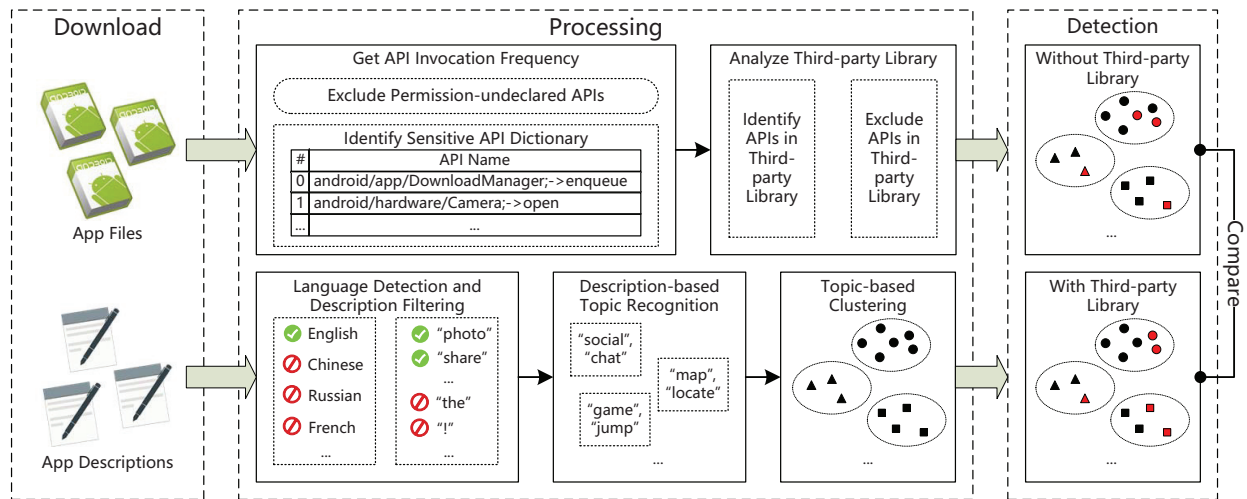


Fig. 2. Study Methodology

95% of them do not explicit show the functionalities of TPLs in app descriptions. For instance, as shown in Figure 1 (1), the app (androiddeveloperjoe.knittingbuddy) uses the Amazon Ads and InMobi library to display ads. It was identified as outliers because several sensitive APIs introduced by TPLs, although there are no sensitive behaviors exist in its main code.

Second, we argue that **it is important to separate TPLs from app custom code, which could be then used to determine whether the sensitive behaviors are introduced by custom code or not.** Otherwise, the malicious behaviors of core functionalities will be obscured by TPLs. For example, for a group of 20 wallpaper apps, the malicious app that tracks user’s location information in its main code will be identified as normal app if most of these apps embed advertisement library that would access location data for targeted advertising. Indeed, more than 70% of popular apps in Google Play use advertising libraries [13]. The purposes of sensitive behaviors used in custom code and TPLs are usually different [14], [15]. As shown in Figure 1 (2), the sensitive behaviors used in TPLs would dilute the behaviors of custom code.

It is worth noting that some TPLs also show aggressive and malicious behaviors, as reported by many previous work [16]–[19]. However, the key idea of this paper is to separate TPLs from app custom code and determine whether the sensitive behaviors are introduced by custom code. The malicious behaviors introduced by app developers in the core functionalities of the app will be more obvious if we remove the noisy introduced by TPLs.

Hence, in this paper, we revisit the study of checking app behavior against app description in the context of TPLs. We take the first step to examine the impact of TPLs in pinpointing outlier apps. Experiment results on more than 400k apps (an order of magnitude higher than CHABADA) from Google Play suggest that TPLs introduce great impact on the outlier detection of Android apps, with more than 54% of apps are no longer identified as outliers after excluding TPLs. Besides, we could identify more than 50% of new outliers

because *removing the impact of TPLs could sharp the sensitive behaviors of the main functionalities.*

## II. STUDY METHODOLOGY

### A. Crawling the dataset

We have downloaded more than 400 thousand free Android apps from Google Play in March, 2015. We crawled the meta-data of these apps, including the app names, app categories, app ratings, the number of installs, etc. We also downloaded all the apk files of these apps.

### B. Overall Architecture

The overall architecture of our study is shown in Figure 2. For the crawled apps, we first apply natural language processing (NLP) techniques to the app descriptions for filtering and stemming. Then we take advantage of LDA to identify topics from app descriptions, and each app is represented as a topic vector. Apps with similar topic vectors will be grouped in the same cluster. For apps in the same cluster, we extract sensitive APIs used in them. Apps with unusual API used patterns will be labeled as outliers. Note that to explore the impact of TPLs on the outliers detection, we use a clustering-based approach to identify TPLs used in these apps and compare the results with and without TPLs.

### C. Description-based Clustering

1) *App Description Preprocessing:* Considering that a too short description cannot represent the functionality of the app well, we first exclude the apps with the length of descriptions less than 10 words in our dataset. Then we use the language-detector [20] tool to detect the language of app description. Note that we only consider the apps whose descriptions are written in English in this work. Take advantage of Mallet [21], we build a list to filter out stop-words. Then we use the Snowball [22] to turn the words into stem form. At last, after app description preprocessing, we have 276,333 app samples left in the dataset.

TABLE I  
THE 30 TOPICS EXTRACTED FROM THE 276,333 APP DESCRIPTIONS

#	Topic Name	Topic Keywords
0	sociality	share facebook twitter social chat app friend messag email send network post love peopl free photo creat sms featur easi
1	picture	pictur photo imag camera beauti make color design fashion galleri dress girl hair style effect frame choos creat app share
2	religion	church god prayer bibl christian book chapter app india indian islam lord read audio listen quran vers holi hindi day
3	racing	race car game speed play free slot drive win machin coin real simul truck park excit fun featur spin experi
4	workout	workout app weight number calcul exercis time train fit bodi measur result unit simpl convert track math system program base
5	shopping	shop servic offer search find book app store price order deal featur product locat custom direct inform home special mobil
6	child	child kid babi children game learn fun play anim app color draw pictur cat dog pet educ age free sound
7	jumping	play game jump run level shoot control fli fun enemi score tap collect challeng featur bird zombi power avoid world
8	sport	sport team score footbal player club world golf soccer countri leagu match app game live cup play includ flag fan
9	geography	citi nation state art year south area counti north west american countri includ san world award histori cultur york local
10	theme	theme icon font keyboard launcher instal download app appli set free select screen widget home press android function phone support
11	video	video mobil watch app movi free download content youtub connect updat channel stream copyright offici disclaim latest devic share
12	scenery	beauti natur enjoy fish water christma magic world tree sea room flower beach night day time light blue hous halloween
13	finance	money account mobil manag app bank credit check pay bill view payment onlin access secur transact transfer balanc servic inform
14	cooking	cook food recip make easi eat drink kitchen step fruit cake ingredi app love delici healthi cream ice chicken restaur
15	business	busi compani manag servic provid product develop market custom client industri experi profession design technolog work job offer
16	audio	audio music song radio listen play record app download station stream artist lyric free favorit rock danc player live featur
17	life	life time peopl make thing good day work person start love feel give tip find learn mind fact back don't
18	mobile	mobil phone call android devic app applic contact connect number messag user sms network wifi control data send password secur
19	calendar	calendar event inform school schedul app mobil view student class date access featur offici news confer connect find updat communiti
20	accounting	list real properti applic estat inform licens agreement licensor provid term data termin sale user mls servic state relat right
21	ringtone	rington sound quot app free applic android phone alarm set funni friend make fun joke inspir sleep effect download notifi
22	dictionary	dictionari word english learn transl languag studi app test question answer chines letter quiz text search spanish french phrase featur
23	system	set button time screen widget app tap click mode display press select devic start phone batteri light version chang featur
24	puzzle	puzzl game play score level point fun time challeng match mode player card number free move block simpl bubbl ball
25	news	news read latest updat magazin app issu star content articl inform access stori world featur free download subscript www page
26	wallpaper	wallpap background live imag set free screen galaxi phone app applic devic support download home pictur anim samsung tablet
27	health	healt medich app treatment inform mortgag care help patient doctor profession complet emerg diseas time featur easi access free
28	map	map citi locat app guid inform travel hotel place find search navig weather gps tour rout time restaur featur trip
29	documenter	file manag android app version devic support applic note data featur search user list creat card googl record save code

2) *Identifying Topics from App Descriptions:* A “topic” consists of a cluster of words that frequently occur together in app descriptions. To identify sets of topics, we resort to topic modeling using LDA based on Mallet [21]. Note that we choose the same number of topics (n=30) to be identified by LDA as CHABADA [11] for comparison. Table I shows the result of 30 topics (with top 20 keywords of each topic) that we have identified from the 276K app descriptions. Note that the “topic name” in the second column is the abstract concept we manually assigned to that topic, which is a meaningful word that can represent the corresponding topic. As a result, each app is represented as a topic vector, and each dimension of the topic vector represents the probability that an app belongs to the corresponding topic.

3) *App Clustering:* To identify the clusters of apps with similar descriptions, we take advantage of K-means++ algorithm<sup>1</sup> for app clustering based on the topic vectors.

Note that the K-means++ algorithm needs to be given either some initial potential centroids, or the number K of clusters to identify. Thus one challenge here is to identify the number of clusters that should be created. One straight-forward idea is to run the algorithm multiple times and each time with a different K number. Based on a set of clustering results, we would then be able to identify the best one.

We propose to use Genetic Algorithm (GA) [23] combined with K-means++ to determine the best number of clusters,

which was shown to be effective in previous work [24]. GA is suitable to this problem space because the selection, crossover and mutation steps of GA could help to choose the optimal value of K. With the generations evolving, the better individuals with higher fitness values will emerge [24]. As a result, take advantage of GA, we will get the best value of K.

We used the silhouette coefficient<sup>2</sup> as the fitness value to measure the effectiveness of clustering results. Note that each cluster is represented by a silhouette coefficient, which is based on the comparison of its tightness and separation. This silhouette coefficient shows how closely each element is matched to the other elements within its cluster, and how loosely it is matched to other elements of the neighboring clusters. The range of the value of silhouette coefficient is -1 to 1. When the value of the silhouette coefficient of an element is close to 1, it means that the element is in the appropriate cluster. Thus, we compute the average of the silhouette coefficient for each solution using K as the number of clusters, and we select the solution whose silhouette coefficient was closest to 1.

In this paper, we use the scikit-learn [25] to implement K-means++ algorithm and the calculation of silhouette coefficient. The implementation of GA is based on Pyevolve [26], an evolutionary computation framework. As a result, to achieve the highest silhouette coefficient on average, we choose 29 as the best number of clusters should be created. The clustering result for the 267,333 apps we analyzed is listed in Table II.

<sup>1</sup><https://en.wikipedia.org/wiki/K-means%2B%2B>

<sup>2</sup>[https://en.wikipedia.org/wiki/Silhouette\\_\(clustering\)](https://en.wikipedia.org/wiki/Silhouette_(clustering))

TABLE II  
THE RESULT OF APP CLUSTERING.

#	# Apps	Most Important Topics
0	5825	sport(94.2%), puzzle, jumping
1	11791	business(95.3%), accounting, finance
2	11508	shopping(98.5%), cooking, picture
3	8660	workout(98.6%), finance, racing
4	4711	cooking(99.8%), geography, theme
5	7967	audio(96.9%), geography, video
6	5079	racing(99.8%), jumping, geography
7	5800	health(98.3%), geography, mobile
8	7853	dictionary(99.4%), theme, puzzle
9	18498	puzzle(99.2%), jumping, racing
10	10449	mobile(99.5%), finance, theme
11	5817	ringtone(97.9%), jumping, wallpaper
12	10084	calendar(99.5%), geography, finance
13	6567	finance(100.0%)
14	12485	documenter(95.6%), mobile, finance
15	7025	scenery(78.2%), jumping(5.4%), wallpaper(3.7%)
16	5954	theme(100.0%)
17	10791	system(95.7%), theme, wallpaper
18	9084	map(98.8%), geography, calendar
19	7952	picture(99.7%), wallpaper, geography
20	8539	sociality(94.3%), mobile, picture
21	17604	jumping(99.9%), geography, wallpaper
22	28138	geography(20.7%), accounting(5.5%), jumping(4.8%)
23	6410	religion(98.5%), geography, wallpaper
24	8194	child(97.0%), puzzle, picture
25	7088	video(99.4%), jumping, wallpaper
26	8069	news(95.4%), geography, video
27	9518	life(88.8%), jumping, cooking
28	8873	wallpaper(99.8%), theme, scenery

Each cluster contains apps whose descriptions contain similar topics, as shown in Column Most Important Topics. The percentages reported in the last column represent the weight of the dominant topic within each cluster.

#### D. App Processing

1) *Identifying the Sensitive APIs*: To identify outliers regarding their actual behavior, we need to identify the sensitive APIs used in each app. CHABADA uses a set of sensitive APIs derived from STOWAWAY [27]. The dataset was published in 2011 and it contains a list of sensitive APIs of Android version 2.2, which is outdated and incomplete. In this work, We use a list of permission-related APIs from PScout [28], which contains 680 sensitive APIs.

As shown in previous work [29], APIs with the same name but different parameters always share the similar functionalities. Thus we only take account of the name of the API, i.e., the APIs with the different parameters but the same name are identified as the same API. At last, there are only 428 APIs left from the 680 permission-sensitive APIs.

For each app, we generate an API feature vector where each dimension represents the invocation frequency for the corresponding API. Note the previous work [30] suggested that there are unreachable APIs in Android apps, i.e., these apps do not declare the corresponding permissions. Thus, we extract the declared permissions for the manifest of each app and filter the APIs that use undeclared permissions.

2) *Identifying Third-party Libraries*: One key idea in this paper is to measure the impact of third-party libraries in description-based outlier detection. Thus, we first need to

identify the code that belongs to third-party libraries, and then we generate the API feature vector for each app with and without third-party libraries respectively.

In this paper, we have implemented a clustering-based approach which is shown to be effective in LibRadar [31]–[33] to identify the common frameworks used in apps. We extract the API call features at the package level and then we enforce strict comparison here to cluster all the features into groups, which means that only when the features of two packages are exactly the same can they be clustered. We choose the 128 as the threshold to identify third-party libraries, which means that as long as the package has occurred in more than 128 apps, we will regard it as the common library. With this threshold, we are able to find that roughly 70% of the code belongs to TPLs. Then we check the usage of sensitive APIs in TPLs to generate the feature vectors without TPLs as comparison.

#### E. Outliers Detection

In this paper, we use the Isolation Forest [34] algorithm to identify outliers, which was shown to be effective in identifying anomalous points in a large number of data. Besides, Isolation Forest algorithm has low memory requirements and linear time complexity, so it is more suitable for processing high-dimensional data. Note that the Isolation Forest algorithm needs two important parameters, one is the amount of trees, the other one is the amount of samples. In our study, we use the default value of sklearn, i.e., the amount of trees is set as 100 and the amount of samples is set as 256.

### III. EVALUATION

#### A. The Impact of Third-party Libraries

First, we want to answer the research question: *what is the impact of TPLs in pinpointing outliers?* For this purpose, we compare the results of outliers detection using Isolation Forest algorithm with and without TPLs respectively. The result of outliers detection is shown in Table IV.

Surprisingly, it turns out that only 721 (46.5%) apps are still identified as outliers while excluding TPLs. Moreover, 723 apps (50.1%) are newly identified as outliers after excluding TPLs in our experiment. Thus we further analyzed the reasons that leading to the inconsistency between the results.

1) *“False Positive” Outliers*: For the apps that are no longer identified as anomalies after excluding TPLs, we randomly picked 5 apps for each cluster (145 apps in total) and manually inspect the code for understanding the reason.

At a result, we found that almost all these apps belong to the reason that *TPLs have introduced sensitive APIs that are not rarely used in the custom code*. For example, the app named “Puzzle Code” (with package name “com.yiqusoft.puzzle”) uses two ad libraries named “domob” and “adsmogo”. These two ad libraries use several sensitive APIs (e.g., “getLastKnownLocation”) to display customized ads. However, these sensitive APIs are never invoked in the custom code. Thus, this app was identified as outliers without filtering TPLs.

TABLE III  
THE RESULT OF MANUALLY ANALYSIS.

Cluster No.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	Total
With TPLs	1	0	2	1	2	6	0	6	1	2	3	6	1	2	0	0	9	2	0	7	3	2	2	6	2	0	0	6	9	81
W/O TPLs	2	0	4	4	2	10	2	10	2	4	4	6	2	4	0	0	10	2	0	8	4	4	4	10	4	0	0	10	10	122

TABLE IV  
THE RESULT OF OUTLIERS DETECTION.

	# Outliers	# Common	% Percent
Before	1,551	721	46.5%
After	1,444	721	49.9%

2) “False Negative” Outliers: For the apps that are newly identified as outliers, we randomly picked 5 apps for each cluster (a total number of 145 apps) and manually inspect the code to explore the reason.

We found that the anomalous behaviors introduced by app developers in the core functionalities of the app will be more sharp if we remove the impact of TPLs. For instance, the app “com.appgame7.fruitsbreak” uses a certain amount of sensitive APIs to collect user’s phone number and location information in both custom code and TPLs. Most of the apps in the same cluster have embedded the ad libraries that share similar sensitive behaviors. The malicious behaviors in the custom code will be more obvious after removing the impact of TPLs.

#### B. The Behaviors of Outliers

We further explore the research question: *whether our approach could be used to identify malicious behaviors of Android apps?* For this purpose, we first manually inspect the top outliers as produced by our technique and classified them as malicious or normal. Then we upload all the identified outliers (2,274 apps in total) to VirusTotal<sup>3</sup> to check how many of them are flagged as malicious.

1) *Manually Inspection*: Only human can interpret properly what is in an app description. Therefore, we manually inspect the top 10 outliers for each cluster (290 apps in total). We first examine their descriptions, the list of sensitive APIs used, and the corresponding decompiled code. Then we install them on a real smartphone (nexus 5) and manually test it. Here we use TCPDump<sup>4</sup> to collect and analyze the network traffic.

Note that some outliers may not be malicious due to the inadequate descriptions. We would classify an app as malicious if: (1) the app collects user’s privacy information (e.g., phone number and location) and uploads it to a remote server without explicit advertise the sensitive behaviors either in its description or in its privacy policy. For example, the app “com.appblast.popclock” is a simple alarm clock app, whereas it collects user’s phone number and uploads it to a remote server. Another Android app “com.yoursite.lockfingerscanner” claims that it is able to achieve finger unlock function, whereas it is a grayware that only contains download links of other apps. (2) the app is reported as malware by the pre-installed

anti-virus tool<sup>5</sup> if we install it on the smartphone. For example, after the app named “TouchNPaint” (with package name “game.child.paint”) is installed, the AVL engine will report it as malware with family name “a.gray.mfpad”. The result of manually analysis is shown in Table III. After eliminating TPLs, we could identify more malicious outlier apps.

TABLE V  
DETECTION RESULT OF VIRUSTOTAL.

	“False Positive” Apps	“False Negative” Apps	Common
Flagged by VT	205	359	434
Total	830	723	721
Percentage	24.70%	49.65%	60.19%

2) *Detection Result of VirusTotal*: We then upload all the identified outliers (with and without TPLs, 2,274 apps in total) to VirusTotal to check how many of them are flagged by current anti-virus engines. As shown in Table V, 639 apps are labeled as malicious by at least one engines before filtering TPLs, and the number rised to 793 apps if we removed the impact of TPLs. It is also interesting to note that, although roughly 25% of the “false positive” apps are flagged by VirusTotal, most of them are labeled as “AdWare”, which means that the malicious behaviors are introduced by ad libraries that embedded in the app. This result suggests that *removing the impact of TPLs could help to better pinpoint the malicious behaviors of custom code.*

#### IV. DISCUSSION

In this paper, we revisit the study of checking app behavior against app description in the context of TPLs. We use several heuristic methods similar with that used in CHABADA [11] to identify the topics and clusters, which could be potentially improved. Moreover, we mostly rely on manually efforts to analyze the results, which may exist bias.

Our study suggests that TPLs play an important role in mobile app analysis, which could be potentially used to enhance a variety of mobile app analysis tasks, e.g., malware detection and app clone analysis [12], [35].

#### V. RELATED WORK

A large amount of related work focus on bridging the gap between app description and user expectation. WHYPER [9] and AutoCog [10] use NLP techniques to infer permission use from app descriptions. Yu et al. [30] revisited this approach and revealed that using description and permission will lead

<sup>3</sup>virustotal.com

<sup>4</sup>http://www.tcpdump.org

<sup>5</sup>AVL for Android, com.antivy.avl

to many false positives. Thus they proposed exploiting the privacy policy and its bytecode to enhance the malware detection based on app description. Our work is motivated by CHABADA [11], which uses LDA on app descriptions to identify the main topics of each app, and then clusters apps based on related topics. By extracting sensitive APIs used for each app, it can identify outliers which use APIs that are uncommon for that cluster. Ma et al. [29] extended CHABADA by proposing an active and semi-supervised approach to detect malware using both known benign and malicious apps. Wang et al. [15], [36], [37] proposed to infer the purpose of permission use. However, all of the previous studies do not consider the impact of TPLs when checking whether the app behaves as it advertised.

## VI. CONCLUDING REMARKS

In this paper, we present a study to show that TPLs play an important role in pinpointing the inconsistency between app description and app behavior. Based on the extensive experiment on more than 400K apps, we show that more than half of the apps are no longer identified as outliers after filtering TPLs, and we could identify more new outliers. The results in this paper could shed a light on a new perspective for researchers that TPLs could be potentially used to enhance a variety of mobile app analysis tasks.

## VII. ACKNOWLEDGEMENT

This work is supported by the science and technology project of State Grid Corporation of China: “Research on Key Technologies of Security Threat Analysis and Monitoring for Power Mobile Terminals” (Grant No. SGRIXTKJ[2017]265).

## REFERENCES

- [1] “2018 malware forecast: the onward march of android malware,” <https:// NakedSecurity.sophos.com/2017/11/07/2018-malware-forecast-the-onward-march-of-android-malware/>, 2018.
- [2] M. Grace, Y. Zhou, Q. Zhang, S. Zou, and X. Jiang, “Riskranker: scalable and accurate zero-day android malware detection,” in *MobiSys 2012*, pp. 281–294.
- [3] D.-J. Wu, C.-H. Mao, T.-E. Wei, H.-M. Lee, and K.-P. Wu, “Droidmat: Android malware detection through manifest and api calls tracing,” in *Asia JCS 2012*. IEEE, 2012, pp. 62–69.
- [4] M. Liu, H. Wang, Y. Guo, and J. Hong, “Identifying and analyzing the privacy of apps for kids,” in *Proceedings of the HotMobile 2016*, pp. 105–110.
- [5] Z. Kan, H. Wang, G. Xu, Y. Guo, and X. Chen, “Towards light-weight deep learning based malware detection,” in *The 42nd IEEE International Conference on Computers, Software & Applications (COMPSAC 2018)*.
- [6] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, “Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones,” *ACM Transactions on Computer Systems*, vol. 32, no. 2, p. 5, 2014.
- [7] L.-K. Yan and H. Yin, “Droidscape: Seamlessly reconstructing the os and dalvik semantic views for dynamic android malware analysis,” in *USENIX security symposium*, 2012, pp. 569–584.
- [8] H. Wang, H. Li, L. Li, Y. Guo, and G. Xu, “Why are Android apps removed from google play? a large-scale empirical study,” in *15th International Conference on Mining Software Repositories (MSR 2018)*.
- [9] R. Pandita, X. Xiao, W. Yang, W. Enck, and T. Xie, “WHYPER: Towards automating risk assessment of mobile applications,” in *USENIX Security 2013*, pp. 527–542.
- [10] Z. Qu, V. Rastogi, X. Zhang, Y. Chen, T. Zhu, and Z. Chen, “Autocog: Measuring the description-to-permission fidelity in android applications,” in *CCS 2014*, pp. 1354–1365.
- [11] A. Gorla, I. Tavecchia, F. Gross, and A. Zeller, “Checking app behavior against app descriptions,” in *ICSE 2014*, pp. 1025–1035.
- [12] H. Wang, Y. Guo, Z. Ma, and X. Chen, “Wukong: A scalable and accurate two-phase approach to android app clone detection,” in *ISSITA 2015*, pp. 71–82.
- [13] N. Viennot, E. Garcia, and J. Nieh, “A measurement study of google play,” *SIGMETRICS Perform. Eval. Rev.*, vol. 42, no. 1, pp. 221–233, Jun. 2014.
- [14] J. Lin, S. Amini, J. I. Hong, N. Sadeh, J. Lindqvist, and J. Zhang, “Expectation and Purpose: Understanding Users’ Mental Models of Mobile App Privacy Through Crowdsourcing,” in *UbiComp ’12*, 2012, pp. 501–510.
- [15] H. Wang, Y. Li, Y. Guo, Y. Agarwal, and J. I. Hong, “Understanding the purpose of permission use in mobile apps,” *ACM Trans. Inf. Syst.*, vol. 35, no. 4, pp. 43:1–43:40, Jul. 2017.
- [16] R. Stevens, C. Gible, J. Crussell, J. Erickson, and H. Chen, “Investigating user privacy in Android ad libraries,” in *MoST 2012*.
- [17] M. C. Grace, W. Zhou, X. Jiang, and A.-R. Sadeghi, “Unsafe exposure analysis of mobile in-app advertisements,” in *WISEC’12*, pp. 101–112.
- [18] F. Dong, H. Wang, L. Li, Y. Guo, G. Xu, and S. Zhang, “How do mobile apps violate the behavioral policy of advertisement libraries?” in *Proceedings of the HotMobile 2018*, pp. 75–80.
- [19] H. Wang, Y. Guo, Z. Tang, G. Bai, and X. Chen, “Reevaluating android permission gaps with static and dynamic analysis,” in *Proceedings of GlobeCom*, ser. *GlobeCom’15*, 2015.
- [20] “Language detection library for Java,” 2016. [Online]. Available: <https://github.com/optimaize/language-detector>
- [21] “Mallet: A machine learning for language toolkit,” 2002. [Online]. Available: <http://mallet.cs.umass.edu>
- [22] “Snowball: A language for stemming algorithms,” <http://snowballstem.org>, 2001.
- [23] D. E. Goldberg and J. H. Holland, “Genetic algorithms and machine learning,” *Machine Learning*, vol. 3, no. 2, pp. 95–99, Oct 1988.
- [24] X. Yang, D. Lo, L. Li, X. Xia, T. F. Bissyand, and J. Klein, “Characterizing malicious android apps by mining topic-specific data flow signatures,” *Information and Software Technology*, vol. 90, pp. 27 – 39, 2017.
- [25] “Scikit-learn: Machine learning in Python,” <http://scikit-learn.org>, 2011.
- [26] “Pyevolve: A complete genetic algorithm framework written in pure Python,” 2009. [Online]. Available: <http://pyevolve.sourceforge.net>
- [27] A. P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner, “A survey of mobile malware in the wild,” in *Proceedings of SPSM 2011*, pp. 3–14.
- [28] K. W. Y. Au, Y. F. Zhou, Z. Huang, and D. Lie, “Pscout: Analyzing the android permission specification,” in *CCS 2012*, pp. 217–228.
- [29] S. Ma, S. Wang, D. Lo, R. H. Deng, and C. Sun, “Active semi-supervised approach for checking app behavior against its description,” in *COMPSAC 2015*, pp. 179–184.
- [30] L. Yu, X. Luo, C. Qian, S. Wang, and H. K. Leung, “Enhancing the description-to-behavior fidelity in android apps with privacy policy,” *IEEE Transactions on Software Engineering*, 2017.
- [31] Z. Ma, H. Wang, Y. Guo, and X. Chen, “Libradar: Fast and accurate detection of third-party libraries in android apps,” in *ICSE ’16*, 2016, pp. 653–656.
- [32] H. Wang and Y. Guo, “Understanding third-party libraries in mobile app analysis,” in *Proceedings of the 39th International Conference on Software Engineering Companion*, ser. *ICSE-C ’17*, 2017, pp. 515–516.
- [33] H. Wang, Y. Guo, Z. Ma, and X. Chen, “Automated detection and classification of third-party libraries in large scale android apps,” *Journal of Software*, vol. 28, no. 6, pp. 1373–1388, 2017.
- [34] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation-based anomaly detection,” *ACM Trans. Knowl. Discov. Data*, vol. 6, no. 1, pp. 3:1–3:39, Mar. 2012.
- [35] W. HaoYu, W. ZhongYu, G. Yao, and C. XiangQun, “Detecting repackaged android applications based on code clone detection technique,” *SCIENCE CHINA Information Sciences*, vol. 44, no. 1, pp. 142–157, 2014.
- [36] H. Wang, J. I. Hong, and Y. Guo, “Using text mining to infer the purpose of permission use in mobile apps,” in *The 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp 2015)*, pp. 1107–1118.
- [37] H. Wang, Z. Liu, Y. Guo, X. Chen, M. Zhang, G. Xu, and J. Hong, “An explorative study of the mobile app ecosystem from app developers’ perspective,” in *Proceedings of the 26th International Conference on World Wide Web (WWW 2017)*, pp. 163–172.

# Whether Android Applications Broadcast Your Private information: A Naive Bayesian-based Analysis Approach

Li Lin<sup>1,2,3</sup>, Jian NI<sup>1,2,3</sup>, Xinya Mao<sup>1,2,3</sup>, Jianbiao Zhang<sup>1,2,3</sup>

<sup>1</sup>College of Computer Science, Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China

<sup>2</sup>Beijing Key Laboratory of Trusted Computing, Beijing 100124, China

<sup>3</sup>National Engineering Laboratory for Classified Information Security Protection, Beijing 100124, China  
linli\_2009@bjut.edu.cn, {17801028290, 13089397966}@163.com, zjb@bjut.edu.cn

**Abstract**—With the rapid development of android smart terminals, android applications are exhibiting explosive growth. However, there remains a challenging issue facing android system, a malicious application may broadcast user's private information. In this paper, we propose a Naive Bayesian-based approach for analyzing private information leakage under the android broadcast mechanism, which calls BRbysA. Firstly, broadcast actions registered in manifest.xml are picked up statically by keyword matching technique. Secondly, with the Xposed framework, the broadcast actions specified at run time are discovered by hooking broadcast callback onReceive() function. Combining the above two ways, we can capture all real-time broadcast actions in an android application. Thirdly, we adopt Naive Bayesian learning algorithm, all broadcast actions which involved in users' privacy leakage are analyzed and classified. Finally, we evaluate the proposed approach by using the dataset from Drebin and Google Play.

**Keywords**- Android broadcast; private information leakage; keyword matching; hook; Naive Bayesian

## I. INTRODUCTION

Nowadays more attention has been paid to individual privacy information protection in android platform [1]. There are different security capabilities in android custom ROM [2]. It leads to the potential loopholes and vulnerabilities in android system [3]. For example, a loophole in BlackPhone has been discovered to allow an attacker to decrypt users' encrypted information [4]. There has occurred a new malware through Google Play Store to inject malicious codes for collecting the user's bank and credit card information [5]. Consequently, it is of great importance to underpin trusted android market by malicious applications analysis approach, which can accurately and efficiently detect APKs that reveal users' private information.

Existing methods are mainly divided into two categories: static detection and dynamic monitoring. Static detection methods involve in the analysis of signature information and permission information. However it is difficult to find the privacy disclosure from android broadcast mechanism by analyzing permission information. Dynamic detection methods

like MonkeyRunner and HIPS can achieve malicious behavior detection with higher detection accuracy, but it also suffer from low code coverage and longtime consumption. In fact, android system may broadcast users' private information [6]. If a malicious program can accept system broadcast, it can obtain users' private information by calling onReceive() method.

To solve these problems, this paper proposes a Naive Bayesian-based approach called BRbysA for analyzing private information leakage under the android broadcast mechanism. Firstly, broadcast actions registered in the manifest.xml are picked up statically by exploiting keyword matching technique. Secondly, referencing Xposed framework, the broadcast actions specified at running time are discovered by hooking broadcast callback onReceive() function. Combining the above two ways, we can capture all real-time broadcast actions in an android application. Thirdly, adopting Naive Bayesian-based learning algorithm, all broadcast actions involved in users' privacy leakage are analyzed and classified. To our knowledge, we are the first one to introduce the machine learning-based method to analyze android broadcast receiver. Furthermore, based on the above analysis, the probability of each broadcast actions appeared in malicious programs is calculated to determine whether the application discloses users' private information Based on Drebin dataset proposed by Daniel et al [7] and Google Play application set, we have test the effectiveness of the proposed method successfully. We summed up several kinds of broadcast actions that really exist the risk of leaking users' private information.

The rest of the paper is organized as follows. Section II presents related work. Section III gives the problem description. Section IV introduces the proposed method in detail. Section V presents our experimental results. Section VI concludes this paper and shows some possible future work.

## II. RELATED WORK

In the aspect of android malware detection, there are two schemes: static detection and dynamic monitoring. Static detection methods [8] mainly concern the analysis of signature information, permission information of APKs. Dynamic

detection methods [9], [10], [11] can trigger the malicious behavior of malicious programs by running android applications. Common dynamic testing tools include Monkey, MonkeyRunner, TaintDroid, droidBox and so on [12].

Most of current malicious applications detection researches focus on the analysis of permissions and malicious APIs. There was a little work on investigating android broadcast mechanism. Fadi Mohsen et al. [6] have developed a tool broadcastViewer, which can be used to view android system broadcast actions that has been installed on your phone. However, the work detects broadcast actions only from the perspective of static analysis and cannot detect the behavior of broadcast receiver when the actual operations are triggered by the application. In addition, Erika Chin et al. [13] proposed ComDroid to analyze android inter-process communication risk. Di Tian et al. [14] analyzed the characteristics of Broadcast Receiver vulnerability and developed a broadcast receiver vulnerability detection system called BRVD. Noam Kogan et al. [15] present an anti-pirate revocation scheme for broadcast encryption systems. The above three work have mentioned that broadcast message may be with user's private information in broadcast transmission and inter-application communication. But they also didn't analyze all broadcast actions systematically. It may lead to miss some broadcast actions that reveal users' private information.

### III. PROBLEM DESCRIPTION

Users download a wide variety of APKs from third-party application market in order to meet their daily commerce and communication needs [16]. An android system consists of four components: BroadcastReceiver, Activity, Service, and Content provider. BroadcastReceiver is used to receive broadcast declared in Manifest.xml file. At this time, BroadcastReceiver also can call its onReceive() method to perform certain operations. Interaction between BroadcastReceiver and android system is shown in Fig. 1. The broadcast information may contain users' private information. A malicious program can perform android broadcast callback function by customizing onReceive() method. For example, if a malicious program has registered SMS BroadcastReceiver, it can receive SMS messages and access the information through intent.getExtras() method. In this paper, our objective is to provide a sound method for detecting and analyzing malicious android APKs, which use broadcast to steal user privacy.

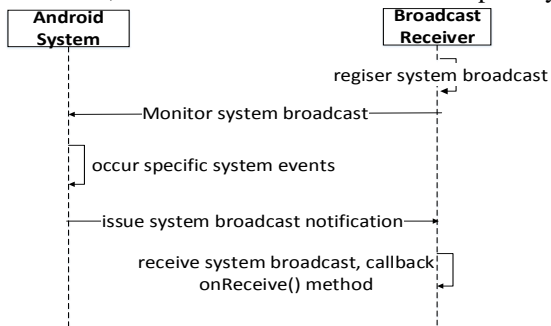


Figure 1. Interaction between broadcast receiver and android system

### IV. DESIGN OF BRBysA

As shown in Fig.2, the BRbysA method is divided into three parts: broadcast action static collection, broadcast action dynamic discover and Naive Bayesian-based privacy leakage analysis. It must combine static and dynamic methods to collect broadcast action, because the existing dynamic technologies do not monitor all broadcast actions in the APK's manifest.xml. Moreover, exploiting Naive Bayesian-based learning algorithm, we can determine whether there exist the risk of leaking users' privacy through broadcast action information in an APK.

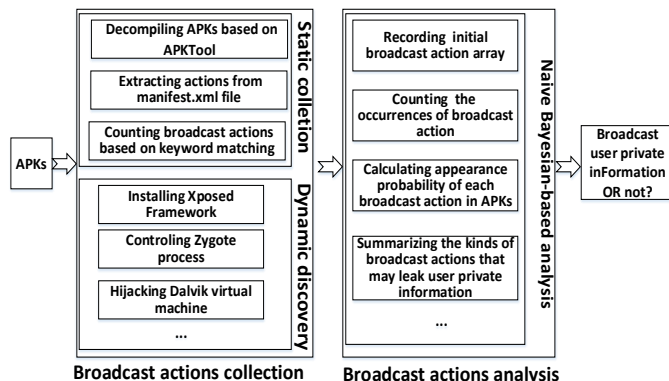


Figure 2. The principle of BRbysA

#### A. Statically collecting broadcast action

In general, self-defined BroadcastReceiver is specified by receiver label. A particular broadcast action can be specified by intent-filter label and received by BroadcastReceiver. In BRbysA, broadcast actions in APK's manifest.xml file are collected using keyword matching algorithm in Table I.

TABLE I. KEYWORD MATCHING-BASED BROADCAST ACTION EXTRACTION ALGORITHM

KEYWORD MATCHING-BASED BROADCAST ACTION EXTRACTION ALGORITHM	
Input:	APK package under test
Output:	broadcast action set
Begin:	
1.	Decompile APK source file based on APKTool.
2.	Extract the manifest.xml file of APK into a specific directory.
3.	Read the manifest.xml using FileReader and BufferedReader.
4.	Split the manifest.xml across blank, store elements into String array.
5.	Create broadcast actions set, to compare with String array
	<pre> while ( bufferedReader.readLine() != null) {     String[] words = line.split(" ");     for (String word : words) {         if (word.equals("SMS_RECEIVED"))         {arrayList.add("SMS_RECEIVED")}} } </pre>
6.	If the broadcast action equals the array element, then add it to the result set.
End.	

#### B. Dynamically discovering broadcast action

To obtain broadcast actions specified at run time, this paper proposes a hook-based dynamic monitoring scheme to observe the actual state of broadcast callback onReceive() method. The hook operation for broadcast registration API is carried out based on Xposed framework. BRbysA states the XposedInstaller entrance class in the configuration file assets/xposed\_init. The algorithm is shown in Table II.

TABLE II. BROADCAST ACTION DYNAMICAL DISCOVERY ALGORITHM BASED ON XPOSED FRAMEWORK

Input: APK package under test
Output: broadcast action set
Begin:
1. Android cellphone install Xposed framework, then configure Xposed framework in project's manifest.xml.
2. Import XposedBridgeApi-54.jar into project as library file, the stating XposedInstaller entry class in the assets/xposed.init file.
3. Main class Module implements IXposedHookLoadPackage Interface, override handleLoadpackage(LoadPackageParam lpparam) method.
4. Assign the hooked package name、method name and param type, across XposedHelpers.findAndHookMethod().
5. When the specific package is loaded, the system will call findAndHookMethod(), and then call beforeHookedMethod and afterHookedMethod.
6. Get broadcast action through ((Intent)param.arg[1]).getAction to, then storage it into broadcast action array.
End.

### C. Naive Bayesian-based privacy leakage analysis

Based on the above methods, all real-time broadcast actions can be captured in an APK. Next, a naive Bayesian machine learning algorithm is introduced to classify broadcast actions. Let  $X_0$  be normal program,  $X_1$  be malicious program,  $C_i$  represents broadcast action  $ba_i$  appearing in an APK, then the formula  $P(C_i|X_i) = P(X_i|C_i) * P(C_i)/P(X_i)$  (1) indicates the occurrence probability of broadcast action  $ba_i$  when the APK is normal or malicious. The algorithm is shown in Table III.

TABLE III. NAIVE BAYESIAN-BASED PRIVACY LEAKAGE ANALYSIS ALGORITHM

Input: Broadcast action set outputted by the above methods
Output: the conditional probability of occurrence of each broadcast action when an APK is malicious
Begin:
1. Construct an array ActionArrays[], each element of ActionArrays[] is from the broadcast action set outputted by the above methods.
2. Array ClassVec[] stores APKs' classification, where '0' indicates normal program, '1' indicates malicious program. The array ArraySingle is used to store the occurrence number of unreplicated broadcast action. The initial elements of ArraySingle are set all '0'.
3. For any broadcast action $ba_i$ , count the number of malicious APKs from ActionArrays under the premise of $ba_i$ occurrence, which is denoted by $N_{i1}$ , count the total number of all APKs from ArraySingle under the premise of $ba_i$ occurrence, which is denoted by $N_{i2}$ . Calculate $P(X_1 C_i)$ through $N_{i1}$ divided by $N_{i2}$ .
4. For any APK, calculate $P(X_1)$ , the probability that APK is malicious.
5. For any broadcast action $ba_i$ , count the number of $ba_i$ in ArraySingle $N_{i3}$ ; Count the total number of all broadcast action in ActionArrays $N_{i4}$ . Calculate $P(C_i)$ through $N_{i3}$ divided by $N_{i4}$ .
6. For any broadcast action $ba_i$ , calculate $P(C_i X_1)$ by Formula (1).
7. For different broadcast actions in an APK, count the average value of $P(C_i X_1)$ and find a suitable threshold. Determine whether an APK will reveal user private information by judging if the above average value exceeds the selected threshold or not.
End.

## V. EXPERIMENT EVALUATION

### A. Experiment environment

We have implemented and deployed *BRbysA* on Android version 4.4. The total kinds of broadcast actions is 136. The Xposed Framework version used in *BRbysA* is No.54. We randomly choose 1000 regular APKs from the Google Play and

download 1000 malicious APKs from Drebin dataset as experimental samples.

### B. Static collection and dynamic discovery effectiveness

Using the algorithms in Table I and II, we have counted the number of all broadcast actions in the 2000 APKs. Due to limited space, here we only release statistic data. As shown in Table IV, only 31 in 136 kinds of broadcast actions appears in the 2000 APKs, the others don't appear. PA represents the average value of conditional probability for broadcast actions occurring under the premise of malicious APK. The result shows that the first five have higher frequency than 0.3.

TABLE IV. THE NUMBER OF ALL BROADCAST ACTIONS IN THE SAMPLE.

Broadcast action	Num.	PA
android.provider.Telephony.SMS_RECEIVED	300	0.4507
android.net.conn.CONNECTIVITY_CHANGE	243	0.4105
android.net.wifi.NETWORK_IDS_CHANGED	156	0.3954
android.intent.action.PHONE_STATE	113	0.3522
android.intent.action.NEW_OUTGOING_CALL	84	0.3013
Intent.ACTION_LOCALE_CHANGED	53	0.2652
Intent.ACTION_REBOOT	22	0.1024
android.hardware.action.NEW_PICTURE	7	0.0202
android.hardware.action.NEW_VIDEO	4	0.0106
android.intent.action.CAMERA_BUTTON	2	0.0103
android.intent.action.DATA_SMS_RECEIVED	2	0.0096
android.intent.action.FETCH_VOICEMAIL	2	0.0043
android.intent.action.NEW_VOICEMAIL	1	0.0043
android.intent.action.PROVIDER_CHANGED	1	0
android.intent.action.PROXY_CHANGE	1	0
android.intent.action.SCREEN_ON	1	0
android.intent.action.SCREEN_OFF	1	0.0043
android.media.VIBRATE_SETTING_CHANGED	1	0.0043
android.net.nsd.STATE_CHANGED	1	0
android.net.wifi.STATE_CHANGE	1	0
android.nfc.action.ADAPTER_STATE_CHANGED	1	0
android.provider.Telephony.SMS_RECEIVED	1	0.0043
android.intent.action.WALLPAPER_CHANGED	1	0
android.media.AUDIO_BECOMING_NOISY	1	0
android.intent.action.PACKAGE_FIRST_LAUNCH	1	0.0043
android.intent.action.MEDIA_SCANNER_SCAN_FILE	1	0.0043
android.intent.action.INPUT_METHOD_CHANGED	1	0
android.intent.action.BATTERY_LOW	1	0
android.intent.action.BATTERY_CHANGED	1	0
android.bluetooth.device.action.UUID	1	0.0043
android.bluetooth.devicepicker.action.LAUNCH	1	0.0043

### C. Malicious APKs detection accuracy

Next, we use the above samples to test malicious APK detection accuracy of *BRbysA*. As shown in Fig. 3, the horizontal axis represents probability value ranges, the vertical axis represents the number of APKs that PA falls in a certain probability value range. In the experiment, 0.3 is selected as a threshold. As shown in Fig. 3, the number of APKs whose PA is greater than 0.3 is  $123+141 + 26 = 290$ . That means, 290 malicious APKs can be detected by *BRbysA*. Meanwhile, the number of APKs registered broadcast actions in 1000 malicious APKs is counted as 310. Considering the most extreme situation, all malicious APKs registered broadcast actions exist the risk of leaking users' private information. Thus the malicious APKs detection rate of *BRbysA* is  $290/310=93.548\%$ . The result shows that *BRbysA* can achieve good malicious APKs detection.



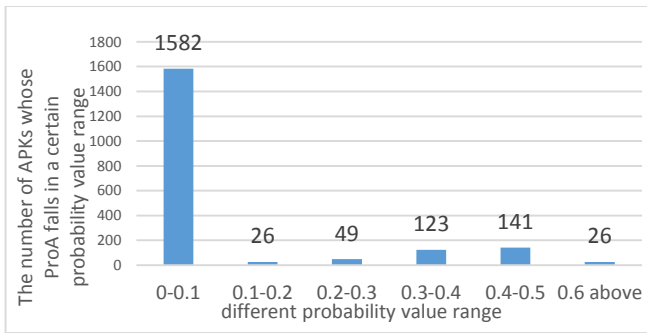


Figure 3. Different probability value range vs the number of APKs whose ProA falls in a corresponding probability value range.

Finally, we compare *BRbysA* with Manilyzer [17]. In the experiment, we randomly select 100 APKs from the above 310 malicious APKs registered broadcast actions and select 100 APKs from the above 1000 regular APKs. *BRbysA* and Manilyzer are implemented to analyze the selected 200 APKs. As shown in Fig. 4, the horizontal axis represents different rounds and the vertical axis represents the number of detected malicious APKs under different malicious APKs detection methods. The result shows that *BRbysA* can work better than Manilyzer at malware detection rate in most cases.

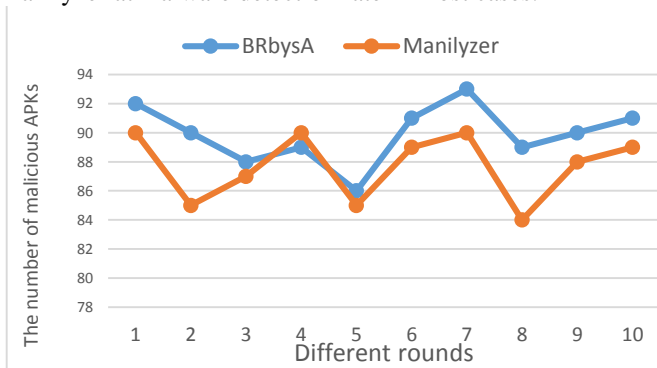


Figure 4. Different rounds vs the number of detected malicious APKs under different malicious APKs detection methods.

## VI. CONCLUSIONS AND FUTURE WORK

This paper proposes *BRbysA* to analyze private information leakage under android broadcast mechanism. The highlights of this paper are as follows. Firstly, combining static collection and dynamic discovery, all real-time broadcast actions in an APK can be captured. Secondly, adopting Naive Bayesian learning algorithm, all broadcast actions involved in users' privacy leakage can be analyzed and classified. Finally, we have evaluated *BRbysA* using the dataset from Drebin and Google Play, which illustrate there are several kinds of broadcast actions that really exist the risk of leaking users' private information.

However, the problem of threshold selection is not covered in this paper. With the change of sample size, we have found that different thresholds may make different decisions. This is desirable to develop some data mining approaches to increase the malware detection accuracy in the set of more malwares. Furthermore, our ongoing research will test the time and load overhead performance of the proposed approach.

## ACKNOWLEDGMENT

This work is partially supported by the National Science Foundation of China (No. 61502017), the Scientific Research Common Program of Beijing Municipal Commission of Education (KM201710005024).

## REFERENCES

- [1] Zhang. W, Li.X, Xiong.N and Vasilakos. A. V, "Android platform-based individual privacy information protection system," *Personal and Ubiquitous Computing*, Vol. 20, no. 6, pp. 875-884, 2016.
- [2] Xu. M, Song. C.Ji. Y, Shih. M, Lu. K, Zheng C, et al, "Toward Engineering a Secure Android Ecosystem: A Survey of Existing Techniques," *ACM Computing Surveys*, Vol. 49, no. 2, pp.38, 2016.
- [3] Yang. K, Zhuge.J, Wang. Y, Zhou. L, and Duan. H, "IntentFuzzer: detecting capability leaks of android applications," in *Proceedings of the 9th ACM symposium on Information, computer and communications security*, pp.531-536, ACM, 2014.
- [4] 360 security broadcast, The world's most secure mobile phone BlackPhone was found loopholes can cause loss of privacy [Online]. Available: <http://bobao.360.cn/news/detail/1171.html> 2015-01.
- [5] Ren. C, Chen. K, and Liu. P, "Droidmarking: resilient software watermarking for impeding android application repackaging," in *Proceedings of the 29th ACM/IEEE international conference on Automated software engineering*, pp.635-646, ACM, 2014.
- [6] F. Mohsen, M. Shehab, E. Bello-Ogunu, and AA. Jarrah, "Android System Broadcast Actions Broadcasts Your Privacy," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp.1484-1486, ACM, 2014.
- [7] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon and K. Rieck, "DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket," in *Proceedings of NDSS 2014*, Internet Society, 2014.
- [8] Quan. D, Zhai. L, Yang.F and Wang. P, "Detection of Android Malicious Apps Based on the Sensitive Behaviours," in *Proceedings of TrustCom 2014*, pp. 877-883, IEEE, 2014.
- [9] S. S. Shinde and S. S. Sambare. "Enhancement on privacy permission management for Android apps," *Communication Technologies*, pp.838-842, IEEE, 2015.
- [10] Yang.W, Xiao. X, P. Rahul, E. William and Xie.T, "Improving mobile application security via bridging user expectations and application behaviors," in *Proceedings of the 2014 Symposium and Bootcamp on the Science of Security*, pp.32, ACM, 2014.
- [11] S. Feese, B. Arnrich, G. Troster, M. Burtscher, B. Meyer and K. Jonas, "CoenoFire: monitoring performance indicators of firefighters in real-world missions using smartphones," in *Proceedings of the ACM international joint conference on Pervasive and ubiquitous computing*, pp.83-92, ACM, 2013.
- [12] S. K. Singh, B. Mishra and P. Gera, "A privacy enhanced security framework for android users," in *Proceedings of the 5th International Conference on IT Convergence and Security*, pp.1-6, IEEE, 2015.
- [13] E. Chin, A. P. Felt, K. Greenwood and D. Wagner, "Analyzing Inter-Application Communication in Android," in *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services, MobiSys'11 and Co-located Workshops*, pp. 239-252, ACM, 2011.
- [14] Tian.D, "Detecting Vulnerabilities of Broadcast Receivers in Android Applications," *ProQuest Dissertations and Theses A&I: The Sciences and Engineering Collection*, 2016.
- [15] N. Kogan, Y. Shavitt and A. Wool, "A practical revocation scheme for broadcast encryption using smartcards," *ACM Transactions on Information and System Security (TISSEC)*, Vol.9,no.3,pp.225, 2006.
- [16] Li. Y, Yao. F, Lan. T and Venkataramani. G. , "SARRE: Semantics-Aware Rule Recommendation and Enforcement for Event Paths on Android," *IEEE Transactions on Information Forensics and Security*: Vol.11, no 12, pp. 2748-2762, 2016.
- [17] S. Feldman, D. Stadther and B. Wang, "Manilyzer: Automated Android malware detection through manifest analysis," in *Proceedings of the 11th IEEE International Conference on Mobile Ad Hoc and Sensor Systems*, pp. 767-772, IEEE, 2015.

# Model Checking Method for SPA Page Transition Based on Component-based Framework

Naito Oshima

Graduate School of Creative Science and Engineering  
Waseda University  
Tokyo, Japan  
always-4869@akane.waseda.jp

Tomoji Kishi

School of Creative Science and Engineering  
Waseda University  
Tokyo, Japan  
kishi@waseda.jp

**Abstract**—In recent years, because web applications have been handling increasingly important processing tasks, it is ever more important to avoid errors. Model checking is one verification method for detecting errors, whereby it is necessary to model the web application in order to verify it. However, typical web application developers may lack knowledge on creating the verification model. Furthermore, web applications have become increasingly diversified owing to web-browsing technological advancements and other factors. Among these, the single-page application (SPA) using a component-based web application framework, such as Angular, has attracted attention because of its excellent user experience. However, it makes modeling more difficult since the page structure is complicated by the intricate combination of components. In this paper, we therefore present a method to automatically generate verification models from source code and perform model checking. The method enables verification of SPA page transitions using the component-based framework. We apply our implemented automated tool to several applications. First, we experiment using sample applications that do not inject bugs and others that intentionally inject bugs. Moreover, we apply the method to real applications published on the Internet. The desired results are obtained, thereby confirming that the proposed method is effective.

**Keywords**—*Model Checking; Single-Page Application (SPA); Web Application Framework; Component-based; Angular.*

## I. INTRODUCTION

In recent years, the number of web applications handling important processing endeavors, such as online shopping, has markedly increased, thus magnifying the importance of avoiding errors. A comprehensive verification method for detecting errors is model checking [1], whereby it is necessary to model the target web application in order to verify it. However, the typical web application developer may have minimal knowledge of model checking.

Meanwhile, web applications have continued to diversify on account of the advancement of highly functional web browsers and web application development technology. Among them, a new type of single-page application (SPA) has been developed. By performing front-end processing, such as control of page transitions, which was traditionally performed on the back-end, it is possible to provide superior user experience (UX) with a fast response. Moreover, front-end frameworks—which are different from back-end frameworks, such as the

conventional Ruby on Rails—are emerging to foster front-end development. Front-end frameworks include Backbone.js, Vue.js, AngularJS, and others.

The recently released Angular framework has a novel component-based architecture that is different from the conventional one. However, in SPA, with the use of a component-based web application framework (henceforth “component-based framework”), the components are intricately combined. Therefore, the framework is more complicated than in the conventional one. For example, the page structure is complex, and errors are easily mixed in the page control part. It is believed that validation using model checking is effective for such an SPA. Nevertheless, since the pages are dynamically constructed by a combination of components, it is difficult to apply the modeling method with the existing conventional static page.

In this paper, we therefore propose a method to automatically generate a verification model and formulas that verify page transitions from SPA source code using the component-based framework. Model checking is also performed. We implement a tool that automates the proposed method for SPA using the Angular component-based framework. Experiments are conducted on several applications. We herein use Simple Promela Interpreter (SPIN) as the model checker. Hence, Process Meta Language (Promela) describes the verification model, and Linear Temporal Logic (LTL) describes the verification formula.

## II. BACKGROUND

### A. Single-Page Application (SPA)

In conventional web applications, each time an event occurs, such as a user interaction, the client synchronously requests the server (e.g., an initial request). The server responds to the client with all HTML of the corresponding page, and the client performs reloading and rendering processes.

In SPA, the server generally returns HTML, CSS, script files, and so on as a response only when responding to the initial request from the client. For subsequent requests from the client, we process and redraw using the front-end and asynchronously acquire data from the back-end in JSON format when needed. Using this mechanism, the SPA redraws

only the corresponding part without reloading the entire page, and it realizes page transitions as being controlled by the conventional back-end [2]. Thus, similar to a native application, the response to the user operation is fast and can provide excellent UX. Meanwhile, to realize SPA, a considerable amount of JavaScript code is necessary, and the front-end implementation and structure are complicated compared to the conventional approach [3].

### B. Component-based Framework

To improve development efficiency and quality, a web application framework is usually employed in web application development. Owing to front-end complexities, web application frameworks are likewise diversified, and front-end framework development has advanced [3]. The framework architecture has also changed, and frameworks adopting a new component-based approach are being developed.

The component-based concept is founded on Web Components [4], for which the World Wide Web Consortium (W3C) developed specifications. The following Web Component functions are the primary ones [4]:

- Custom Elements
- HTML Imports
- HTML Template
- Shadow DOM

The component-based framework usually includes these four functions. Unlike the back-end framework based on the model-view-controller (MVC) architecture [6] described for each role in the component-based framework, independent component grouping views, logic, and so on are defined for each element constituting the page.

SPA using the component-based framework dynamically constructs the whole page by combining those components. It is thereby possible to improve its reusability and other aspects. Furthermore, both page generation and page transitions are controlled by the front-end; thus, the component-based framework has a routing function to associate a component with a path (URL pattern) and to control page transitions.

### C. Angular

Angular [5] is a component-based framework developed by Google. For the development language, TypeScript, a superset of JavaScript, is recommended. It is not compatible with the earlier version of “AngularJS” (version 1). Since version 2 (September 2016 release), Angular has been referenced as “Angular.” In this research, an evaluation experiment is conducted on Angular 4.0.1. Additionally, SPA using Angular is defined as “Angular SPA.”

Basically one Angular component consists of the following:

- HTML Template, CSS Template
- TypeScript Class
- Metadata Using a Decorator

The entire page is comprised of one or more components, including a root component, which is the first component to be called when Angular SPA is activated.

In Angular, it is possible to dynamically replace parent components under the root component according to a path by RouterModule with a routing function. It is thus possible to realize page transitions, similar to the conventional one controlled by the back-end, without reloading the entire page.

In addition, a custom element can be created as a user-specific non-standard HTML tag (CustomTag) using the selector parameter of the @Component decorator describing the meta-information of each component. By inserting the value of the selector parameter as the tag name in the other parent component, the content of the HTML template of the given custom element can be displayed in the parent component. By using the custom element, a hierarchical structure can be composed of a parent component and a child component. Additionally, several of them can be arranged on one page, the components can be reused on different pages, and one component may be used on multiple pages. Figure 1. depicts an Angular page configuration example.

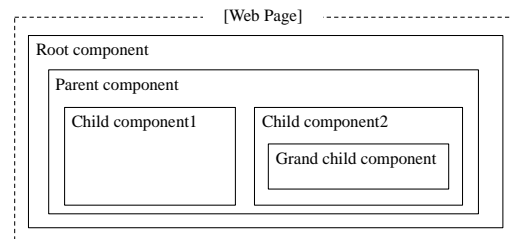


Figure 1. Example of a page structure using many components.

Thus, in general, the entire page of Angular SPA is largely divided into three elements:

- Root component
- Parent component controlled by the router (hereafter the “parent component”)
- Child component group to be inserted using tags of values declared by the selector parameter of @Component decorators as the custom element (hereafter the “custom child component”)

### III. RELATED WORK

In this section, we briefly discuss some research related to verification for applications, focusing especially on model checking and page transition verification.

In [7], page transition diagrams that are used during the design phase are addressed. A method of model checking for one aspect of the whole application, depending on the page transition and system environment, is proposed. The approach differs from ours in that the former handles the page transition diagram at the design stage, not the implementation stage. In [8], modeling is performed in the UML model and the reachability of the page is verified. Test cases are generated; nevertheless, verification is not performed using a formal method.

In [12], a method using JPF-Android, an Android application verification tool, applies Java PathFinder (JPF) to detect errors, such as deadlocking of Android applications. Analysis of the actual source code of the application is similar to that in the present research. However, the authors of [13] target native Android applications, not web applications.

In [13], the authors focus on Apache Struts of the MVC architecture web application framework. A method, “Web Automation by Changing View,” is proposed to model the behavior model of the web application. It is targeted at the implementation stage and is intended for web applications that use the Struts web application framework. However, when applying the model-checking method to the SPA page transition using the component-based framework, it is difficult to use a method of modeling one element of the MVC view as one page. Moreover, the extraction method is based on static page information.

In [9], [10], and [11], the authors focus on the implementation of a web application framework using dynamically typed languages. They respectively propose methods for extracting symbolic models to verify the data integrity of the model part and the access control security. Hence, the objectives differ from those of our research.

An example of front-end operation verification is the Rich Internet Application (RIA) (e.g., [15], [16]). The present paper differs from those works in that test cases are generated from execution traces of actual applications, attention is focused on interactions by event handlers, and search is performed by crawling. In the case of the crawling method, it cannot be verified that the back-end implementation has not been completed. However, our proposed approach focuses on the page transition part of the front-end. It can thus be verified without relying on the back-end implementation.

#### IV. METHODS

Our proposed method extracts necessary information from Angular SPA code for transforming the verification model and verification formulas for page transitions. Verification by model checking is performed to detect errors and improve the quality using verification models and their verification formulas. By implementing a tool that automates our proposed methods, ordinary web developers with minimal knowledge of model checking can also apply this method. The flow of the proposed method is shown as follows:

- A. Extraction of information from Angular SPA
- B. Construction of static page information
- C. Transformation to the Promela model
- D. Transformation to LTL formulas
- E. Verification by SPIN

Figure 2. depicts the overall extraction input and output processes. Meanwhile, the page transitions herein are defined as follows: “Changing of the parent component embedded in the router—the router-outlet tag in the root component corresponding to a path by RouterModule—consequently

changes the rendering of the entire page, similar to the transition of the conventional web application.”

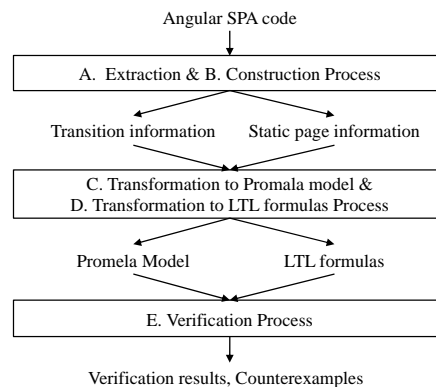


Figure 2. Overview of our proposed flow.

The page transition information is a set of <before page, path, after page>, and it is divided into two sets: “page information” and “transition information.” Page information indicates a path that can be transitioned from each page. It is a set of combinations of <before page, path>. Transition information indicates the transition destination page corresponding to the path described in the routing. It is a set of <path, after page>.

In this study, certain restrictions are placed on the Angular SPA description, such as not using child attributes and route parameters in routing. Since page transitions under the control of RouterModule are targeted, changes due to links to external web sites and data binding are not treated as page transitions.

##### A. Extraction of Information from Angular SPA

Transition information is included in routing defined by RouterModule and we thus extract it. Specifically, we extract path parameters and corresponding parent component names controlled by RouterModule.

Page information indicates a path that can transition from one page. In SPA, using the component-based framework, it consists of the set of “Page information in the root component” and “Page information in each component.” These pieces of information are included not only in the parent component controlled by the RouterModule, but also in the custom child component that can be inserted using the selector element as descendants of the parent component. Therefore, page information is extracted from all components. Furthermore, since information indicating the parent–child relationship of each component is also necessary, the “Custom element information in each component,” of which the parent component controlled by the RouterModule contains the descendant component, is also extracted.

##### B. Construction of Static Page Information

We automatically construct static page information from the information extracted in Section IV-A. Static page information is a set of transition-capable paths contained in those pages. In this paper, each of their page names is defined by the parent component name controlled by RouterModule.

We use the information of Section IV-A from the earlier flow outline to solve the parent-child relationships among the root component, parent component, and custom child component. We construct static page information representing the page information contained in each parent component controlled by RouterModule. It expresses the parent component name (the key part on the left of Table I) controlled by RouterModule, as well as the transition-capable paths (the value part on the right of Table I). We transform the Promela model and LTL formulas based on this static page information and transition information extracted in Section IV-A.

TABLE I. EXAMPLE OF GENERATED STATIC PAGE INFORMATION

<pre>{   "component1": ["/component2", "/component3"],   "component2": ["/component1"],   "component3": ["/component4"], }</pre>
--

### C. Translation to the Promela Model

Given the transition information of Section IV-A and the static page information generated in Section IV-B, we convert the verification model necessary for model checking. In this paper, since SPIN is used as the model checker, we express the verification model by Promela, the modeling language used in SPIN.

The page transitions controlled by routing are indicated by a set of “pages that can transition from one page and the page to which the path transitions,” such as <page 1, path, page 2>.

An example of page information by Promela is:

```
state == page1 -> state = path
```

An example of transition information by Promela:

```
state == path -> state = page2
```

As described above, a combination of page information and transition information expresses the Promela model of the page transition. In this Promela model, the state changes alternately with page, pass, page, pass... and so on. When it is possible to transition from one page to multiple pages, it is written as if it occurs non-deterministically using the syntax of “if...fi.” The process is repeatedly performed using “do ... od” of the guard command of the repeating syntax.

### D. Translation to LTL Formulas

The properties that generally hold in web applications are the following with reference to [7]:

- (1) The page reachable from the top page always has a next page in the transition (property 1).
- (2) Every page is reachable from the initial page (property 2).
- (3) The initial page is reachable from all pages (property 3).
- (4) A page transition is triggered only after several assumed pages (property 4).

We examine the above four properties. For property 1, we do not generate verification formulas because we do not input a formula. Rather, we perform verification using the default

deadlock-free of SPIN (1). For property 2, for the initial page p and arbitrary page q, we have the following LTL formula:

$$\neg \diamond(p \ \& \ \diamond q) \tag{2}$$

which will be verified. If a transition is possible, an error occurs, and it is confirmed that a transition from the initial page to any page is possible. By changing an arbitrary page q and repeatedly verifying all pages, it can be confirmed that the model satisfies property 2. For property 3, as with property 2, for initial page q and any page p, we have the following LTL formula:

$$\neg \diamond(p \ \& \ \diamond q) \tag{3}$$

If a transition is possible, an error occurs, and it is confirmed that a transition from an arbitrary page to an initial page is possible. By changing arbitrary page p and repeatedly verifying all pages, it can be confirmed that the model satisfies property 3.

As described above, in the validation of property 2 and property 3, since formulas are necessary for input, we automatically generate LTL formulas for all page names that can be transitioned from all paths defined in routing using transition information extracted by Section IV-A. This supports the verification.

In addition to the properties referencing [7], we verify property 4. For this property, in specifying one page q, and for any page p, we have the following LTL formula:

$$\neg \diamond(p \ \& \ XXq) \tag{4}$$

To use the next operator in SPIN, we must attach “-DNXT” option at gcc compile time. It is confirmed that p is included in the next page that can be transitioned from page q.

As described above, in this research, we model to include path transitions between page transitions, such as a path from a page and another page from a path. That is why the formula contains the two next operators (XX). By changing an arbitrary page, p, and repeating the verification for all pages, we can confirm that the model satisfies property 4. Specifically, it can be checked whether the next page is directly transferred from the unintended page to page p, and whether the next page can be transitioned directly from the intended page to page p.

### E. Verification by SPIN

We input the Promela model generated in Section IV-C and LTL formulas generated in Section IV-D into SPIN and verify the model for the formulas. In the automation tool, we verify each generated LTL formula. If the verification result is false, it automatically analyzes the trail file and automatically outputs the verification result and counter example simulation result as files, respectively.

## V. EXPERIMENTAL RESULTS

We employed an automated tool that implements the proposed method to conduct from the information extraction to the verification for SPA. When inputting Angular SPA, the automation tool can automatically perform all processes, from information extraction to generation of the verification model, generation of formulas, and execution of SPIN.

By applying our method to several sample applications, we checked whether the intended model was output. We then confirmed the feasibility of the flow of the proposed method and the feasibility of actually verifying it. In addition, we applied it to sample applications that intentionally incorporated errors to make properties false. We confirmed whether they could be verified correctly. Furthermore, to show the effectiveness of this method, we applied it to real applications published on the Internet.

### A. Experiment 1: Sample Applications

We show the page transitions of sample application 1 (hereafter “sample app1”) in Figure 3. First, we verified sample app1 with no errors in all properties. Next, we experimented using three of sample app2, sample app3, and sample app 4, in which errors for each property were injected.

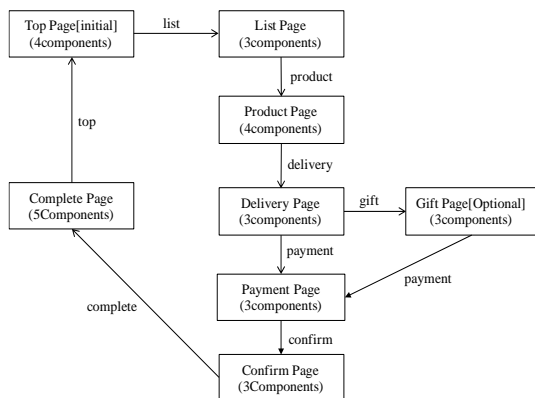


Figure 3. Page transition diagram of sample app1.

#### 1) Verification Model

We applied sample app1 to the automated tool and checked if the validation model was correctly generated automatically to represent the page transitions in Figure 6. As a result, the automatically generated verification model was correctly generated. It was generated automatically, as assumed from extraction to the modeling. Similarly, for sample app2, 3, and 4, the assumed verification model was correctly generated. Next, we verified the four properties using this verification model.

#### 2) Verification without Injecting Errors

We verified property 1 against the verification model of sample app1 automatically generated by (1). As a result, no error was output, and it was confirmed that there was no problem in its properties, as expected. Likewise, we verified property 2, property 3, and property 4.

#### 3) Verification with Injecting Errors

We verified sample app2, which intentionally injected the error of property 1 into sample app1. Specifically, in sample app2, there was no transition from “Complete Page” to “Top Page” of sample app1. We confirmed that the result was an error. As a result of verifying that property 1 was deadlock-free, it was possible to detect an intended error. Furthermore, a counter example was simulated using the output trail file. As a result, it was confirmed that the transition from “Complete Page” to “Top Page” was not completed and it stopped at “Complete Page,” as expected.

Similarly, we verified sample app3, which intentionally injected the error of property 2 into sample app1. Specifically, in sample app3, there is no transition from “Confirm Page” to “Complete Page.” Since the verification result of that part don’t result in an error, it is observed that there was no path to reach “Complete Page” from the initial page “Top Page,” and the intended error can be detected. As a result of the verification, the verification was completed without verifying the transition as an error. Therefore, it could detect the intended error.

Next, a bug injection experiment of property 3 was performed using sample app2 above. In sample app2, there was no transition from “Complete Page” to “Top Page” on the initial page. Therefore, contrary to property 3, it could not transition from all pages to the initial page. As a result, verification was completed without causing errors for all pages. Therefore, it was confirmed that an intended error was detected.

Finally, we tested sample app4, which intentionally injected the error of property 4 into sample app 1. Sample app4 added the transition from “Product Page” to “Payment Page” to sample app1. We confirmed the pages that could transition directly to “Payment Page.” From the verification results, transitions from “Delivery Page,” “Gift Page,” and the additional “Product Page” were possible. Specifically, we verified that it was possible to transition from “Product Page” to the next “Payment Page.” As a verification result, an intended error was detected. Owing to the simulation of the counter-example, we confirmed that it was possible to transition from “Product Page” to the next “Payment Page,” as intended.

### B. Experiment 2: Real Applications

To further demonstrate the effectiveness of our method, we applied the experiment to two different real applications (hereafter “Small App” and “Large App”), whose source code is published on GitHub [17]. We examined properties 1, 2, 3, and 4. In the case of property 4, we selected one of the parent component names defined in each routing and conducted the experiment. The scale of the two applications is shown below.

TABLE II. SCALE OF REAL APPLICATIONS

	LOC	Number of pages
Small App	1824	6
Large App	6893	23

In this experiment, we automatically generated the automatic verification model and formulas using the automated tool. The verification results using the automatically generated verification model and formulas are shown below.

TABLE III. RESULTS OF EXPERIMENTS FOR REAL APPLICATIONS

	Property 1	Property 2	Property 3	Property 4
Small	No error	No error	No error	No error
Large	No error	2 errors	2 errors	No error

First, as a result of verifying property 1, it was confirmed that there was no problem in its properties because no error was

output in either application. Similarly, we verified property 2. As a result, in Large App, there were two pages for which no error was output, and bugs were detected in the transition to two pages. We confirmed that part of the application, the reachable transition to that page, was described in none of the pages.

Next, we examined property 3. As a result, similar to property 2, in Large App, there were two pages wherein no error was output, and an error was detected in the transition from two pages. We confirmed that aspect of the application. Finally, we verified property 4. For each Small App and Large App, a single subsequent page was specified and verified. We visually checked whether the SPA could actually transition directly to that screen for pages that were made transition-capable by the verification result.

In property 4, it was self-evident that no bug existed. However, when the developer actually verifies it, it can be considered effective because it can detect whether the SPA directly transitions to an unintended page.

### C. Discussion

We conducted experiments on several applications using automated tools that we implemented. First, in the experiment on sample applications, it was possible to automatically correctly from the information extraction to verification of Angular SPA. In experiments with applications that did not inject bugs, and with applications that intentionally injected bugs, we obtained the desired verification results. Furthermore, even when we applied this method to applications published on the Internet, we could perform the task correctly. In practice, one application could detect multiple errors. Thus, our approach showed greater effectiveness.

Based on several experimental results, we confirmed that the proposed method enables correct generating and verifying of the verification model for the page transitions of Angular SPA. Moreover, by using our automated tool, it is considered that this method can be applied, even by ordinary developers who have minimal knowledge of model checking. As stated above, this study assumed certain constraints. These constraints mainly come from the first step of our method, i.e., the extraction of information from Angular SPA. By improving the analysis of SPA, these constraints can be decreased.

## VI. CONCLUSION

In this paper, we proposed a model checking method for page transitions of SPA using a component-based framework. In addition, we implemented an automated tool that applies this method of automatically generating verification models and formulas from extraction from source code of SPA. The tool additionally performs the verification. By using the tool, even typical developers with minimal model checking knowledge can apply the proposed method. Furthermore, it was confirmed that there was no problem in the flow of the proposed method by using real applications intentionally mixed with errors and those that actually showed the source code.

Focus on the front-end page without reliance on the back-end is a strength of our proposed approach. However, the

information extraction part of our implementation tool is directed to Angular SPA and thus assumes certain constraints. Therefore, it may be challenging to improve the tool and expand the application scope. Our future work will address this issue. Additionally, we will apply the method to various more complex Angular SPAs. Moreover, the information extraction part of the tool depends on Angular. Thus, we will consider implementing automation tools for SPAs using other component-based frameworks, such as Aurelia, or component-based libraries, such as React.

## REFERENCES

- [1] Edmund M. Clarke; Orna Grumberg; Doron Peled, "Model Checking," The MIT Press, 1999.
- [2] Madhuri A. Jadhav; Balkrishna R; Sawant, Anushree Deshmukh, "Single Page Application using AngularJS," International Journal of Computer Science and Information Technologies (IJCSIT), Vol.6, No.3, pp.2876-2879, 2015.
- [3] Ning Zhang; Yizhen Cao; Shengyan Zhang, "Research of web front-end engineering solution in public cultural service project," Computer and Information Science (ICIS), IEEE/ACIS 16th International Conference on, pp.623-626, 2017.
- [4] Web Components, <https://www.webcomponents.org/>, accessed: 2018-03-01.
- [5] Angular, <https://angular.io/>, accessed: 2018-03-01.
- [6] Glenn E. Krasner; Stephen T. Pope, "A cookbook for using the model-view controller user interface paradigm in Smalltalk-80," Journal of Object-Oriented Programming, Vol.1, No.3, pp.26-49, 1988.
- [7] Kei Homma; Satoru Izumi; Kaoru Takahashi; Atsushi Togashi, "Modeling, Verification and Testing of Web Applications Using Model Checker," IEICE Transactions on Information and Systems, Vol.94, No.5, pp.989-999, 2011.
- [8] Filippo Ricca; Paolo Tonella, "Analysis and testing of Web applications," Proceedings of the 23rd International Conference on Software Engineering (ICSE), Vol.47, No.6, pp.25-34, 2001.
- [9] Joseph P. Near; Daniel Jackson, "Rubicon: bounded verification of web applications," Proceedings of ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering (FSE), No.60, pp.1-11, 2012.
- [10] Joseph P. Near; Daniel Jackson, "Finding security bugs in web applications using a catalog of access control patterns," Proceedings of IEEE/ACM 38th International Conference on Software Engineering (ICSE), pp.947-958, 2016.
- [11] Ivan Bocić; Tefvik Bultan, "Symbolic Model Extraction for Web Application Verification," Proceedings of IEEE/ACM 38th International Conference on Software Engineering (ICSE), pp.724-734, 2017.
- [12] Heila van der Merwe, "Verification of android applications," Proceedings of the 37th International Conference on Software Engineering (ICSE), Vol. 2, pp.931-934, 2015.
- [13] Shoji Yuen; Keishi Kato; Daiju Kato; Kiyoshi Agusa, "Web automata: A behavioral model of web applications based on the MVC model," Information and Media Technologies, Vol.1, No.1, pp.66-79, 2006.
- [14] Alessandro Marchetto; Paolo Tonella; Filippo Ricca, "State-based testing of Ajax web applications," Software Testing, Verification, and Validation (ICST), 1st International Conference on, pp. 121-130, 2008.
- [15] Domenico Amalfitano; Anna Rita Fasolino; Porfirio Tramontana, "Rich internet application testing using execution trace data," Software Testing, Verification, and Validation Workshops (ICSTW), Third International Conference on, p. 274-283, 2010.
- [16] Frederik Nakstad; Hironori Washizaki; Yoshiaki Fukazawa, "Finding and Emulating Keyboard, Mouse, and Touch Interactions and Gestures while Crawling RIAs," International Journal of Software Engineering and Knowledge Engineering (SEKE), Vol.25, pp.1777-1782, 2015.
- [17] GitHub, <https://github.com/>, accessed: 2018-03-01.

# A Systematic Mapping Study on Software Comments Analysis

Amanda F. O. Passos  
Federal Institute of Bahia and  
Salvador University  
Salvador, Brazil  
amandainf@gmail.com

Mário André de F. Farias  
Federal Institute of Sergipe  
Aracajú, Brazil  
mario.andre@ifs.edu.br

Crescencio Lima  
Federal Institute of Bahia and  
Federal University of Bahia  
Salvador, Brazil  
crescencio@gmail.com

Manoel Mendonça  
Department of Computer Science,  
Federal University of Bahia  
Fraunhofer Project Center @ UFBA  
Salvador, Brazil  
manoel.mendonca@ufba.br

Rodrigo Oliveira Spínola  
PPGCOMP, Salvador University  
Fraunhofer Project Center @ UFBA  
State University of Bahia  
Salvador, Brazil  
rodrigo.spinola@unifacs.br

**Abstract—Context:** Mining software repositories has been used as an important tool to support software engineering research. Recent studies indicate that code comments are one of the most explored objects of analysis in the area. **Objective:** This work investigates how analysis of comments has been used to support software engineering activities by identifying its purposes, focuses, techniques, tools, evaluation methods, and the research type performed in the area. **Method:** We performed a systematic mapping study of the literature that considered papers from 1990 to 2016. **Results:** We analyzed 36 primary studies. The collected data pointed out that comment analysis has been used mainly for understanding and identifying the quality of software artifacts. The Dictionary/Vocabulary and Natural Language Processing are among the most used techniques, and most of them are performed in a semiautomatic way. We also organized a set of tools that have been used for mining software comments. Most of the primary studies are a solution proposal paper. Regarding evaluation methods, we found that experiments and case studies are the most considered. **Conclusion:** The results of this mapping study can help to identify points that still require further investigation in comment analysis research.

**Keywords—Code comment analysis; mining software repository; systematic mapping study.**

## 1. INTRODUCTION

Software repositories contain large amount of historical data embedded in different artifacts, such as source code, commit data, logs, e-mails, and comments. These data usually have rich cue to support the understanding of code changes, defects, quality issues in the evolution of software projects, and so on [10]. The Mining Software Repositories

DOI reference number: 10.18293/SEKE2018-013

(MSR) area focuses on analyzing and cross-linking the data available in different types of repositories to discover useful information about software projects [8].

MSR has been used as an important tool to support research on Software Engineering (SE) with different purposes such as prediction of defects analyzing commit data, identification of defects analyzing bug track system, comprehension of software evolution analyzing e-mail and commit data, and identification of technical debt [1] analyzing code comments. Recently, Farias *et al.* [4] performed a systematic mapping (SM) study to analyze studies on MSR by considering five editions of Working Conference on Mining Software Repositories (MSRConf). They reported that comments analysis is one of the most explored objects of study in the MSR area.

Comments analysis can, for example, reveal information such as the reason for adding new lines of code, knowing the progress of a collective task, or even why relevant changes were performed. Besides, code comments may also describe the developers' point of view about quality issues in the software development [5][6]. Due to such diversity of topics, our research group has faced some difficulty to have a broad view of the area when was starting the development of a new technology to support the identification of technical debt through code comment analysis.

Although some secondary studies have been performed in the MSR area [3][4][9][10], none of them has focused on comments analysis as object of study. It would be beneficial to have a broad view of the current research that has been performed on the area, so new directions of research could be better defined. In this context, this work presents the results of a mapping study performed to investigate the



following research question: “How has comments analysis been explored with the purpose of supporting software engineering activities?”. By answering this question, we intend to identify which purposes, focuses, techniques, tools, research types and evaluation methods have been considered on the research on comment analysis to support software engineering.

In total, 36 primary studies were selected for data extraction. We identified that analysis of comments has mainly been explored with the purpose of “Comprehension” and “Identification” of software engineering artifacts. Concerning the focuses, we observed that the most considered was “Quality of Software Artifacts” followed by “Technical Debt”. We also identified that the most commonly used mining techniques are Vocabulary/Dictionary, and they are usually performed in a semiautomatic way. Our analysis also concluded that the majority of studies can be characterized as “Solution Proposal”. However, we identified a rising number of “Evaluation Paper” in the last few years. As for the empirical evaluations, we verified that most of the studies have carried out “Controlled Experiments” followed by “Case Studies”.

We believe that the results of this mapping study will be beneficial for both researchers and practitioners. For the research community, this mapping will provide information about the current state of comments analysis research, as well as topics that require further investigation. For practitioners, the study presents a set of techniques and tools that can be used to improve or develop new approaches to explore comments analysis.

The remainder of this paper is structured as follows: Section 2 presents some related works. Next, Section 3 describes the systematic mapping protocol. Section 4 discusses the main outcomes of the study. Section 5 considers the threats to the validity of this study. Finally, Section 6 presents some concluding remarks.

## 2. RELATED WORKS

To the best of our knowledge, there are four secondary studies on MSR area. Kagdi *et al.* [10] performed a comprehensive literature survey on approaches for MSR in the context of software evolution. As a result, the authors proposed a taxonomy of different terms used by researchers for presenting purpose, focus, and object of analysis into categories. In another work, Hemmati *et al.* [9] analyzed 117 papers published in the MSRConf between 2004 and 2012. They codified a set of guidelines, tips, and recommendations, and provided a set of best practices that can be continuously used and updated as the MSR community matures and advances.

Next, Demeyer *et al.* [3] aimed to identify how the research on MSR evolved in the last decade. They focused on: (i) outdated research topics, (ii) the most (and less) frequently cited cases, (iii) emerging mining infrastructure, and (iv) software engineering state-of-the-practice. Finally,

Farias *et al.* [4] investigated recent studies on MSR approaches collecting data about software analysis goals (purpose, focus and object of analysis), data sources, evaluation methods, tools, and how the area is evolving.

The mapping study presented in this paper and the works discussed above are complementary to each other. Different from the others, our mapping study has a more specific focus and intends to investigate how comments analysis have been explored in MSR area with the purpose of supporting software engineering activities.

## 3. SYSTEMATIC MAPPING PROTOCOL

This work follows a well-organized set of guidelines for carrying out SMs in the context of software engineering [7], and the defined protocol is presented in the next subsections.

### A. Definition of Research Questions

For this study, a primary research question (RQ) was defined: “How has comments analysis been explored with the purpose of supporting software engineering activities?”. The following complementary research questions were derived from this main one. By answering these questions, we will have a detailed characterization of the identified studies.

#### **RQ1. Which are the main purposes and focus of researches on analysis of comments?**

The objective of this question is to identify the goal of MSR approaches in the area of analysis of comments. To perform the classification of the extracted data, we used a taxonomy (available at <https://goo.gl/qO6nUa>) discussed by Farias *et al.* [4]. By identifying the purpose, we classify the primaries goals of the studies (e.g., identification, characterization, prediction). In complement, by identifying their focuses (e.g., technical debt and defect), we classify the main attributes of interest in the studies between the purpose and the object of analysis (comment analysis). Thus, for example, we could have: to identify (purpose) technical debt (focus) exploring comments analysis (object of analysis).

#### **RQ2. What are the techniques used by researchers to analyze comments?**

This question intends to identify what techniques of comments analysis have been used to extract, process, and analyze comments. We also intend to categorize the techniques as manual, semiautomatic and automatic.

#### **RQ3. What are the tools used to extract, process, or analyze comments?**

This question aims to identify what tools have been used to extract, preprocess, and analyze comments. The result of this RQ is a set of tools that can be used in comments mining process.

#### **RQ4. Which empirical evaluations have been performed in the area?**

This question aims to identify whether the proposed approaches have been evaluated through empirical methods, and if so, which method was used. To classify the types of studies, we considered the empirical evaluation types discussed by Farias *et al.* [4]. The taxonomy is available at <https://goo.gl/qO6nUa>.

### RQ5. What are the identified research types?

This question intends to categorize the studies according to the research type facets defined by Wieringa *et al.* [11]: evaluation research, experience papers, opinion papers, philosophical papers, solution proposal, and validation research. By doing this, we intend to understand the overall contribution provided by the studies. In combination with the answers of RQ4, it also allows the identification of gaps and needs in the area.

#### B. Search strategy

In this study, we defined a generic search string:

((Software OR System OR Program OR Application) AND ((Comment analysis) OR (Comment Identification) OR (Code Comments) OR (Comment detection) OR (Examination of comments) OR (Analysis of comment) OR (Comments analysis) OR (Study of comments) OR (Comments study)))

We applied this search string to Titles and Abstracts. We chose not to do full text search because we found that it resulted in a very large number of studies out of scope.

#### C. Data Source

In choosing data sources, we aimed to include important journals and conferences regarding the research topic. For this, we considered the list recommended by Brereton *et al.* [2]: ACM Digital Library, IEEE Xplorer, Science Direct, Engineering Village, Springer Link, Scopus, and Citeseer.

#### D. Study Selection

After applying the search strings in the digital libraries, we filtered the relevant primary studies from the search results using the following selection criteria:

**Inclusion criteria:** (i) Published works that describe how comments analysis are used in software engineering activities; (ii) when several papers reported the same study, only the most recent was included; (iii) the publication date of the article should be between 1990 and 2016; and (iv) papers published in workshops, conferences or peer reviewed journals;

**Exclusion criteria:** (i) Studies out of the scope of this research; (ii) papers that are only available in the form of workshop/conference reports, abstracts or PowerPoint presentations; (iii) duplicated papers; and (iv) book chapters and articles published without revisions (white papers).

#### E. Screening of Papers

The screening of papers process to identify the primary studies comprises the following steps: (i) apply the selection criteria by reading the paper title and abstract and select the

relevant studies from the search results; and (ii) read introduction and conclusion in case the researcher needs further information to decide on the study selection. A master and a Ph.D. student performed these steps and other two experienced researchers reviewed the results.

#### F. Data Extraction

Before the data extraction execution, we performed a pilot extraction, aiming to align the researchers' understanding of the research questions. During the extraction process, researchers carefully read the primary studies in a peer-reviewed process. Two researchers extracted data for the same study and a third researcher solved the possible disagreements. All relevant data of each study was registered in a spreadsheet. At the end, one experienced researcher reviewed the extracted data. The complete data are available at <https://goo.gl/PGFVim>.

#### G. Data Analysis and Synthesis

We considered a quantitative method to analyze the extracted data. Although we have done an analysis on the results, most of them were summarized to present an overview of the findings. Thus, this work is characterized as a scoping study, which maps the primary studies on mining code comments in the software engineering area.

## 4. RESULTS AND DISCUSSION

We selected the papers following four steps. First, we searched papers in the digital libraries, resulting in 402 studies. Then, we removed the duplicated papers, resulting in 240. Next, we read the title and abstract to remove the studies out of the scope. This activity resulted in 68 studies, which were fully read. During this last step, we removed 32 studies, resulting in 36 papers to perform the data extraction. The complete list of selected studies is available at <https://goo.gl/q7nekb>.

We have 24 papers published on conferences, 9 on journals, and 3 on workshops. From a temporal point of view (Fig. 1), the studies are more concentrated from 2011. We also observed that 2015 was the year with more publications (6 – 16.7%), followed by 2016 (5 – 13.9%), 2011 and 2014, both with the same number of studies (4 – 11.43%). The increasing number of publications in the last two years (2015 and 2016) is partially justified by the presence of works relating comments analysis and Quality of Software Artifacts/Technical Debt. We can observe this trend in Fig. 2.

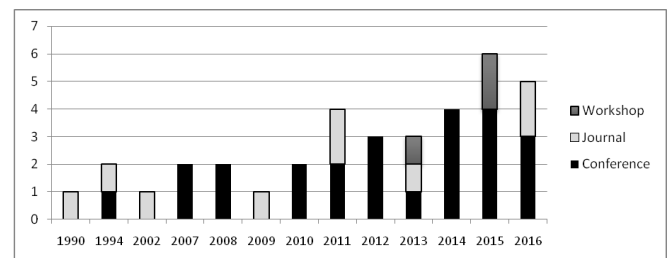


Figure 1. Temporal view of the selected studies

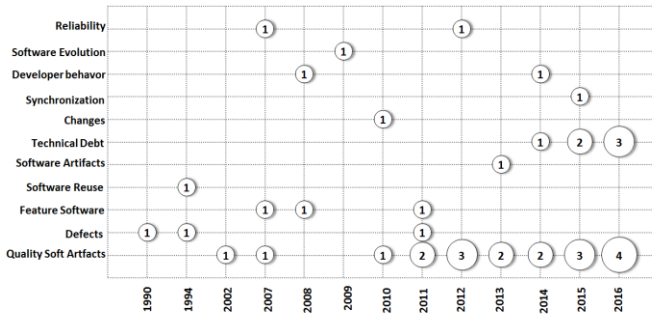


Figure 2. Studies' focus over the years

### A. Purposes and focus of researches (RQ1)

To analyze RQ1 and classify the studies according their purpose and focus, we considered the taxonomy discussed in [4]. We also considered that each study could be categorized into more than one classification. For example, a study can explore comments to *identify* and *comprehend* (purpose) the quality of software artifacts (focus).

**Purpose:** “Comprehension” was considered in the majority of the studies (29 – 80.6%), followed by “identification” (10 – 27.8%). Studies related to the purpose comprehension intend to understand the behavior of a specific attribute by analyzing code comments (e.g., comprehension of the quality of software artifacts or comprehension of defect using code comment analysis). Whereas, studies with the purpose identification aim to detect a specific attribute through comment analysis (e.g., to identify defect). Only two studies had “improvement” as main purpose and all other categories (“classification”, “evaluation”, “localization”, “association”, and “characterization”) were identified in only one study.

**Focus:** Fig. 2 shows the classification of the focus of the studies over the years. The majority of studies focused on “Quality of Software Artifacts” (19 – 52.8%) followed by “Technical Debt” (6 – 16.7%). “Quality of Software Artifacts” appeared nearly every year, but the number of studies with this focus has increased in the last years. The focus on “Technical Debt” only started to be considered more recently, in 2014, and it has also increased in the last years, revealing a new area of investigation.

**Purpose x Focus:** Fig. 3 presents a bubble chart showing the relationship between the facets purpose and focus. We can observe that “Comprehension” of “Quality of Software Artifacts” is the most explored purpose and focus in 16 out of 36 studies. The second most identified purpose and focus is “Identification” of “Quality of Software Artifacts”, “Comprehension” of “Technical Debt”, and “Identification” of “Technical Debt” with 4 studies each.

The results also pointed out that, while the purposes “Evaluation”, “Localization”, “Improvement”, “Classification”, “Association”, and “Characterization” explore just one focus, the purpose “Comprehension” explores almost all focuses identified in this work.

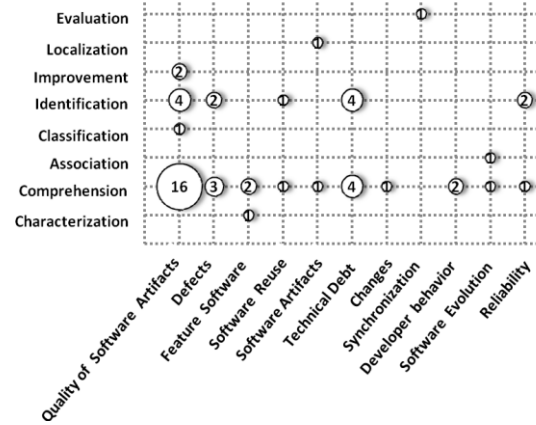


Figure 3. Purpose vs Focus

Regarding technical debt, we can observe that the studies on the area have been performed with the purposes of “Comprehension” or “Identification”.

### B. Techniques to analyze comments (RQ2)

We analyzed the techniques that have been used to mine comments and identified 14 techniques at total. 80.6% of the studies used: Dictionary/Vocabulary (13 studies – 36.1%), NLP (10 – 27.8%), and Statistic/Statistic Analysis/Method Statistic (6 – 16.7%).

Fig. 4 shows the relationship between the study purpose and the technique used to explore comments analysis. Comprehension and Dictionary/Vocabulary were the purpose and technique most used together. Next, NPL was used together with Comprehension and Identification. In the following, we have Identification and Dictionary/Vocabulary. We can also observe that Comprehension and Identification were combined with almost all techniques. On the other hand, some techniques have been used by only one study (e.g. Dynamic analysis and clustering).

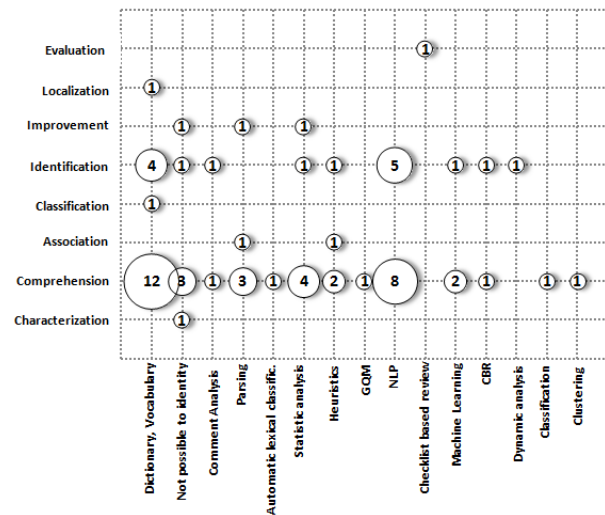


Figure 4. Studies' purpose vs Technique

By analyzing how each technique works, we also identified that 55.6% of them were semiautomatic, followed by automatic (27.8%), manual (8.3%), and the other 8.3% was not determined. A possible reason for the low usage of manual techniques can be the cost to perform a manual analysis in terms of effort and also the fact that the process would be error prone.

C. Tools used to extract, process, or analyze comments (RQ3)

Table 1 presents the tools identified in this work by each step of a mining process (extraction, processing, and analysis). We identified 16 tools used in the extraction step, 14 for the analysis, and 2 for the processing. We did not identify any tool in 50% of the selected studies.

By analyzing Table 1, we can see that most of the tools were presented in recent studies. Another point is that only one tool is used in more than one step (iComment). The others were developed to support only one step of the comment mining process. We also identified that most of the tools are only presented in one study. Therefore, in general, researchers develop new tools as a result of their work. A possible explanation for this is that each study has a specific need (not considered in existing tools) of exploring comments in order to achieve its goals.

This set of information about tools might be useful for researchers and practitioners to develop new approaches or evolve the existing tools with the aim of exploring new perspectives on comment analysis.

D. Empirical evaluations (RQ4)

In this mapping study, we found that 16 papers (44.4%) performed controlled experiments, 8 (22.2%) case studies, 4 (11.1%) exploratory studies, and only 1 paper (2.8%) performed a survey. We also identified 1 ethnographic study (2.8%). The other 6 (16.7%) studies did not present any evaluation. This result indicates that the works in the area of software comments analysis are characterized by the use of empirical methods to assess the proposed approaches.

Fig. 5 represents the evaluation methods performed per purpose. We can observe that controlled experiment and case study are the main research methods used for evaluating “comprehension” and “identification” tasks. “Classification”, “evaluation” and “characterization” were

TABLE 1. LIST OF TOOLS AND MINING STEP

Process Step	Tool	References
<b>Extraction (16)</b>	CLOC, tComments, @Randoop, Prototype Tool, RBG tool, SLOCCount, Jdeodorant, srcML, SSLdoctet, ConQAT, C-REX, Evolizer and ChangeDistiller, eXcomment, iComments, JavaMethodExtractor	S2 (2016), S36 (2012), S14 (2010), S5 (2015), S13 (2015), S1 (2014), S9 (2016), S28 (2013), S3 (2015), S7 (2011)
<b>Processing (2)</b>	iComments, LI Tools	S33(2007), S3(2015)
<b>Analysis (14)</b>	tComments, @Randoop, QDA Analysis tool, RBG tool, CommentCounter, LOCCounter, COMTOR, Evolizer and ChangeDistiller, iComments, Javadocminer, MineHEAD, Stanford Parser, next word prediction tool	S36 (2012), S4 (1994), S5 (2015), S32 (2014), S8 (2012), S10 (2009), S33 (2007), S34 (2011), S17 (2015)

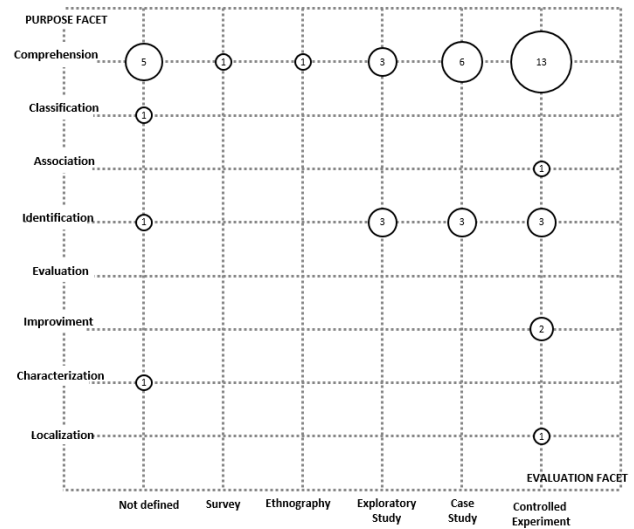


Figure 5. Purpose vs Evaluation methods

the three categories in which researchers have not used empirical methods to evaluate their approaches.

E. Research types (RQ5)

Considering the taxonomy of research types presented in [11], we found that the majority of studies were a “Solution Proposal” paper (19 – 52.8%), 12 were a “Evaluation Research” (33.3%), and 3 were a combination of “Solution Proposal” with “Evaluation Research” (8.3%). Only 1 study performed a “Validation Research” (2.8%) and 1 study used “Opinion Research” (2.8%). Solution proposal is paper where a solution for a problem is proposed. The potential benefits and the applicability of the solution are shown by a small example or a good line of argumentation. On the other side, evaluation research is a type of paper in which techniques are implemented in practice and an evaluation of the technique is conducted.

Fig. 6 presents the distribution of the performed research types over the years. We can observe that while the number of “Solution Proposals” published during the years is stable, there is a rising number of “Evaluation Research” in the last few years (2014, 2015 and 2016) indicating a tendency in the area to perform and report more empirical studies.

Fig. 7 presents the relationship between research type and focus. It shows that the types “Solution Proposal” and “Evaluation Research” were widely adopted by researchers to investigate “Quality of Software Artifacts”. We can also see that the focus “Technical Debt” is the second most explored considering these same types. The other types of research appear only as isolated initiatives.

5. THREATS TO VALIDITY

The results of this systematic mapping may have been affected by some threats to validity, such as:

**Search string:** Even though our search string is broad, it is possible that it did not address some studies. Our search string was designed to find the maximum number of works

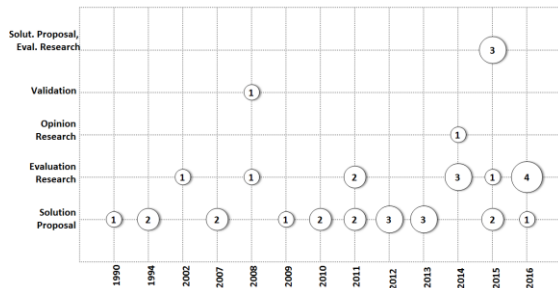


Figure 6. Evolution of the research type over the years

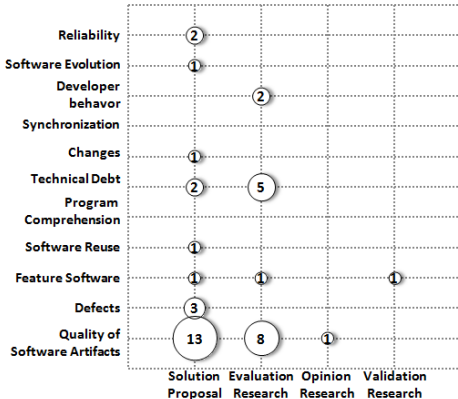


Figure 7. Focus vs Research type

on code comments, but it is possible that it missed studies that did not use the term “comment” in their text. We tried to mitigate this threat by a process of string calibration.

**Selection Bias:** We cannot ensure that all relevant primary studies were selected for this mapping. We addressed this threat during the selection step. We selected each study based on the judgment of the inclusion and exclusion criteria by more than one researcher. However, some studies could still have been categorized incorrectly. To mitigate this, we discussed the study protocol among the researchers to guarantee a common understanding. Moreover, this step was performed by two researchers and, when both disagreed, we considered a third opinion.

**Publication Bias:** It is difficult to ensure that all relevant work was returned as results in the performed searches. To minimize this threat, the main digital libraries in computing were considered.

**Research Questions:** The research questions investigated in this study may not cover all software comments analysis area. To address this risk, the defined questions were analyzed by at least two researchers, one of who acted as an external reviewer of the protocol.

**Data Extraction:** this threat can affect the analysis of selected studies. To reduce this risk, initially, we performed a pilot extraction, aiming to align the researchers' understanding of the research questions. Next, two researchers analyzed each paper to perform the data extraction. A third researcher analyzed the issues on each classification or extracted information to make sure that the extracted data were valid and clear for further analysis.

## 6. CONCLUDING REMARKS

In this paper, we performed a systematic mapping study on software comments analysis. We have extracted and analyzed data from 36 papers. The results can guide researchers in further studies in MSR area focused on code comments. For practitioners, we catalogued a set of tools and techniques to analyze comments with several purposes. This information can help them avoiding reinventing the wheel when developing approaches to extract, processing or analyzing comments.

In our future research agenda, we intend to combine the evidence identified in this work with new theories and empirical studies developed by our group to create new methods and tools to support comments analysis with focus on technical debt identification and management activities.

## ACKNOWLEDGMENT

This work was partially supported by the CNPq Universal grant 458261/2014-9, by the State of Bahia's SECTI-Fraunhofer-UFBA cooperation agreement 2012-1, and by the RESCUER project Grant: 490084/2013- 3.

## REFERENCES

- [1] N.S.R. Alves, T.S. Mendes, M.G. Mendonça, R.O. Spínola, F. Shull, and C. Seaman, Identification and management of technical debt: A systematic mapping study. *Information and Software Technology* v. 70, p.100-121. 2016.
- [2] 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", *J. Syst. Softw.* 80, 2007.
- [3] S. Demeyer, A. Murgia, K. Wyckmans, and A. Lamkanfi, Happy birthday! A trend analysis on past MSR papers, *IEEE Int. Work. Conf. Min. Softw. Repos.*, p. 353–362, 2013.
- [4] M.A.F. Farias, R. Novais, M. Colaço, L.P.S Carvalho, M. Mendonça, and R.O. Spínola, A systematic mapping study on mining software repositories," in 31st ACM/SIGAPP, 2016.
- [5] M.A F. Farias, A.B. Silva, M.G. Mendonça, Spínola, R.O., and M. Kalinowski, "Investigating the use of a contextualized vocabulary in the identification of technical debt: A controlled experiment," in 18Th Int. Conf. on Enterprise Information System, vol. 1, pp. 369–378, 2016.
- [6] M.A.F Farias, M.G. Mendonça, A.B.D. Silva, and R.O. Spínola, A contextualized vocabulary model for identifying technical debt on code comments," in *IEEE 7th Int. Work. on Managing Technical Debt, MTD* 2015.
- [7] B. Kitchenham, Guidelines for performing systematic literature reviews in software engineering. EBSE Technical Report, Software Engineering Group, School of Computer Science and Mathematics, vol. 3 (2). Keele University, Keele, UK. 2007.
- [8] E. Hassan, "The road ahead for mining software repositories," *Front. Softw. Maintenance. FoSM* 2008., pp. 48–57, 2008.
- [9] H. Hemmati, S Nadi, O. Baysal, O. Kononenko, W. Wang, R. Holmes, M. W. Godfrey, "The MSR cookbook: Mining a decade of research," 10th Conference Mining Software. Repository., pp. 343–352, May 2013.
- [10] H. Kagdi, M.L. Collard, and J.I. Maletic, "A survey and taxonomy of approaches for mining software repositories in the context of software evolution," *Journal of Software Maintenance and Evolution: research and practice* pp. 77–131, 2007.
- [11] R. Wieringa, N. Maiden, N. Mead, and C. Rolland, Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. *Requirements Engineering*, pp. 102–107, 2005.

# Process metrics for system quality with specifications' shifts from a bid phase to an operation phase

Noriko Hanakawa  
Information management department  
Hannan University  
Matsubara, Osaka, Japan  
hanakawa@hannan-u.ac.jp

Masaki Obana  
Department of Information Science and Technology  
Osaka Institute of Technology  
Hirakata, Osaka, Japan  
masaki.obana@oit.ac.jp

**Abstract**— Recently, large-scale computer system is important for social human life. Such large-scale system is basically ordered through a competitive bidding. A development company that won the bidding starts developing. After that, customers use the computer system on their business operations. However, customers do not use some functions. Functions proposed in a system proposal in a bid phase are not same functions that customers frequently use in an operation phase. Functions change in various phases. Therefore, we propose process metrics for system quality. The feature of the metrics is target duration from a bid phase to an operation phase. Specification shifts of functions are clear in a simple rule. The metrics quantitatively measure specification shifts. We check active or non-active of each function on each phase. The total number of change of active or non-active of all functions is an important element of the metrics. Moreover, timing of the change of active or non-active of functions is also a significant factor of the metrics. As a result of applying a real project, we found that usage of the uncompleted package software makes development process complicated.

**Keywords**— competitive bidding; usage frequency; active or non-active function; unreasonable price and time; system proposal.

## I. INTRODUCTION

Recently, large-scale computer system has been frequently developed in industry. The large-scale computer system means society infrastructure system such as public transportation system, civic service system of government, banking system, and education system of universities. The system is not only important but also essential for our modern society.

In general, such large-scale computer system has difficulty for development process including requirement analysis, design software, programming techniques, test techniques, and operation and maintenance. In addition, a bidding activity is important for development of large-scale computer system. A bid activity means that a customer organization chooses a system development company using system proposals from development companies. Customer organization selects the best system proposal, then the customer organization orders the new computer system from a development company that made a successful bid. A system proposal includes not only system functions (software and hardware) but also total price, and schedule (due day). Customer organization contracts with the development company on the basis of the prices and schedule

that are provided in the system proposal. Hence, system proposal is very important in a view of company management. If system proposal is inadequate or insufficient, implemented functions may be inadequate, and cost may be unexpectedly increased, and delivery day may be late.

In addition, it is more important to make use of new functions by customers in real business activities. A system consultant says that 64% functions of new system are not used by customers [1]. Of course, new system must run without faults and errors. However, if new system has many unnecessary functions, customer organization wastes much money and time although faults and errors do not occur. Implemented functions are based on a system proposal that made a successful bid in a bidding phase. There are deep relationships between a system proposal and frequency of new function usage.

Therefore, we propose process metrics for system quality from a bid phase to an operation phase. Especially, we focus on specification shift during system development. System specification shifts mean that functions of new system change as new system is developed. For example, Function A is proposed in a system proposal in a bid phase. However, customers realize that Function A is unnecessary when a system design phase. In contrast, Function B is not included in the system proposal. However, at an operation phase, customers desire Function B although Function B is not included design documents and total prices. System specifications are changing on each development phase. If a system proposal is perfect, specification shifts do not occur because all specification are necessary and sufficient. If a system proposal is not perfect, specification shifts frequently occur. Our proposed metrics indicates system quality based on such specification shifts.

In section 2, related work is shown. Section 3 shows the proposed metrics in process model. In section 4, the metrics and model are applied to real computer system development. Section 5 shows summary and future researches.

## II. RELATED WORK

In top level process researches, Edward et al. shows a model of the early estimating/planning stages of a project (EEPS

model) [2]. Because of unclear data in requirement analysis, there were 30% budget error. Jamieson et al. gave a model for pre- and post-contract phases in agile development [3]. These researches mostly cover our research topics. However, environment surrounding development computer system continuously change. Such continuous changes have to be considered in the present day.

On the other hand, in recently, “Chojoryu” for software engineering has been proposed by Muroya [4]. Muroya also provided a method of a contract for software development, and important of customers’ activities in top level software development process. Breiner et al. also discuss requirement engineering in a bidding stage [5]. These researches claim importance of competitive bidding. However, research of competitive bidding process in software engineering just have started. Concrete research results are not described. In addition, Takano et al. show effective bidding strategy in a competitive bidding simulation [6]. Pablo et al. propose effective competitive bidding model in scoring and position probability graph [7]. Also the other researches discussed effective bidding and accuracy of cost estimation. Management fields actively study bidding way, bidding accuracy, and bidding simulation. However, these researches discuss just ways of bidding. Our research target is not bidding system. Our research target is whole development process including competitive bidding in software engineering research field.

### III. PROCESS METRICS FOR SYSTEM QUALITY

#### A. Concept

Fig.1 shows our proposed metrics’ concept. The concept is simple. We focus on specification shifts during a system development period. The feature is the duration of a system development period. The duration includes not only development phases (analysis, design, implement, and test) but also a bid phase (system proposals) and operation phase. In operation phase, we check frequency of each function usage. In Fig.1, “O” means that a function is active. “X” means that a function is not active.

For example, "Function A" in Fig.1 is continuously active from a bid phase to operation phase. That is, customers recognized necessary of "Function A" at a bid phase. Then, developers designed "Function A", implemented "Function A", tested "Function A". Finally, customers frequently use "Function A" in the operation phase. That is the perfect process because the specification does not change. The case is the best process. In contrast, "Function E" is the worst case. Customers needed "Function E" at a bid phase. Then, developers developed "Function E". However customers did not use "Function E". This is the worst process because the specification changed in the final development stage. Although “Function E” was developed at great expense, customers did not actually need “Function E”. Cost and time for “Function E” are wasteful. Moreover, "Function C" is not a so bad case. At first, customers recognized unnecessary of "Function C". However, at the design phase, customers and developers perceived necessity of "Function C". Development of "Function C" starts from the design phase. That is no bad case because design activity is useful to detect the lack of "Function C".

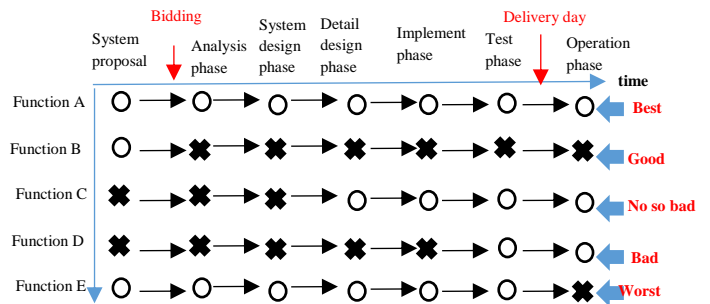


Fig. 1. A concept of process metrics for system quality

In short, our proposed metrics measure specification shifts during a period from a bid phase to an operation phase. If specification shifts do not occur, system quality is high. If specification shifts frequently occur, system quality is low. Especially, in our paper, specification shifts mean that a function’s necessity changes from active "O" to non-active "X". Or, specification shifts mean that a function’s necessity changes from non-active "X" to active "O".

#### B. 3.2 A proposed process basic metric $SQ(F_i)$ for each function

We propose a basic metric  $SQ(F_i)$ .  $SQ(F_i)$  measures a process of a function “ $F_i$ ” specification shift. A software is consists of many functions. The proposed metric  $SQ(F_i)$  is as follows;

$$\text{if } A_i = \emptyset \quad SQ(F_i) = 1 \quad (1)$$

$$\text{if } A_i \neq \emptyset \quad SQ(F_i) = \prod_{k=1}^{aa} \left( \frac{nn - a_k + 1}{nn} \right) \quad (2)$$

$SQ(F_i)$ : process quality of the  $i$ -th function

$aa$ : the total number of element of set  $A_i$

$a_k$ : the  $k$ -th element of set  $A_i$

$A_i = \{a \mid a \text{ is phase No. and } F_{i(n)} \neq F_{i(n+1)}\}$

$F_{i(n)}$ : Mark("O" or "X") of the  $n$ -th phase of the  $i$ -th function

$nn$ : the total number of phase

Based on Fig.2, the  $SQ(F_i)$  is process quality of a function. At first, a meaning of a set “ $A_i$ ” is explained. Elements of the set “ $A_i$ ” are phase numbers that the current phase mark is different from the next phase mark. The phase mark means “O” or “X”, that is, “O” is the function  $F_i$  is active in the phase, “X” is the function  $F_i$  is non-active in the phase. For example, in  $F_2$  of Fig.2, the mark of the phase  $P_1$  is “O” although the mark of the phase  $P_2$  is “X”. The marks of the phase  $P_2$  or later are “X”. Then, the set  $A_i$  is  $\{2\}$ . Moreover, in the case of  $F_{20}$  of Fig.2, there are twice changes the marks. The mark “X” of phase  $P_2$  changes to the mark “O” of phase  $P_3$ , the mark “O” of phase  $P_9$  changes to the mark “X” of phase  $P_{10}$ . Then, the set  $A$  is  $\{3,10\}$ .

Next, we explain equation (1) and equation (2). In the equation (1), a value of  $SQ(F_i)$  is 1 if the set  $A_i$  is an empty set. That is, all marks are same like  $F_1$  of Fig.2. The value of  $SQ(F_i)$  is maximum. In equation (2), a value of  $SQ(F_i)$  is calculated. The values of  $SQ(F_i)$  is basically more than 0 and less than 1. The value of  $SQ(F_i)$  is influenced the position of marks’ changes. If a mark changes at early phase like  $F_2$  of Fig.2, the value of  $SQ(F_i)$  is large. If a mark changes at last phase like  $F_4$  of Fig.2,

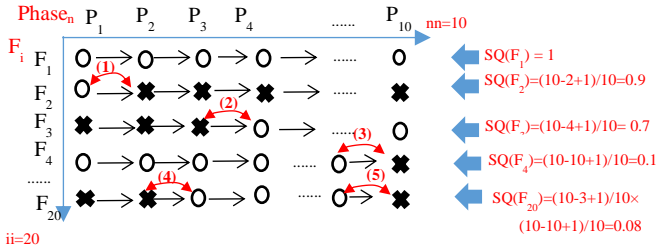


Fig. 2. Calculation sample of the proposed metric  $SQ(F_i)$

the value of  $SQ(F_i)$  is small. In addition, if there are several mark changes like  $F_{20}$  of Fig.2, the value of  $SQ(F_i)$  is smaller and smaller. The meaning of the calculation of equation (2) is that the mark's change at the early stage of development is not so bad, the mark's change at the last stage of development is bad, frequent changes is worse than once mark's change at the last stage.

### C. Features of the proposed metric $SQ(F_i)$

A most feature of the proposed metrics is the first phase and the last phase of Fig.2. The first phase of Fig.2 means a bid phase. The functions of the first phase are described in a system proposal. The system proposal is written by a bid company. The system proposal includes system development price and development time based on the described functions [9]. Customers usually contract with the system development company based on the prices and the time. The functions of the first phase is very important at the view of business management. On the other hand, the last phase is an operation phase. An operation phase usually does not includes a development process. However, the operation phase is important. Basically, frequencies of new functions are counted by operational user logs. If customers frequently use the new functions, the marks at the last phase will be "O". If customers do not use the new functions, the marks at the last phase will be "X". Of course, trial use or test use of the new function is eliminated from the frequencies. The new functions have to be useful for customers' business processes.

The first phase mark and the last phase mark are very important. Fig.3 shows a summary of process pattern on the first phase mark and the last phase mark. A development process is good when a mark of the first phase is same to a mark of the last phase. A good system proposal may lead to a good process. That is, a function is proposed in a system proposal, then, the function is implemented in development phases. At last customers frequently use the function. The proposed function in the system proposal is valuable. However, if a mark of the first phase is different from a mark of the last phase, the development process has various problems. When a mark of the first phase is "O" although a mark of the last phase is "X", customers did not need the function although the function is proposed. The cost of the unnecessary function is added to the development price in the system proposal. Customers wasted the cost of the function. In contrast, when a mark of the first phase is "X" although a mark of the last phase is "O", customers need the function although the function is not proposed. In this case, customers notice the necessity of the function halfway through their development. The cost of the necessary function is not included the development price in the system proposal. Therefore, customers

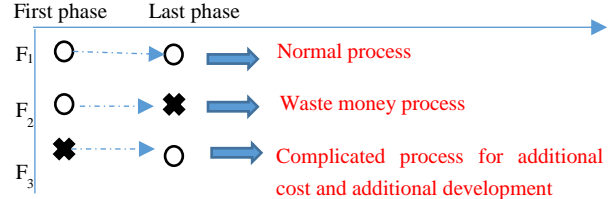


Fig. 3. Various process patterns on the first and the last phase

need additional budget for the new function. In addition, developers have to review influence of adding the new function to the original functions. The development process may be complicated, customers may raise money in order to add the function.

In this way, the proposed metric have a potential of significant problems such as process complexity and development cost's increment.

### D. System metrics for all functions

$SQ_{avg}$  is an average value of all  $SQ(F_i)$ . Of course, the best value of  $SQ_{avg}$  is 1, worst value of  $SQ_{avg}$  is extremely near zero.

$$SQ_{avg} = \text{AVG}(SQ(F_1), SQ(F_2), \dots, SQ(F_{ii})) \quad (3)$$

ii: the total number of function

$$SQ_{normal} = \frac{\text{the number of function with normal process}}{\text{the total number of function}} \quad (4)$$

$SQ_{normal}$  means percentage of the number of functions with normal process in the total number of functions. Normal process means that all marks are "O". That is, the function was included in the system proposal, then the function was implemented. At last, customers frequently use the function on business operation. Of course, the best value of  $SQ_{normal}$  is 1, the worst value of  $SQ_{normal}$  is 0.  $SQ_{avg}$  and  $SQ_{normal}$  may be one of guides of system quality from a bid phase to an operation phase.

## IV. APPLICATION TO A REAL PROJECT

### A. Target project

The target project is an educational system development project. The system request was provided at November of 2015, the competitive bidding was held at April of 2016, starting design phase was at May of 2016, starting implement phase was at September of 2017, and starting operation was at April of 2017. Therefore, the number of phase (the value of  $nn_i$  of formula (2)) is 4. The system had 82 functions, then the number of function (the value of  $ii$  of formula (3)) is 82.

### B. Checking active functions and non-active functions

Judgement of active or non-active is based on objective data such as development documents and operation log. In the system proposal phase, we checked the system proposal documents. If a function was described in the system proposal, the function is active. If a function was not described in the system proposal, the function is non-active. As same way, we checked all design documents, and we confirmed the all implementations. In the operation phase, user operation logs of the system is helpful. The



user operation log is recorded customers' operations. A huge amount of operation logs have been accumulated. If any users used a function, the function is active. No one used a function, the function is non-active.

### C. Patterns of active "O" and non-active "X"

Fig.4 shows the results of active "O" and non-active "X". There are 7 patterns of the active and non-active on the target project. Pattern A (OOOO) is normal and the best process pattern. The number of functions with Pattern A is 36. Pattern B (OOOX) is a waste process because users do not use functions although the functions were developed consuming cost and time. The number of function with Pattern B is 12. Pattern C (OOXX) is a bad process. On the way of implement phase, developers confirmed unnecessary of the functions. Pattern D (OXXX) is no good process. However, the Pattern D is better than the Pattern C because developers knew unnecessary of the functions at the design phase. Pattern E (XOOO) is a problematic process in cost management. System proposal was not included the functions, however, the functions were developed in the development phase. The cost was not included in the system proposal. Pattern F (XXOO) is a troubled process because developers and customers confirmed unnecessary of the functions at implement phase. That is too late. Cost and time may be in serious trouble. Pattern G is the worst process. No one knows necessity of the functions until system running. Obviously, cost and time are shorted in the Pattern G.

### D. Measuring metrics

We measured the value of  $SQ(F_i)$  in equation (1) and (2),  $SQ_{avg}$  in equation (3),  $SQ_{normal}$  in equation (4). Fig.5 shows  $SQ_{normal}$ . The value of  $SQ_{avg}$  is 0.747, the value of  $SQ_{normal}$  is 0.44. In the target project, only 44% functions had the normal process "OOOO". The 56% functions had any changes between active and non-active. In short, the functions proposed in the system proposal were not necessary and sufficient. The 30% functions in the system proposal were not necessary, the 26% functions that were not proposed in the system proposal were necessary.

In this way, functions proposed in a system proposal are uncertain information. Development price and time are decided based on such uncertain information in a system proposal. Then, the price and the time often become a basic information for making a contract with a development company. That is an unreasonable procedure. This unreasonable procedure will be discussed in the following section.

## V. SUMMARY

We proposed process metrics  $SQ(F_i)$ ,  $SQ_{normal}$ ,  $SQ_{avg}$  for system quality from a bid phase to an operation phase. The concept of the metrics is simple. The best case is that functions are always active in all phases. The worst case is that active functions change to non-active functions at the last phase. The metrics were applied to a real project. As a result, only 44% of functions have the best process like pattern "OOOO". 15% of functions have the worst process like pattern "OOOX".  $SQ_{avg}$  is 0.747. The 30% functions in the system proposal were not necessary, the 26% functions that were not proposed in the system proposal were necessary.

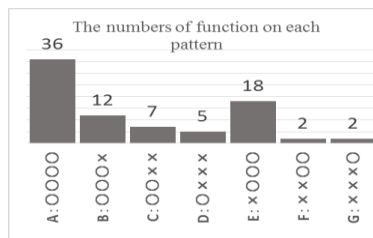


Fig. 4 The numbers of functions on each pattern

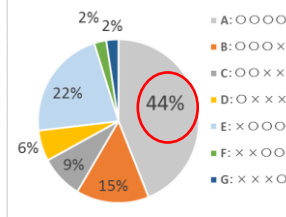


Fig. 5. A value of  $SQ_{normal}$

In future, we will apply the metrics to several projects and several organizations. The features of projects and organizations will be clear using values of the metrics. Especially, system development including package software [8] will be discussed. Moreover, we will achieve a concrete way of estimating price and time in a system proposal [10] using values of the metrics on each organization.

## ACKNOWLEDGEMENTS

This work was partially supported by JSPS KAKENHI Grant Number JP26330093.

## REFERENCES

- [1] Rising sun consultation company HP <http://risingsun-system.biz/pdca-cycle-it-investment>, last accessed 2017/6/3.
- [2] J.S.Edwards, T.T.Moores, "A conflict between the use of estimating and planning tools in the management of information systems", *European Journal of Information Systems* 3(2), 1994, pp.139-147.
- [3] D. Jamieson, K. Vinsen, G. Callender, "Agile procurement: New acquisition approach to agile software development", *Proceedings of 31st EUROMICRO Conference on Software Engineering and Advanced Applications*, 2005, pp. 266-273.
- [4] IPA Software Engineering Center: A "Cho-joryu" Approach. In *The Seventeen Principles for System Development*. <http://www.ipa.go.jp/english/sec/reports/20120502.html>, last accessed 2017/6/3.
- [5] K.Breiner, M.Gillmann, A. Kalenborn, C. Müller, "Requirements Engineering in the Bidding Stage of Software Projects – A Research Preview", In: Fricker, Samuel A., Schneider, Kurt (eds.) *REFSQ 2015 21st International Working Conference, LNCS*, vol. 9013, pp.270-276, Springer (2015).
- [6] Y.Takano, N.Ishii, M. Muraki, "A sequential competitive bidding strategy considering inaccurate cost estimates", *Omega*, vol. 42, issue 1, 2014, pp.132-140.
- [7] P.Ballesteros-Pérez, M. C. González-Cruz, A. Cañavate-Grimal, "On competitive bidding: scoring and position probability graphs", *International Journal of Project Management*, vol. 31, issue 3, 2013, pp.434-448.
- [8] N.Kataoka, H.Koizumi, K.Takasaki, N.Shiratori, "Remote joint application design process using package software", In *ICOIN-12 Twelfth International Conference on Information Networking*, 1998, pp.495-500.
- [9] C.Lopez-Martin, C. Isaza, A. Chavoya, "Software development effort prediction of industrial projects applying a general regression neural network", *International journal Empirical Software Engineering*, vol.17, num.6, 2012, pp.738-756.
- [10] M. Usman , E. Mendes , F. Weidt , R. Britto, "Effort estimation in agile software development: a systematic literature review", *Proceedings of the 10th International Conference on Predictive Models in Software Engineering*, 2014.

# Exploratory Recommender Systems Based on Reinforcement Learning for Finding Research Topic

Li Yu

School of Information  
Renmin University of China  
Beijing, China  
buaayuli@ruc.edu.cn

Zhuangzhuang Wang

School of Information  
Renmin University of China  
Beijing, China  
w1194690657@ruc.edu.cn

Hongrun Xin

International School  
Beijing University of Posts and  
Telecommunications  
Beijing, China  
xinhongrun@bupt.edu.cn

Wei Zhang

School of Information  
Central University of Finance and  
Economics  
Beijing, China  
kddzw@163.com

Xuefeng Li

School of Information  
Central University of Finance and  
Economics  
Beijing, China  
xuefeng@cufe.edu.cn

Haiyan Wang

School of Information  
Beijing Forestry University  
Beijing, China  
haiyan@bfu.edu.cn

**Abstract**—Traditional recommender systems try to select few items from some candidate items to users. Unfortunately, a user often hope recommender system help him to make a decision or finish a task based on his uncertain preference. For example, a researcher could hope recommender system to help him to find an advanced research topic by recommending literatures paper and refining his research interest and. In this paper, we develop an exploratory paper recommender system based on reinforcement learning, which can navigate a researcher to identify research topic by recommending papers continuously. In order to refine and focus user's research preference, as a reinforcement learning method, Multi-Armed Bandit (MAB) is employed for navigating recommendation paper. And two improved MAB methods are proposed, including  $\epsilon$ -Greedy Stochastic Perturbation ( $\epsilon$ -Greedy-SP) and Continuous Upper Confidence Bound (Con-UCB). Also, a weighted-LDA method is proposed for constructing the topic tree. A prototype system is developed and used to make experiments. Empirical research is made to analyze the change process of users' preference. The results show that the system is very effective for focusing and finding research topic.

**Keywords**- Recommender Systems; Multi-Armed Bandit; Reinforcement Learning; Research Topic

## I. INTRODUCTION

Along with the development of Internet and the explosive growth of information, recommender system has become a hot research issue in the past ten years. However, the goal of most recommender systems is just to recommend some to user from a lot of candidate items. A few researches are made on recommender system oriented to task. For example, recommender systems help a user to decide a research topic. It is a fact that a researcher often needs to read a lot of literatures

in order to know state of the art and to find a research topic. Although lots of researches on paper recommendation are made, there are a few recommender systems whose goal is to help a user to find a research topic.

In this paper, we explore the application of reinforcement learning to exploratory recommender systems, whose goal is to help user to find research topic. The contributions of this paper are as follows: 1) A weighted LDA (Latent Dirichlet Allocation) method is proposed to build multi-layer topic tree; 2) Two exploratory recommendation methods based Multi-Armed Bandit (MAB) are proposed, respectively including  $\epsilon$ -Greedy Stochastic Perturbation ( $\epsilon$ -Greedy-SP) and Continuous Upper Confidence Bound (Con-UCB) ; 3) A paper recommender system oriented to finding research topic is developed, and its performance is tested and evaluated.

This paper is organized as following. Next, we survey the state of art on reinforcement learning and recommender system. In section 3, overall framework of developed paper recommender system is proposed. In section 4, two recommendation methods based on MAB are presented. In section 5, the experiments are made and system performance is tested. Finally, conclusions and future research is discussed.

## II. RELATED WORK

Finding a research topic often starts by reading a large number of papers, so paper recommender system is very useful for scholars to select their research fields. Tang et al. used focused collaborative filtering which is added with users clustering for paper recommendations [1]. Lee used collaborative filtering methods to develop a paper recommender system [2]. Beel et al. compared several different evaluations for research paper recommendation [3]. Melnick focused on how to display a research paper [4], and the result showed that organic recommendations performed better than commercial recommendations.

DOI reference number: 10.18293/SEKE2018-063

The research is supported by National Natural Science Foundation of China (No.71271209, No.71331007).

Personalized recommendation means to recommend objects, such as goods, music, websites or papers, based on analysis of unique user through the recommendation process. In the past ten years, machine learning methods have been introduced to recommendation field. Wang J. et al. [5] combined Convolutional Neural Network and Wide & Deep model to recommend articles and applied attention model to solve the sequential problem. Tajima A et al. [6] used Factorization Machine to extract features and Gated Recurrent Unit to recommend news for large amount of users. Yang C. et al. [7] combined CF and semi-supervised learning to recommend POIs. However, all these methods assume that there are some labeled data for filling the matrix (CF methods) or training the network (NN methods), so the cold start problem is still not solved very well and the fluctuation of users' preferences cannot be evidently detected.

The conception of reinforcement learning is firstly proposed by Barto [8], who defined reinforcement learning as a goal-oriented learning from interaction. Multi-Armed Bandit problem is a classical problem in reinforcement learning, and the research about MAB has lasted for decades. The latest achievements are as follows. Xu [9] used MAB models to balance exploiting user data and protecting user privacy in dynamic pricing. Shahrampour [10] proposed a new algorithm for choosing the best arm of MAB, which outperforms the state-of-art. Lacerda [11] proposed an algorithm named as Multi-Objective Ranked Bandits for recommender systems.

### III. SYSTEM DESIGN OF EXPLORATORY RECOMMENDER SYSTEM

#### A. System Overview

The overview of our proposed recommendation method is shown in Fig.1. It includes three key modules, respectively *Topic Tree Building Module*, *Recommendation Module* and *User Preferences Updating Module*.



Figure 1. Exploratory Recommendation Process

In the topic tree building module, firstly, all literatures are separated into  $N$  topics of 1<sup>st</sup> layer based on Latent Dirichlet Allocation (LDA) method and get the Distribution Matrix ( $DM$ ) and Belong Matrix ( $BM$ ) of 1<sup>st</sup> layer. Then, a weighted-LDA method proposed in section 3.2 is used to subdivide each topic of 1<sup>st</sup> layer into  $N$  topics, so we get  $N*N$  topics at 2<sup>nd</sup> layer. We generalize weighted-LDA to more layers and get  $DM$  and  $BM$  of all layers. So, a topic tree is built, in which every non-leaf node has  $N$  child nodes. The process will be described in detail in Section III(B).

Recommendation module is the core module of our system. For a new user, we select papers from different topics at 1<sup>st</sup> layer randomly as the recommendation of 1<sup>st</sup> round. Then we obtain user's preference distribution of  $N$  topics at 1<sup>st</sup> layer according to feedback (ratings to the recommended papers). Afterwards, recommendations are carried out in the following steps.

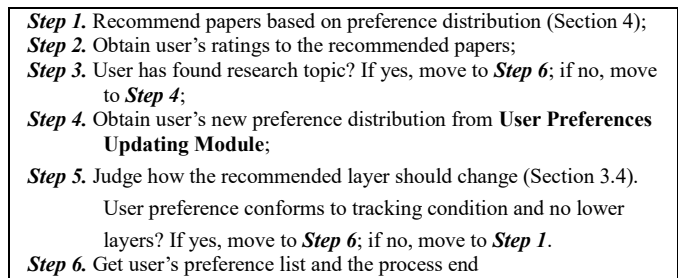


Figure 2. Process of Recommendation Module

The function of user preferences updating module is to update user preferences of different layers according to user's ratings to the recommended papers, and its procedure is detailed in Section 3.3.

#### B. Topic Modeling

The structure of the topic tree is shown in Fig. 3.

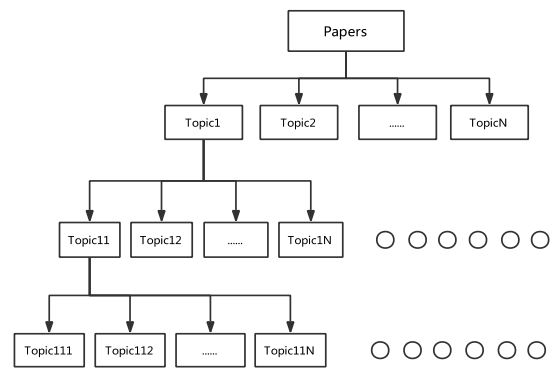


Figure 3. Structure of Topic Tree

**Topic modeling of 1<sup>st</sup> layer based on LDA.** Latent Dirichlet Allocation (LDA) is a topic discovery model for documents. According to LDA, each word from a paper obeys the following process: select a topic related to this paper with some probability and select a word from the selected topic with some probability. The process of LDA can be explained as factorizing the known Document-Word Matrix.

So, each paper  $p_m$  is mapped to a  $N$ -dimensional vector using basic LDA method, represented by  $(d_{m,T_1}, \dots, d_{m,T_n}, \dots, d_{m,T_N})$ . And we will get topic distribution matrix (denoted by TDM) at 1<sup>st</sup> layer, shown in Table 1.

TABLE I. TOPIC DISTRIBUTION MATRIX (1ST LAYER)

	$T_1$		$T_n$		$T_N$
$p_1$	$d_{1,T_1}$		$d_{1,T_n}$		$d_{1,T_N}$
..	...		...		...
$p_m$	$d_{m,T_1}$		$d_{m,T_n}$		$d_{m,T_N}$
..	...		...		...
$p_M$	$d_{M,T_1}$		$d_{M,T_n}$		$d_{M,T_N}$

Based on topic distribution matrix, we can determine the topic that paper  $p_m$  belongs to as following,

$$T_m^* = \text{indexof}(\text{argmax}\{d_{m,T_i} | i = 1, \dots, N\}) \quad (1)$$

Thus, we get a belong-to matrix at 1<sup>st</sup> layer (denoted by  $BM(1^{\text{st}} \text{ layer})$ ), where each row of  $BM(1^{\text{st}} \text{ layer})$  is a  $N$ -dimensional vector contains of a one and  $N-1$  zeros. It will be used in  $\text{randSelect}()$  function in recommendation methods.

**Subtopic modeling based on weighted-LDA.** We describe the process of 2<sup>nd</sup> layer and it's easy to promote to all the lower layers. At first, which papers should be brought into the subdivision of which topics is determined as follows: For the distribution vector of

$$p_m = (d_{m,T_1}, \dots, d_{m,T_n}, \dots, d_{m,T_N}) \quad (2)$$

We sort it in descending order and get an adjusted vector

$$(d_{m,T_{adj_1}}, \dots, d_{m,T_{adj_n}}, \dots, d_{m,T_{adj_N}})$$

Then accumulate the vector until the sum is greater than a threshold  $\theta$ . And we can get the related topic of paper  $p_m$ ,

$$\text{Topic}(p_m) = \{T_{adj_1}, \dots, T_{adj_{count}}\} \quad (3)$$

TABLE II. AN EXAMPLE OF SUBDIVISION MEMBER DETERMINATION

$\text{paper\_id}$	$d_{T_1}$	$d_{T_2}$	$d_{T_3}$	$d_{T_4}$	$d_{T_5}$
$p_1$	0.072	0.113	0.496	0.236	0.083
$p_2$	0.129	0.041	0.062	0.728	0.041
$p_3$	0.192	0.137	0.253	0.283	0.136

For example, as shown in Table 2, when  $\theta = 0.6$ ,  $p_1$  is related to topic  $T_3$  and  $T_4$ ,  $p_2$  is related to  $T_4$ ,  $p_2$  is related to  $T_4$ ,  $T_3$  and  $T_1$ . After determining the participants, we use weighted-LDA for topic discovery. In the topic discovery process of basic LDA, every word in every document is not separated by the conception of weight, and is considered just as count 1. We assume that  $p_1, p_2$  are both participants of subdivision of  $T_1$ , but  $d_{1,T_1} = 0.8$  and  $d_{2,T_1} = 0.2$ . In this situation,  $p_1$  and  $p_2$  should apparently be distinguished, because  $p_1$  belongs to  $T_1$  more than  $p_2$  does. The rule is defined as: the more a paper belongs to a topic, the higher weight its words will get in subdividing the topic. This is the thought of weighted-LDA. In subdivision of  $T_n$ , for every

participant  $p_m$ , its Document-Word Matrix is multiplied by  $d_{m,T_n}$ , and this adjusted matrix will be the input of LDA. In this way, we will get  $N$  distribution matrices and  $N$  belong-to matrices, denoted by  $DM(2^{\text{nd}} \text{ layer})$  and  $BM(2^{\text{nd}} \text{ layer})$ .

### C. User Preference Updating

We map the user's ratings to the recommended papers to user's scores to topics through DM, which is also denoted as  $d$ . Figure 4 shows the updating process. At the beginning of  $t^{\text{th}}$  round, assume the current layer is  $L$ , which means the user has got clear preference from layer 1 to  $L-1$ , we will get a preference list with the length of  $L-1$ , denoted as  $pre^{(t-1)}$ .  $pre_i^{(t-1)}$  represents the most preferred topic at layer  $i$ . If  $pre = [2,1]$ , it means the user likes  $T_{21}$  at the end of  $(t-1)^{\text{th}}$  round and the current recommendations are among the subtopics of  $T_{21}$ , which are  $T_{211} \sim T_{21N}$ .

Correspondingly, we maintain a  $L$ -length list named as  $US^{(t-1)}$ , which stores the user's preference scores to different layers from 1 to  $L$ .  $US_i^{(t-1)}$  represents the scores to the topics at  $i^{\text{th}}$  layer. It should be noticed that  $User\_score$  just stores the scores alongside the user's preference path, so every element in  $US$  is an  $N$ -dimensional vector and  $User\_score$  should be explained in conjunction with  $pre$ . When  $pre = [2,1]$ ,  $US_1$  represents the scores to  $T_1 \sim T_N$ ,  $User\_score_2$  represents the scores to  $T_{21} \sim T_{2N}$ , and  $User\_score_3$  represents the scores to  $T_{211} \sim T_{21N}$ .  $Paper^{(t)}$  is the union of the recommended papers at  $t^{\text{th}}$  round, which consists of  $K$  papers denoted as

$$\{Paper_1^{(t)}, Paper_2^{(t)}, \dots, Paper_k^{(t)}, \dots, Paper_K^{(t)}\}$$

The distribution of  $Paper_k^{(t)}$  at  $i^{\text{th}}$  layer is  $(d_{k(t),T_{pre_1pre_2\dots pre_{i-1}1}}, \dots, d_{k(t),T_{pre_1pre_2\dots pre_{i-1}N}})$ , where  $k(t)$  means the id of  $Paper_k^{(t)}$ . At the situation of  $pre = [2,1]$ , the distributions of  $Paper_k^{(t)}$  from 1<sup>st</sup> layer to 3<sup>rd</sup> layer are  $d_{k(t),T_1} \sim d_{k(t),T_N}$ ,  $d_{k(t),T_{21}} \sim d_{k(t),T_{2N}}$  and  $d_{k(t),T_{211}} \sim d_{k(t),T_{21N}}$ . User's ratings to the  $K$  papers are  $Rating^{(t)} = \{R_1^{(t)}, R_2^{(t)}, \dots, R_k^{(t)}, \dots, R_K^{(t)}\}$ , and  $ac$  is the attenuation coefficient.

<p><b>Input:</b> <math>d</math> (the shorthand of <math>DM</math>)  <math>pre^{(t-1)}</math> (user's preference path at <math>(t-1)^{\text{th}}</math> round)  <math>U^{(t-1)}</math> (user's preference score at <math>(t-1)^{\text{th}}</math> round)  <math>Rating^{(t)}</math> (user's ratings to recommended papers at <math>t^{\text{th}}</math> round)  <math>ac</math> (attenuation coefficient)</p> <p><b>Output:</b> <math>US^{(t)}</math></p> <p>For <math>k = 1</math> to <math>K</math>  <math>\widehat{R}_k^{(t)} = R_k^{(t)} - 2.5</math>  End for  <math>L = \text{length}(pre^{(t-1)}) + 1</math>  For <math>l = 1</math> to <math>L</math>  <math>Temp\_score_l^{(t)}</math>  <math>= (\sum_{k=1}^K \widehat{R}_k^{(t)} d_{k(t),pre_1pre_2\dots pre_{l-1}1}, \dots, \sum_{k=1}^K \widehat{R}_k^{(t)} d_{k(t),pre_1pre_2\dots pre_{l-1}N})</math>  <math>US_l^{(t)} = US_l^{(t-1)} * ac + Temp\_score_l^{(t)} * (1 - ac)</math>  End for  Return <math>US^{(t)}</math></p>
---

Figure 4. Preference Updating

It is important to notice that  $R_k^{(t)}$  is limited in  $[0,1,2,3,4,5]$  and  $\widehat{R}_k^{(t)}$  is a revise of  $R_k^{(t)}$ . Without this process, if the score of a specific topic is close to 0, we can hardly distinguish that whether the user dislikes the topic or there are not enough recommendations from this topic, and these two situations cannot be confused. After the revise, when the score is close to 0, the confusion is the user neither likes nor dislikes the topic or not enough chances for the topic, and it's acceptable. The key is we can easily separate the topics which are preferred by the user (a relatively large positive number) and those topics the user dislikes (a relatively small negative number).

#### D. Backtracking and Tracking

Backtracking condition indicates that user's preference becomes not so clear at the upper layer, so the recommendation process should trace back to upper layer. On the contrary, tracking condition shows that user's preference at current layer is clear enough and the recommendation process should traverse down alongside the topic tree. The two conditions are defined as follows: Backtracking condition. At the end of  $t^{\text{th}}$  round, if  $L > 1$  and

$$\max_{n=1,\dots,N} US_{L-1,n}^{(t)} - \text{secondMax}_{n=1,\dots,N} US_{L-1,n}^{(t)} < \theta_2 \quad (4)$$

which means the score of the most preferred topic at upper layer is not obviously larger than the second one, we pop the last element of  $pre$  and the last  $N$ -dimensional vector of  $User\_score$ , and let  $L = L - 1$ . Tracking condition. At the end of round  $t$ , if  $L < \text{Max\_layer}$  and

$$\max_{n=1,\dots,N} US_{L,n}^{(t)} - \text{secondMax}_{n=1,\dots,N} US_{L,n}^{(t)} > \theta_3 \quad (5)$$

which means the difference between the score of the most preferred topic at current layer and the score of the second one is clear enough, the recommendation process remote to lower layer, and  $pre$  should be added with  $\text{indexof}(\max_{n=1,\dots,N} US_{L,n}^{(t)})$  and we add an  $N$ -dimensional zero vector to the tail of  $US$ ,  $L = L + 1$ ,  $\theta_2$  and  $\theta_3$  are the thresholds of the two conditions.

#### E. System Interfaces

Several interfaces of our system are shown as follows. Fig.5 shows the login interface. Fig.6 is the main recommendation interface, which contains of the information of recommended papers and the buttons for user to give the rating. Fig. 7 shows the word cloud generated after each round of recommendation.

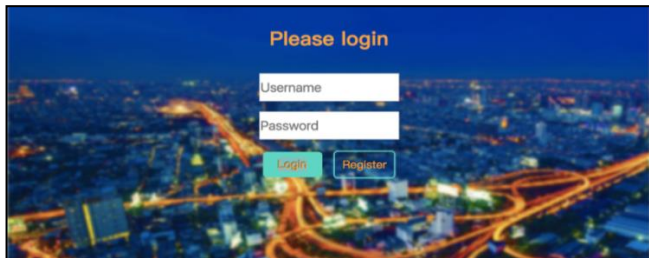


Figure 5. Login Interface of System

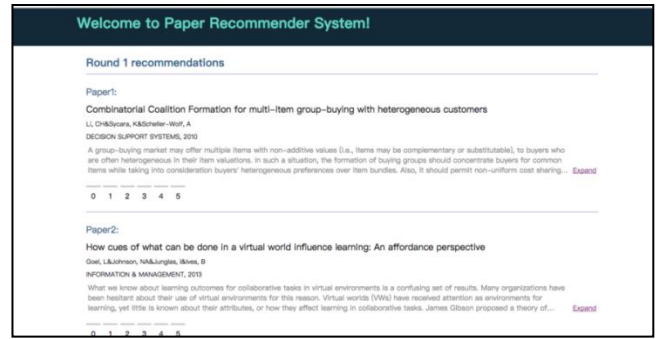


Figure 6. Recommendation Interface of System



Figure 7. Generated Word Cloud for Each Round

### IV. EXPLORATORY RECOMMENDATION METHODS BASED ON MAB

#### A. $\epsilon$ -Greedy Stochastic Perturbation ( $\epsilon$ -Greedy-SP)

$\epsilon$ -Greedy is a classic method of solving MAB model. Give a threshold named as  $\epsilon$ , and generate a random number named as  $\xi$ , if  $\xi > \epsilon$ , the arm with the highest average profit will be chosen, when  $\xi \leq \epsilon$ , a random arm will be selected. We apply  $\epsilon$ -Greedy method to the situation of exploratory recommendation in the following two ways.

**Classic  $\epsilon$ -Greedy.** Fig.8 shows the process of  $\epsilon$ -Greedy.  $L$  is the current layer, if the generated random value  $a > \epsilon$ , the most preferred topic at current layer will be chosen as  $T^*$ , else if  $a \leq \epsilon$ , one of the  $N$  topics at current layer will be chosen randomly as  $T^*$ . The purpose of  $\text{randSelect}()$  function is to determine related topics  $T^*$  which a paper which belongs to.

<p><b>Input:</b> <math>b</math> (the shorthand of <math>BM</math>),  <math>pre^{(t-1)}</math> (user's preference path at <math>(t-1)^{\text{th}}</math> round),  <math>User\_score^{(t-1)}</math> (user's preference score at <math>(t-1)^{\text{th}}</math> round),  <math>\epsilon</math> (threshold of <math>\epsilon</math>-Greedy)</p> <p><b>Output:</b> <math>RecResult^{(t)}</math></p>
<pre> RecResult<sup>(t)</sup> = {} L = length(pre<sup>(t-1)</sup>) + 1 For k = 1 to K   a = Rand();   If a &gt; ε Then T* = T<sub>pre<sub>1</sub>pre<sub>2</sub>...pre<sub>L-1</sub>{n} max<sub>n=1,2,...,N</sub> User_score<sub>L,n</sub><sup>(t-1)</sup>}   Else T* = T<sub>pre<sub>1</sub>pre<sub>2</sub>...pre<sub>L-1</sub>random(1,N)</sub>   RecResult<sup>(t)</sup> ← randSelect(b<sub>m,T*</sub> = 1) End for Return RecResult<sup>(t)</sup> </sub></pre>

Figure 8. Classic  $\epsilon$ -Greedy Algorithm

**$\epsilon$ -Greedy\_SP.** Based on classic  $\epsilon$ -Greedy algorithm, we add a stochastic perturbation to the current preference scores in  $\epsilon$ -Greedy\_SP for catching user's preference as soon as possible. As show in Fig.9, the function  $randVector(\epsilon)$  is used to generate a  $N$ -dimensional vector, consist of one  $\epsilon$  and  $N-1$  zeros. The design of  $randVector(\epsilon)$  ensures the exploration of the recommendation process.

<p><b>Input:</b> <math>b</math> (the shorthand of <math>BM</math>)  <math>pre^{(t-1)}</math> (user's preference path at <math>(t-1)^{th}</math> round),  <math>US^{(t-1)}</math> (preference score at <math>(t-1)^{th}</math> round)  <math>\epsilon</math> (threshold of <math>\epsilon</math>-Greedy)</p> <p><b>Output:</b> <math>RecResult^{(t)}</math></p> <p><math>RecResult^{(t)} = \{\}</math>  <math>L = length(pre^{(t-1)}) + 1</math>  For <math>k = 1</math> to <math>K</math>  <math>US'_{L,n}{}^{(t-1)} = US_{L,n}{}^{(t-1)} + randVector(\epsilon)</math>  <math>T^* = T_{pre_1, pre_2, \dots, pre_{L-1}\{n\} \max_{n=1,2,\dots,N} US'_{L,n}{}^{(t-1)}}</math>  <math>RecResult^{(t)} \leftarrow randSelect(b_{m,T^*} = 1)</math>  End for  Return <math>RecResult^{(t)}</math></p>
---

Figure 9.  $\epsilon$ -Greedy\_SP Algorithm

**The design of  $\epsilon$ .** For any method derived from  $\epsilon$ -Greedy, the value of  $\epsilon$  is apparently an important aspect. We believe that there is a close connection between the value of  $\epsilon$  and user's preference. When the preference is not so clear,  $\epsilon$  should be relatively large for a greater degree of exploration. On the contrary, with the difference between the preferred topics and the disliked ones is large enough,  $\epsilon$  should decrease to a relatively small value. We design  $\epsilon$  as follows in our system.

$$\epsilon = 1 - S(gap) \quad (6)$$

$$gap = \max_{n=1,\dots,1N} US_{L,n}{}^{(t-1)} - \text{secondMax}_{n=1, \text{on}N} US_{L,n}{}^{(t-1)} \quad (7)$$

where  $S()$  is the Sigmoid function.  $S()$  can be stretched or shrunk in both axes according to the actual situation.

#### B. Continuous Upper Confidence Bound (Con-UCB)

The classic Upper Confidence Bound (UCB) method is to express exploitation and exploration as two parts of a total score. The basic formula of UCB is  $Score_i = \overline{Gain}_i(t) + \sqrt{\frac{2 \ln t}{T_{i,t}}}$ , in which the first part represents the average gain of an arm and the second part represents the possibility of the arm, where  $t$  is the round numbers,  $\overline{Gain}_i(t)$  denotes the average gain of arm  $i$  and  $T_{i,t}$  is the times that arm  $i$  used. For exploratory recommendation, UCB method needs to combining the average *User score* and how many times the topics are recommended which is calculated through  $BM$ . For example, if a recommended paper belongs to  $T_1$  according to  $BM$  ( $1^{st}$  layer), the  $T_{i,t}$  of  $T_1$  in the basic UCB formula will be added by 1.

**Continuous-UCB.** Continuous-UCB is a UCB based method by considering the weights of topics. The difference in Continuous-UCB is that we alternate  $BM$  in classic UCB with  $DM$ , that is to say in Continuous-UCB, we sum the

distributions on  $T_1$  of all the recommended paper as the  $T_{i,t}$  of  $T_1$  in the UCB formula. The process is shown in detail in Figure 10. This method also gives an idea to the situation that there are complicated matches between recommendation items and categories.

<p><b>Input:</b> <math>d</math> (the shorthand of <math>DM</math>)  <math>b</math> (the shorthand of <math>BM</math>)  <math>pre^{(t-1)}</math> (user's preference path at <math>(t-1)^{th}</math> round)  <math>US^{(t-1)}</math> (user's preference score at <math>(t-1)^{th}</math> round)  <math>round</math> (rounds made at current layer),  <math>Paper</math> (all the recommended papers)</p> <p><b>Output:</b> <math>RecResult^{(t)}</math></p> <p><math>RecResult^{(t)} = \{\}</math>  <math>L = length(pre^{(t-1)}) + 1</math>  For <math>k = 1</math> to <math>K</math>  For <math>n = 1</math> to <math>N</math>  <math>UCB\_score_{L,n}{}^{(t)} = User\_score_{L,n}{}^{(t-1)} + \sqrt{\frac{2 * \ln(K * round)}{\sum_{time=t-round}^{t-1} \sum_{num=1}^K d_{paper_{num}^{(time)}, T_{pre_1, pre_2, \dots, pre_{L-1}, n}}}}</math>  End for  <math>T^* = T_{pre_1, pre_2, \dots, pre_{L-1}\{n\} \max_{n=1,2,\dots,N} UCB\_score_{L,n}{}^{(t)}}</math>  <math>RecResult^{(t)} \leftarrow RandSelect(b_{m,T^*} = 1)</math>  End for  Return <math>RecResult^{(t)}</math></p>
--

Figure 10. Con-UCB Algorithm

In Fig.10,  $round$  represents order of rounds when recommendation are made at current layer,  $Paper$  represents the set of all the recommended papers,  $paper_k^{(t)}$  represents the  $k^{th}$  paper at round  $t$ .

## V. EXPERIMENTS

### A. Experiment Dataset and Design

In this paper, Web of Science journal articles in from 2009 to 2013 are used, and they are from the Science Citation Index Expanded (SCI- EXPANDED), Social Sciences Citation Index (SSCI) and Arts and Humanities Citation Index(AHCI).

In order to preliminarily test our developed system and compare two proposed MAB methods, 5 undergraduates are invited to participate in experiments and to determine their research topics. Three of them use recommender system based on  $\epsilon$ -Greedy\_SP while other two students use recommender system based on Con-UCB.

Number of topics at every layer is set to 5,  $Max\_layer$  is 2 and  $K$  is 5. We will compare the two methods in several indices and analyze the focusing process of user preferences through the change of their ratings in Section 5.2. Besides, the empirical analysis of two typical user shows the flexibility of our system to difference type of users.

### B. Experiment Result

**Overall Result.** First, average ratings of five invited students are shown in Figure 11, where user1, user2 and user3 employ  $\epsilon$ -Greedy\_SP while user4 and user5 employ Con-UCB.

It is easy to find that users' ratings increase gradually and the preferences are focused little by little. It shows that the proposed paper recommender systems can catch and focus user's preference gradually.

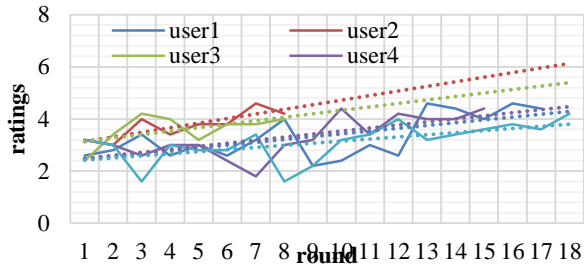


Figure 11. Average ratings of five students

More detailed result is shown in Table 3. It shows that  $\epsilon$ -Greedy\_SP performs better than Con-UCB in the case of insufficient samples. Due the lack of samples, the result can be also caused by the individual difference, so the result of the algorithm comparison here is just for reference and is not on a high confidence level.

TABLE III. COMPARISON OF  $\epsilon$ -GREEDY\_SP AND CON-UCB

Algorithm	Average round	Average rating	Average variance ratio
$\epsilon$ -Greedy_SP	11	3.491	6.250%
Con-UCB	16.5	3.2	5.455%

**Empirical Analysis.** In order to understand focusing processing of user's preference, we select user 3 as for empirical analysis. The recommendation process will promote to 2<sup>nd</sup> layer only if  $US$  conforms to tracking condition at least once, the preference at 2<sup>nd</sup> layer has no value at 1<sup>st</sup> round, and it could be discontinuous because of backtracking condition, so we choose the topics at 1<sup>st</sup> layer to analyze the focusing process. Preference Score of user3 to  $T_1 \sim T_5$  is shown in two diagrams from Figure 12 to Figure 13. It is visible that after 2<sup>nd</sup> round, user3 has a comparatively preference of  $T_5$ . The score of  $T_5$  is growing steadily and opens up a gap with other topics gradually until user3 finds the direction of research. This result shows that user3 has a roughly concept of his research topic, and our system here is a concrete refinement tool for user3.

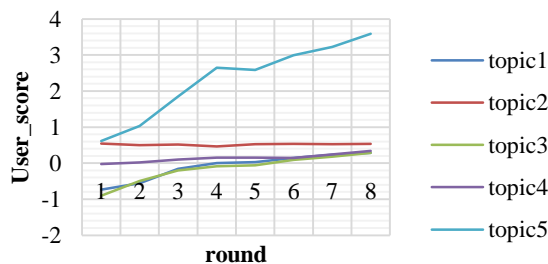


Figure 12. Line chart of user3's preference score

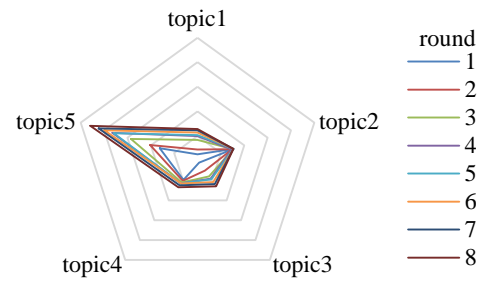


Figure 13. Radar chart of user3's preference score

## VI. CONCLUSION

Recommender system devote to finding individual items to users. Traditional recommender system does not work well when helping users to finish a task. In this paper, we propose a novel paper recommender system for finding research topic, where the user only needs to give feedbacks of recommended items. Two exploratory recommender methods based on MAB models are proposed. A prototype system is developed, and show good performance. In the future, the system will be tested by more invited students.

## REFERENCES

- [1] Tang T. Y., McCalla G.: Mining implicit ratings for focused collaborative filtering for paper recommendations. The Workshop on User and Group MODELS for Web-Based Adaptive Collaborative Environments, International Conference on User Modeling, 45-56 (2003)
- [2] Lee J., Lee K., Kim J. G.: Personalized Academic Research Paper Recommendation System. Computer Science (2013).
- [3] Beel J., Langer S.: A Comparison of Offline Evaluations, Online Evaluations, and User Studies in the Context of Research-Paper Recommender Systems. Research and Advanced Technology for Digital Libraries (2015).
- [4] Melnick S. L., Shahar E., Folsom A. R.: Sponsored vs. Organic (Research Paper) Recommendations and the Impact of Labeling. International Conference on Theory and Practice of Digital Libraries, 395-399 (2013).
- [5] Wang J.: Dynamic Attention Deep Model for Article Recommendation by Learning Human Editors' Demonstration. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2051-2059 (2017).
- [6] Tajima A.: Embedding-based News Recommendation for Millions of Users. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1933-1942 (2017).
- [7] Yang C., Bai L., Zhang C.: Bridging Collaborative Filtering and Semi-Supervised Learning: A Neural Approach for POI Recommendation. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1245-1254 (2017).
- [8] Barto A. G.: Reinforcement learning. Springer Berlin Heidelberg, 665-685 (1998).
- [9] Xu L., Jiang C., Qian Y.: Dynamic Privacy Pricing: A Multi-Armed Bandit Approach With Time-Variant Rewards. IEEE Transactions on Information Forensics & Security 12(2), 271-285 (2017).
- [10] Shahrampour S., Noshad M., Tarokh V.: On Sequential Elimination Algorithms for Best-Arm Identification in Multi-Armed Bandits. IEEE Transactions on Signal Processing, (2017).
- [11] Lacerda A.: Multi-Objective Ranked Bandits for Recommender Systems. Neurocomputing (2017), 246.

# Evaluating Multiple User Interactions for Ranking Personalization Using Ensemble Methods

Frederico Araújo Durão, Bruno Souza Cabral  
Department of Computer Science  
Federal University of Bahia  
Salvador, BA – Brazil  
fdurao@ufba.br, bruno.cabral@ufba.br

Renato D. Beltrão, Marcelo G. Manzato  
Mathematics and Computing Institute  
University of São Paulo  
São Carlos, SP – Brazil  
rdompieri@usp.br, mmanzato@icmc.usp.br

**Abstract**—The variety of interaction paradigms on the Web, such as clicking, commenting or rating are important sources that help recommender systems to gather accurate information about users’ preferences. Ensemble methods can be used to combine all these pieces of information in a post-processing step to generate recommendations that are more relevant. In this paper, we review the application of existing ensemble methods to improve ranking recommendations in the multimodal interactions context. We compared four ensemble strategies, ranging from simple to complex algorithms including Gradient Descent and Genetic Algorithm to find optimal weights. The evaluation using the HetRec 2011 MovieLens 2k dataset with three different types of interactions shows that a considerable 7% improvement in the Mean Average Precision can be achieved using ensembles when compared to the most performant single interaction.

**Index Terms**—recommender system, multimodal, user interaction

## I. INTRODUCTION

According to [1], from 2005 to 2020, the information in the digital universe will grow by a factor of 300, from 130 exabytes to 40,000 exabytes, or 40 trillion gigabytes (more than 5,200 gigabytes for every man, woman, and child in 2020). It is simply not possible to grasp even a small percentage of it in a single lifetime, there is too much information to process and to choose. The expression *Information Overload* was introduced to describe the sensation of fatigue and distress that follows the cognitive surplus required to handle the volume of information we have to deal with everyday [2].

Recommender Systems (RS) have emerged in response to the information overload problem in order to support users during content consumption decisions. They learn the users’ interests using their past interactions (ratings, votes, ranked lists, mouse clicks, page views, product purchases, etc.) and suggest products that are likely to be appreciable. In order to obtain users’ interests, three different forms can be used: explicit feedback; implicit feedback and hybrid approaches. Implicit feedback is the kind of information collected indirectly, such as mouse movements or clicks. In explicit feedback, the preferences are intentionally provided by the user, such as a “like” option or a rating. This type of information is considered more reliable, since the user is the one who exposes his interests, rather than being inferred. The problem is that

it requires an additional effort from the user to intentionally provide the feedback, who is not always willing to cooperate with the system [3]. Finally, the hybrid approach consists of applying the implicit and explicit feedback together to obtain more information about his preferences [4].

Despite the variety of ways to collect users’ preferences, actual recommender algorithms are modeled based on a single or a few types of interactions [5]. However, the accuracy can be improved if the system utilizes all available information. An approach for generically handling multimodal interactions is with ensemble methods. An ensemble method combines the predictions of different algorithms, or the same algorithm with different parameters to obtain a final prediction. Ensemble methods were the top performing solution in the Netflix Prize contest [6].

The most simple ensemble method is to compute the final prediction as the mean over all the predictions [3]. Better results can be obtained if the final prediction is given by a linear combination of the ensemble predictions. In this case, the combination weights have to be determined by some optimization procedure such as regularized linear and logistic regressions. However, not all available ensemble methods are practical for large-scale recommender systems because the massive amount of data demands vast amount of time and memory consumption.

In this paper, we analyze four ensemble strategies ranging from simple rank list merging to advanced strategies using Gradient Descent and Genetic Algorithm to find optimal weights to unify different types of feedback. We provide an experimental evaluation of those strategies with the HetRec2011 MovieLens 2k [7] dataset, simulating and inferring three classical users’ interactions: tagging, rating and browsing history.

This paper is structured as follows: Section II depicts the related work; Section III details the evaluated ensemble framework and strategies; Section IV presents the evaluation and validation of the approach with HetRec dataset with 800,000 ratings, along with an analysis of the performance of the four strategies; and finally Section V discusses the final remarks and future works.



## II. RELATED WORK

Recommender systems can be extended in several ways aiming at improving the understanding of users and items, incorporating new types of interaction in the recommendation process and making the combination of them. One of these improvements is the support for multi-criteria interactions, so as to provide greater flexibility and less obtrusive types of recommendations [8]. In this context, with more studies in the area of recommender systems, various algorithms enabled the usage of more than one type of user interaction.

These studies resulted in works such as Johansson [9], responsible for developing the MADFILM, a movie recommendation system that addresses the integration of prediction and organization of content, through explicit and implicit user's feedback. The work proposed by [10] developed a recommendation system for online video based on explicit and implicit feedback, plus feedback from relevant information provided by the user. The used video was composed of multimedia content and related information (such as query, title, tags, etc.). The project aimed to combine these types of interactions with the information provided by users in order to generate a more precise rank of relevant items. In order to automatically adjust the system, it was implemented a set of adjustment heuristics given new user interactions.

The SVD++ algorithm proposed by [11] uses explicit and implicit information from users to improve the prediction of ratings. As explicit information, the algorithm uses the ratings assigned by users to items, and as implicit information, it simulates the rental history by considering which items users rated, regardless of how they rated these items. However, it use a stochastic gradient descent to train the model, which requires the observed ratings from users. Thus, it is impossible to infer preferences for those users who provided only implicit feedback.

Ensemble is a machine learning approach that uses a combination of similar models in order to improve the results obtained by a single model, and can be used to combine multiple interactions. In fact, several recent studies, such as [12], demonstrate the effectiveness of an ensemble of several individual and simpler techniques, and show that ensemble-based methods outperform any single, more complex algorithm. Most of the related works in the literature point out that ensemble learning has been used in recommender system as a way to combine the prediction of multiple algorithms (heterogeneous ensemble) to create a more accurate rank [12], in a technique known as *blending*. Furthermore, they have been used with a single collaborative filtering algorithm (single-model or homogeneous ensemble), with methods such as *Bagging* and *Boosting* [3].

Cabral et al. [13] proposed three ensemble strategies that combine predictions from a recommender trained with distinct item metadata into a unified rank of recommended items. In comparison, da Costa et al. [14], proposed a similar ensemble strategy based on machine learning in order to combine different types of interactions generated by multiple recommenders.

Those strategies differ from the aforementioned works because they adopt a post-processing step to analyze the rankings created separately by different algorithms. The advantage of this approach is that it does not require the algorithm to be modified, or to be trained multiple times with the same dataset, and therefore, it is easier to extend the models to other types of interactions and recommenders. We implemented all strategies in a public available repository and evaluated three types of interactions (Ratings, Tags and Visualized Items) using the HetRec dataset [7].

## III. ENSEMBLE MODELS

In this section, we describe four ensemble strategies used in this work to combine multimodal interactions: **Most Pleasure**, the simplest ensemble strategy, combines predictions based on score; **Best of All strategy**, determines a preferred metadata for a user and uses it to create the ensemble; **Weighting strategy**, uses multiple metadata and weighs them with a Genetic Algorithm, optimizing for maximum Mean Average Precision (MAP); **BPR Learning Strategy** [15], which uses the Learn BPR to learn the optimal weights, optimizing for the Area under the ROC curve (AUC) .

### A. Most Pleasure Strategy

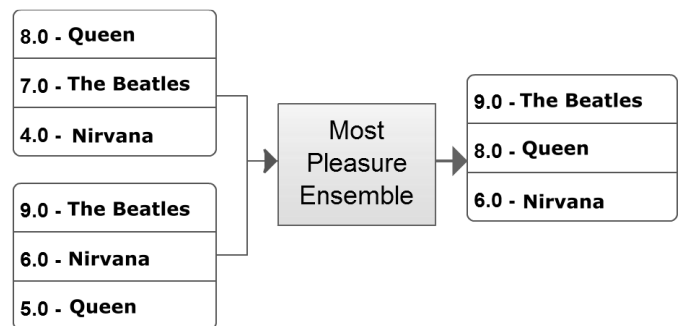


Fig. 1. Most Pleasure Strategy.

The *Most Pleasure* strategy is a classical aggregation method, often used for combining individual ratings for group rating [16]. It takes the maximum of individual ratings for a specific item and creates a unified rank. Figure 1 illustrates the *Most Pleasure* strategy, in which the output comprehends a ranked list of artists with highest ratings from two distinct input sets. It only needs the generated prediction set as an input, composed of the predictions from the recommender algorithm trained with one of the item's metadata. For each user, a new prediction is created, selecting the highest score of an item among all the individually-trained algorithms.

The idea behind this strategy is that differently trained algorithms have a distinct knowledge about the user's preferences, and the predicted score can be considered an indicator of the algorithm's confidence. Consequently, the created ensemble is a list of items in which the distinct algorithms have more confidence to recommend.

<https://github.com/wendelad/RecSys>

## B. Best of All Strategy

Differently from *Most Pleasure*, *Best of All* strategy assumes that different types of metadata can affect users differently. It considers the recommendation algorithm that provides the best results for a specific user (as illustrated by Figure 2).

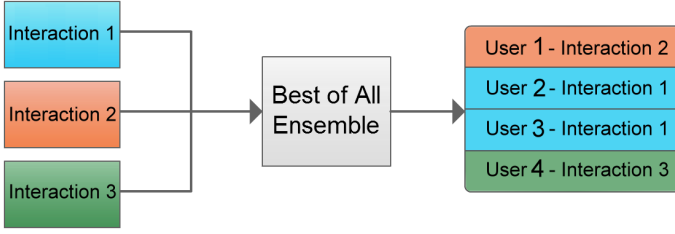


Fig. 2. Best of All Strategy.

The *Best of All* Strategy requires as input: i) the recommendation algorithm, ii) a training dataset, iii) a probe dataset, and iv) the set of item's metadata. Unlike the *Most Pleasure* strategy, this one requires a probe run to determine which is the best performing algorithm. Therefore, the dataset is divided in training and probe. The recommender algorithm is firstly trained using each of item metadata individually. Then, for each user, a probe run is made to determine the metadata with the highest performance (in terms of MAP). Finally, the recommender algorithms are retrained using all data (including the probe set), and the final ensemble is the result of the combination of predictions using, for each user, the prediction from the algorithm with the highest performance in the probe test.

The idea behind this strategy is that a single metadatum can greatly influence the user's preferences, and this should be used for future predictions. For instance, if a User A enjoys music from a particular genre such as "pop", and other User B enjoys music of some specific performer such as "Metallica", the ensemble will contain predictions from the recommendation algorithm trained with both: the genre metadatum for User A, i.e. "pop", and a performer metadata for user B, i.e. "Metallica".

## C. GA Weighting Strategy

One drawback of the *Best of All* strategy is that it considers that only one type of metadata influences the user preference. The *GA Weighting* strategy assumes that the interests of a user may be influenced by more than one metadatum, and with different levels. It considers all available metadata, assigning different weights for each prediction as shown in Figure 3.

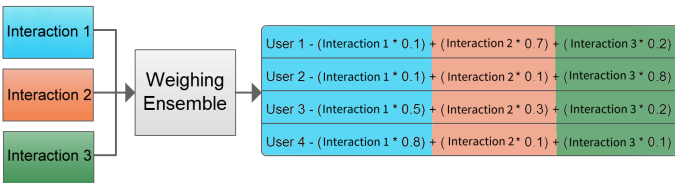


Fig. 3. Weighing Strategy.

Similarly to the previous strategy, it requires as an input the i) recommendation algorithm, ii) a training and probe

dataset, and iii) the set of item metadata. After training the algorithm using each of item metadata individually, a probe run is needed; however, the objective is to determine the optimal weights for each user. This is an optimization problem that was solved using Genetic Algorithm (GA).

The probe part consists of running the GA to find out the optimal weights. It was implemented using the GA Framework proposed by Newcombe [17], where the weights are the chromosomes, and the fitness function is the MAP score against the probe dataset. Other GA characteristics includes the use of 5% of Elitism, Double Point crossing-over, and Binary Mutations. Finally, the algorithms are retrained using all data (including the probe set), and the final ensemble uses, as the item score, the sum of individual predictions multiplied by the weights found in the probe phase and divided by the total number of metadata.

The idea behind it is that the different types of interactions influence differently the user preference. Still in the context of music, let us consider that a User A enjoys songs of a specific set of genres regardless of the performer and a User B that does not care about music genre or country of production. For the User A, the ensemble should give a higher weight for the music genre, and a lower weight for the production country. In contrast, to the User B, the ensemble should equally distribute the weights between those metadata.

## D. BPR Learning Strategy

In order to combine the output generated by each recommendation technique trained with a different kind of interaction, this ensemble strategy is based on a machine learning algorithm [14].

Firstly, it extracts information about users' interactions from the database, such as sets of tags, ratings and browsing history. With these interactions available, it runs the recommendation algorithms, which receive as input the users' interactions. In this step, each algorithm runs with a particular set of feedback, resulting in a feedback-specific personalized ranking (individual ranking) for each user. Thus, a feedback-specific ranking contains the items and their associated scores, which represent how much a user likes an item described by the considered set of attributes. The final step consists of combining all considered rankings into a final list of recommendations. To do that, it assigns weights according to the relevance of each type/set of attributes. This combination is performed according to a linear function, represented by  $\hat{r}_{u,i}^{final}$ :

$$\hat{r}_{u,i}^{final} = \beta_a r_{u,i}^a + \beta_b r_{u,i}^b + \dots + \beta_n r_{u,i}^n \quad (1)$$

where  $r_{u,i}^a, r_{u,i}^b, \dots, r_{u,i}^n$  indicate the scores computed previously by each individual recommendation algorithm for a  $(u, i)$  pair, and  $\beta_a, \beta_b, \dots, \beta_n$  are the weights of each individual score for the final prediction, learned using Learn BPR algorithm [18]. This is possible because of the natural strategy of BPR, which in a each interaction, select randomly a couple of items  $i$  and  $j$  for a user  $u$ , a known item  $i$  and one unknown item  $j$ .

Finally, the algorithm predicts scores for items not seen by each user and sorted these scores in descending order

resulting in the final ranking, which will be recommended in a top  $N$  ranking list. The underlying characteristic of this algorithm is the ability to learn the users' preferences to employ this information to match the recommendations generated individually for each type of interaction.

#### IV. EVALUATION

The evaluation consists in comparing the four ensemble strategies as presented in Section 3, using a standard dataset available in the literature. Three different interactions, history(watched movies), tags and ratings were trained individually and combined using the ensemble techniques. The combined results and individual interactions were evaluated to check the contribution of each aspect to the final recommendation improvement.

##### A. Dataset

In order to evaluate the performance of the ensemble strategies, we used the HetRec MovieLens  $2k$  dataset [7]. MovieLens  $2k$  consists of 800,000 ratings, 10,000 interactions tags applied to 2,113 users and 10,197 movies. As explicit information, we used the ratings that users assigned to items, and as implicit interaction, we considered: i) whether a user tagged an item or not; and ii) the history of visited items, which is simulated by boolean values (visited or not) generated by the ratings and tagging activities.

In this paper, we adopted a classical methodology used by the research community with regard to recommender systems evaluation [8]. We divide the full dataset into two sets, 80% for training and 20% for testing. The training set is used to run the isolated algorithms and predict weights for each pair of algorithms (simulate the real-time interaction from the user); and test set is used with the *All but One protocol* to evaluate the approaches.

##### B. Experimental Setup and Evaluation Metrics

In this evaluation we use the All But One [19] protocol for the construction of the ground truth and the 10-fold-cross-validation. We randomly divided the dataset into 10 disjoint subsets of equal size and for each sample we use  $n - 1$  of data for training and the rest for testing. The training set  $t_r$  was used to train the proposed ensemble and test system  $T_e$  randomly split an item for each user to create the truth set  $H$ . The remaining items form the set of observable  $O$  is used to test the unimodal algorithms. We also evaluated using the standard protocol, where all items are considered. To assess the outcomes of the systems we use evaluation metrics Precision and Mean Average Precision (MAP) [20]. Then, we compute Precision and Mean Average Precision as follows:

**Precision** calculates the percentage of recommended items that are relevant. This metric is calculated by comparing, for each user in the test set  $T_e$ , the set of recommendations  $R$  that the system makes, given the set of observables  $O$ , against the set  $H$ :

$$Precision(T_e) = \frac{1}{|T_e|} \sum_{j=1}^{|T_e|} \frac{|R_j \cap H_j|}{|R_j|} \quad (2)$$

**Mean Average Precision** computes the precision considering the respective position in the ordered list of recommended items. With this metric, we obtain a single value accuracy score for a set of test users  $T_e$ :

$$MAP(T_e) = \frac{1}{|T_e|} \sum_{j=1}^{|T_e|} AveP(R_j, H_j) \quad (3)$$

where the average precision (AveP) is given by

$$AveP(R_j, H_j) = \frac{1}{|H_j|} \sum_{r=1}^{|H_j|} [Prec(R_j, r) \times \delta(R_j(r), H_j)] \quad (4)$$

where  $Prec(R_j, r)$  is the precision for all recommended items up to ranking  $r$  and  $\delta(R_j(r), H_j) = 1$ , iff the predicted item at ranking  $r$  is a relevant item ( $R_j(r) \in H_j$ ) or zero otherwise.

The *GA Weighting Strategy*, which utilizes a Genetic Algorithm (GA), uses a population of size 40 with 90 generations, a crossover probability of 80% and a mutation probability of 8%. This is a small number of generations, and usually a much higher number of generations is used for convergence; however, due to the size of our dataset, we traded precision for speed.

In this work we used Precision@ $N$  and MAP@ $N$ , where  $N$  took values of 1, 3, 5 and 10 in the rankings returned by the system. For each configuration and measure, the 10-fold values are summarized by using mean and standard deviation. In order to compare the results in statistical form, we apply the two-sided paired t-test with a 95% confidence level [21].

##### C. Methodology

For implicit data interactions (history and tags), we used the BPR-MF implementation available in the MyMediaLite library [22]. It is an implementation of the Bayesian Personalized Ranking (BPR) [23], a generic framework for optimizing different kinds of models based on training data containing only implicit feedback information. For explicit interactions (ratings), we used SVD++ [4], also from MyMediaLite library. All four ensemble strategies were implemented using MyMediaLite library and are publicly available.

All the runtime evaluations were executed in the same machine, a Core i7-2670QM with 8GB of RAM, with the .NET 4.5 framework with all available patches applied. The result is the average of 10 runs.

##### D. Results

Table 1 and Table 2 show the results of this evaluation for single interactions and ensembles. We compared our results to *tags*, the best performing interaction. As seen, the *BPR Learning* strategy achieved statistically better results than the

<https://github.com/wendelad/RecSys>

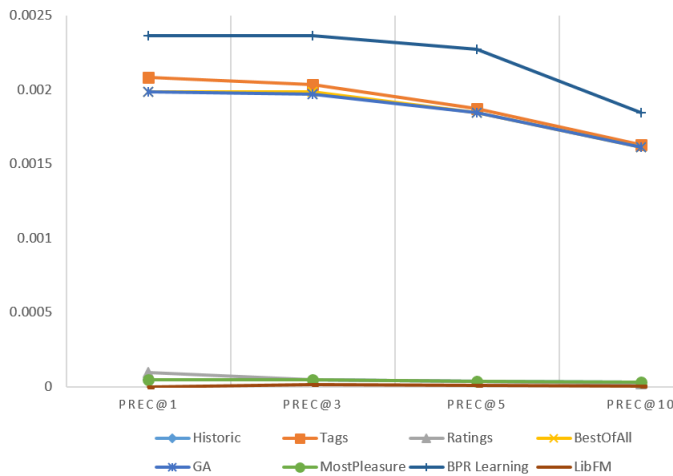


Fig. 4. Precision@1, 3, 5, 10 using All But One Protocol. baseline, as proven by the t-student analysis (with  $p < 0.05$ ) in Table 3. Figure 4 illustrates the algorithms' precision@1, 3, 5 and 10 using the All But One Protocol.

TABLE I  
ALGORITHMS' PERFORMANCE IN PRECISION@1, 3, 5 AND 10.

	Prec@1	Prec@3	Prec@5	Prec@10
History	0.000047	0.000047	0.000037	0.000033
Tags	0.002082	0.002035	0.001874	0.001628
Ratings	0.000094	0.000047	0.000037	0.000018
BestOfAll	0.001988	0.001988	0.001845	0.001614
GA	0.001988	0.001971	0.001845	0.001614
MostPleasure	0.000047	0.000047	0.000037	0.000033
BPR Learning	<b>0.002366</b>	<b>0.002366</b>	<b>0.002271</b>	<b>0.001845</b>
<b>Ensemble improvement</b>	12.0 %	12.1 %	21.1 %	13.3 %

TABLE II  
ALGORITHMS' PERFORMANCE IN MAP@1,10 AND MAP UNDER STANDARD PROTOCOL.

	MAP@5	MAP@10	MAP(Std)
History	0.000104	0.000120	0.000226
Tags	0.004569	0.005456	0.004729
Ratings	0.000119	0.000119	0.000047
BestOfAll	0.004458	0.005345	0.004956
GA	0.004441	0.005334	0.004999
MostPleasure	0.000104	0.000120	0.000239
BPR Learning	<b>0.005229</b>	<b>0.006044</b>	<b>0.005075</b>
<b>Ensemble improvement</b>	14.4 %	10.7 %	7.2 %

The results from Table 1 and Table 2 indicate that in some cases, ensembles got significantly higher scores than single interactions. The improvement level was between 7.2% and 21.1% compared to the best performing interaction. These improvements were significant as increasing the MAP and precision is a challenge, and every increment in MAP is hard to achieve. Surprisingly, the *tags* interaction achieved higher scores compared to other single interactions. This is an interesting result, because *tags* contain a more diverse set of information, which probably simulate an ensemble. The *BPR Learning* strategy was optimal for all given scenarios since it uses all interactions to make predictions, and it assigns different weights to the most relevant metadata according to the taste of each individual user. On the other hand, the *MostPleasure* strategy achieved the lowest performance among the ensemble strategies.

TABLE III  
T-TEST COMPARING MAP@5 USING BPR LEARNING WITH TAGS.

	BPR Learning	Tags
Mean	0.005115	0.004569
Variance	3.16E-07	1.07E-07
Observations	10	10
df	14	
t Stat	-2.65099	
P(T<=t) one-tail	0.009496	
t Critical one-tail	1.76131	
P(T<=t) two-tail	0.018992	
t Critical two-tail	2.144787	

The *GA Weighting* and *Best of All* strategies obtained a good performance, close to the best performing interaction, except in MAP with the standard protocol, where it archived a better result. The *Best of All* strategy is simple to implement and does not require weight optimization, an expensive step in the process required for *BPR Learning* and *GA Weighting*. Alternatively, *GA Weighting* does requires a weight optimization step, but as it uses a Genetic Algorithm, one can manually set the parameters and tradeoff speed or performance.

TABLE IV  
COMPARISON OF ENSEMBLE ALGORITHMS RUNTIME IN MINUTES AND IMPROVEMENT (IN MAP) OVER THE BEST PERFORMING INTERACTION (TAGS).

Ensemble	Probe Run	Time(min)	Improvement
MostPleasure	No	↓ 0.1	-94%
BestOfAll	Yes	↓ 0.1	4.7%
GA	Yes	43.2	5.6%
BPR Learning	Yes	52.1	7.2%

Table IV lists the ensemble strategies runtime and the need for a probe run. *Most Pleasure* has the advantage of not requiring a probe run, but in our evaluation achieved the worst result of all compared strategies with a 94 % lower performance. *BPR Learning* and *GA Weighting* achieved 5.6 % and 7.2 % MAP improvement respectively with a slight runtime advantage for *GA Weighting*. *Best of All* achieved a good performance improvement compared to the best performing single interaction with the advantage of being fast to compute.

The overall absolute scores obtained and described in this paper are small because of the Sparsity and evaluation protocol used in the experiments. The All But One protocol hides one item from each user in the test set and considers it as the ground truth. As we are recommending top N items, the precision and MAP will decrease because the system considers there are N relevant items, although the protocol has set only the hidden item as relevant. The high sparsity stands as another challenge, as many movies were not rated, only tagged. In this case, the rating prediction cannot be made. Another issue is that the rating rank is build using the rating predictions in a decreasing order from the SVD++ algorithm and the dataset can contain items with a low score, lowering the metrics related to this interaction as the test dataset is generated randomly. In this way, it is important to rely only on the differences among the approaches, and we managed to increase the results of our proposal when compared to the baselines.

Finally, we conclude that ensemble algorithms significantly improved the recommender prediction performance, with the

*BPR Learning* strategy standing out with higher performance improvement on most of the scenarios followed by *GA Weighting* strategy with a lower performance but with a slight smaller runtime and the *Best of All* strategy, whose the highlight is being almost instantaneous to compute.

## V. CONCLUSION

In this paper we evaluated four ensemble strategies to unify different types of feedback from users when consuming content in order to provide better recommendations. The advantage is that more information about the interests of the user can be obtained when analyzing multimodal interactions. All strategies evaluated do not require modification of the recommender algorithm, namely *Most Pleasure*, *Best of All*, *Genetic Algorithm Weighting* and *BPR Learning*. The considered recommender algorithms did not take advantage of multiple types of interactions and the evaluated ensemble algorithms were able to enable those recommenders to take advantage of all interactions. *Most Pleasure*, the simplest strategy, consisted of combining the predictions based on score. *Best of All* determined a single metadata that was more preferred for a user, and the *Weighting* strategy uses multiple interactions and weights them with a Genetic Algorithm that optimizes the MAP and finally, *BPR Learning* uses LearnBPR to optimize the weights related to AUC. Results from the experiments show the effectiveness of combining various types of interactions in a single model for recommendation using ensemble learning. Our evaluation showed a considerable MAP improvement between 10.7% and 21.1% when using the ensemble algorithms, with the *BPR Learning* producing the best recommendation for the majority of scenarios. These encouraging results indicate that ensemble algorithms can be used to enhance the recommender algorithms with multiple interactions.

As future work, we plan to implement more complex ensemble strategies and evaluate the algorithms with a higher number of metadata in order to verify whether multimodal information can generate better recommendations. In order to do so, it will be necessary to find a more extensive dataset and to evaluate the algorithms runtime performance with this increased work.

## ACKNOWLEDGMENT

The authors would like to thank the financial support from FAPESP (proc. number 2016/2028-6).

## REFERENCES

- [1] I. Management Association, *Big Data: Concepts, Methodologies, Tools, and Applications: Concepts, Methodologies, Tools, and Applications*. IGI Global, 2016. [Online]. Available: <https://books.google.com.br/books?id=BKEoDAAAQBAJ>
- [2] C. Musto, "Enhanced vector space models for content-based recommender systems," in *Proceedings of the Fourth ACM Conference on Recommender Systems*, ser. RecSys '10. New York, NY, USA: ACM, 2010, pp. 361–364. [Online]. Available: <http://doi.acm.org/10.1145/1864708.1864791>
- [3] A. Bar, L. Rokach, G. Shani, B. Shapira, and A. Schlar, "Improving simple collaborative filtering models using ensemble methods," in *Multiple Classifier Systems*. Springer, 2013, pp. 1–12.
- [4] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '08. New York, NY, USA: ACM, 2008, pp. 426–434. [Online]. Available: <http://doi.acm.org/10.1145/1401890.1401944>
- [5] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 1, no. 1, pp. 67–82, Apr 1997.
- [6] A. Töscher, M. Jahrer, and R. M. Bell, "The bigchaos solution to the netflix grand prize," *Netflix prize documentation*, 2009.
- [7] I. Cantador, P. Brusilovsky, and T. Kuflik, "2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011)," in *Proceedings of the 5th ACM conference on Recommender systems*, ser. RecSys 2011. New York, NY, USA: ACM, 2011.
- [8] F. Ricci, L. Rokach, and B. Shapira, "Introduction to recommender systems handbook," in *Recommender Systems Handbook*. Springer, 2011, pp. 1–35.
- [9] P. Johansson, "Madfilm - a multimodal approach to handle search and organization in a movie recommendation system," *Proceedings of the 1st Nordic Symposium on Multimodal Communication*, pp. 53–65, 2003.
- [10] B. Yang, T. Mei, X.-S. Hua, L. Yang, S.-Q. Yang, and M. Li, "Online video recommendation based on multimodal fusion and relevance feedback," in *Proceedings of the 6th ACM International Conference on Image and Video Retrieval*, ser. CIVR '07. New York, NY, USA: ACM, 2007, pp. 73–80.
- [11] Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," *IEEE Computer*, vol. 42, no. 8, pp. 30–37, 2009.
- [12] M. Jahrer, A. Töscher, and R. Legenstein, "Combining predictions for accurate recommender systems," ser. KDD '10. New York, NY, USA: ACM, 2010, pp. 693–702. [Online]. Available: <http://doi.acm.org/10.1145/1835804.1835893>
- [13] B. Souza Cabral, R. Dompieri Beltrao, M. Garcia Manzano, and F. Araújo Durão, "Combining multiple metadata types in movies recommendation using ensemble algorithms," in *Proceedings of the 20th Brazilian Symposium on Multimedia and the Web*, ser. WebMedia '14. New York, NY, USA: ACM, 2014, pp. 231–238. [Online]. Available: <http://doi.acm.org/10.1145/2664551.2664569>
- [14] A. da Costa Fortes and M. G. Manzano, "Ensemble learning in recommender systems: Combining multiple user interactions for ranking personalization," in *Proceedings of the 20th Brazilian Symposium on Multimedia and the Web*, ser. WebMedia '14. New York, NY, USA: ACM, 2014, pp. 47–54.
- [15] A. F. d. Costa and M. G. Manzano, "Multimodal interactions in recommender systems: An ensembling approach," in *2014 Brazilian Conference on Intelligent Systems*, Oct 2014, pp. 67–72.
- [16] J. Masthoff, "Group recommender systems: Combining individual models," in *Recommender Systems Handbook*. Springer, 2011, pp. 677–702.
- [17] J. Newcombe, "Intelligent radio: An evolutionary approach to general coverage radio receiver control," Master's thesis, DeMontfort University, UK, 2013.
- [18] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, ser. UAI '09. Arlington, Virginia, United States: AUAI Press, 2009, pp. 452–461.
- [19] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," ser. UAI '98, San Francisco, CA, USA, 1998, pp. 43–52. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2074094.2074100>
- [20] E. M. Voorhees and D. K. Harman, *TREC: Experiment and Evaluation in Information Retrieval (Digital Libraries and Electronic Publishing)*. The MIT Press, 2005.
- [21] T. M. Mitchell, *Machine Learning*, 1st ed. New York, NY, USA: McGraw-Hill, Inc., 1997.
- [22] Z. Gantner, S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "MyMediaLite: A free recommender system library," in *Proceedings of the 5th ACM Conference on Recommender Systems*, ser. RecSys '11, New York, NY, USA, 2011, pp. 305–308.
- [23] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thieme, "Learning attribute-to-feature mappings for cold-start recommendations," in *2010 IEEE 10th International Conference on Data Mining (ICDM)*, dec. 2010, pp. 176–185.

# Weighted Data Set Reduction for Bug Triaging

Miaomiao Wei, Shikai Guo and Rong Chen

Dalian Maritime University, Dalian, China  
 {weimiaomiao, shikai.guo, rchen}@dlmu.edu.cn

**Abstract**—Despite the great potential to save the labor cost of developers, automated bug triaging as a text classification problem has not been thoroughly investigated on long descriptions, which are informative but often noisy. In this paper an effective bug triage technique is proposed to build a high quality set of bug data by removing the noisy and non-informative bug reports while assigning new bugs to an appropriate developer. The proposed technique – weighted data set reduction – is built upon three feature selection algorithms and four instances selection algorithms with intention to recommend the bug and to automatically assign it more accurately even with noisy bug descriptions. Several experiments are conducted and the experimental results show that the reduced training sets by the proposed approach can achieve better accuracy in several cases, about 2-3% on average better than the original ones.

**Keywords**—Bug Triaging; Bug Reports; Machine Learning;

## I. INTRODUCTION

With the huge information about bugs reported [1] by today’s bug tracking systems (e.g. , Buzilla, JIRA, mantis ), there is an increasing need to introduce some form of automation within the bug triaging process, so that no time is wasted on the assignment of new issues to appropriate developers. In this paper, we propose a weighted data reduction technique that combines feature selection and instance selection algorithms to yield a small size and high-quality training set that can enhance mainstream classifiers.

In doing so, three feature selection algorithms (CHI, Information Gain (IG), and One Rule (OneR)) can distinguish important attributes with meaningless ones, while four instance selection algorithms ICF, Condensed Nearest Neighbor (CNN), Minimal Consistent Set (MCS) and Edited Nearest Neighbor (ENN) can choose more representative examples.

## II. MODEL

Figure 1 shows our bug triaging model based on data set reduction and text classification. The present approach is conducted in four steps: (1) Part of bug reports with the label are selected from the original data. (2) Data preparation. The selected bug reports are transformed into standard data set. (3) Data set reduction. The data set is reduced with feature selection and instances selection algorithms. (4) Classification. When a new bug report is submitted, a size of K recommendation list is generated by a concrete classifier.

Before data set reduction, the data are prepared as follows: first, select the bug not empty. Second, select the fixed bugs and duplicate bugs. Third, label the bugs and delete the console information. Fourth, remove the inactive developers. Fifth, separate words and remove stop words. At last, generate a feature matrix as the standard data.

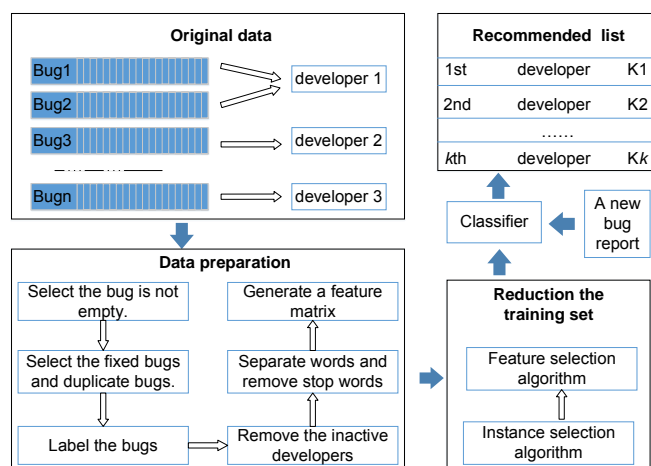


Figure 1. The text categorization approach for bug triaging.

Algorithm 1. Weighted Reduction training set algorithm.

Input: training set X,  
 Output: X\*

1. for each bug report in training set
2.   for each word in short description
3.     word frequency\*  $\eta$
4.   end for
5. end for
6. generate a weighted feature matrix
7. while not approach the scale of reduction
8.   apply FS  $\rightarrow$  IS or IS  $\rightarrow$  FS to weighted feature matrix
9. return X\*

## III. EXPERIMENT AND CONCLUSION

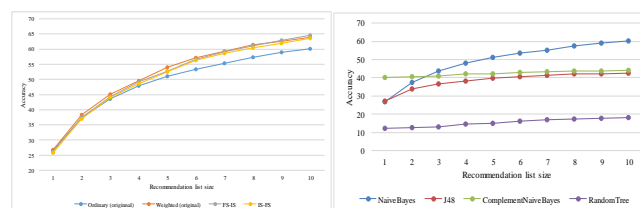


Figure 2 the results of dataset reduction.

Figure 2 shows the results of our approach. NB classifier has the best triaging effect in our study compared with the other three bug classification algorithms. Therefore, FS-IS is a good choice for the training set reduction.

## REFERENCES

- [1] Zhang J, Wang X Y, Hao D, et al, “A survey on bug-report analysis,” Science China Information Sciences, 2015, 58(2):1-24.

# Integrating Challenge Based Learning Into a Smart Learning Environment: Findings From a Mobile Application Development Course

Rafael Chanin, Alan R. Santos, Nicolas Nascimento, Afonso Sales, Leandro Pompermaier and Rafael Prikladnicki  
Pontifical Catholic University of Rio Grande do Sul, School of Technology  
Av. Ipiranga 6681, Porto Alegre, Brazil, CEP: 90619-900  
{rafael.chanin, alan.santos, nicolas.nascimento, afonso.sales, leandro.pompermaier, rafaelp}@pucrs.br

**Abstract**—Training students on mobile application development inherits the challenges of teaching software engineering where the target computer is a device that has a large number of features accessible by software. Furthermore, the most related experience in teaching students reveals difficulties in developing software engineering competencies. In this paper we present results from an iOS course held in a smart learning environment that adopted the Challenge Based Learning framework as the teaching methodology. Our results indicate that combining an active methodology along with an engaging environment can foster and improve software development learning.

**Keywords**—Smart Learning Environments, Challenge Based Learning, Mobile Software Development, Teaching.

## I. INTRODUCTION

Active learning methodologies like Problem-Based Learning, Project-Based Learning (PBL) and Challenge Based Learning (CBL) engage students and improve their performance. These approaches moves students to a different role when compared to traditional classes, engaging students in real problems [6]. In this study, we focus on the use of CBL in a smart learning environment. CBL enables students to conduct research by integrating practices with theory and application of knowledge and skills, such as collaboration and problem solving, and taking action in the community context [5].

Moreover, the educational context is changing due to the advancements in technology [4]. Traditional classes, often perceive as formal, passive, and not engaging, have been slowly replaced by student-centric approaches. This new way of perceiving education, combining active methodologies, technology and modern infrastructure has led to the creation of the term “Smart Learning Environments” [3].

In this sense, this paper presents findings from a 6-weeks mobile application development course that uses CBL as a teaching methodology. Our preliminary results indicate that applying a pro-active pedagogy framework that fosters collaboration and personalized learning within a smart learning environment can be beneficial to students.

The remainder of this paper is organized as follows. Section II presents the background on smart learning en-

vironments. In Section III the concept of Challenge Based Learning is explained and, in Section IV, we describe the mobile application course in detail. Section V depicts the methodology used in this research. In Section VI we present our preliminary results and, finally, we draw our conclusion and future works in Section VII.

## II. SMART LEARNING ENVIRONMENTS

There are several definitions for *Smart Learning Environments* (SLE) in the literature. However, in the context of this paper, we use the one from Koper [3], which is “*physical environments that are enriched with digital, context-aware and adaptive devices, to promote better and faster learning*”. Even though the word “smart” relates to the use of smart technologies, the main goal of a SLE is to provide learning guidance and all the necessary infrastructure to make the learning process effective, efficient and engaging [2].

One of the main goals of a SLE is to meet the educational needs from current students. Traditional classes, in which instructors are in the center of the process and students do not actively participate, have been replaced by approaches that foster collaboration, action and engagement. However, in order to effectively implement this change, not only the physical environment needs to be adapted, but also instructors need to put in place a different teaching methodology [4].

In this new context, instructors are no longer the only source of knowledge. Similarly, students are not only knowledge consumers. In fact, the roles of instructors and students may become less distinct and could be interchangeable [4].

## III. CHALLENGE-BASED LEARNING

Experiential learning is the source of a variety of learning frameworks that are used all over the world. Problem-Based Learning, Project-Based Learning, Task-Based Learning and Challenge Based Learning are just a few examples of these frameworks. “*The foundations of experiential learning can be found within the history of most cultures, but were formally organized and presented by David Kolb drawing heavily on the works of John Dewey and Jean Piaget*” [6]. Challenge Based Learning (CBL) [5] is a learning framework based on solving real world challenges.

The CBL process begins with the definition of a *big idea*, which is a broad concept that can be explored in several ways. The big idea has to be engaging and important to students and society. Once the big idea is chosen and the *essential question* is created, the *challenge* is defined. From this point, students must come up with the *guiding questions* and *guiding activities and resources*, which will guide them to develop a successful solution. The next step is *analysis*, which will set the foundation for the definition of the *solution*. Once the solution is agreed upon, the *implementation* begins. Finally, *evaluation* is undertaken in order to check out the whole process and verify if the solution can be refined.

#### IV. THE COURSE

The course curriculum applied in this work can be divided into two portions: *iOS Programming* and *User Experience*. All participants received CBL training and were required to dedicate 20 hours per week during six weeks. The learning was facilitated by instructors, which had different levels of industry and academic experience. All instructors had previous iOS development training and CBL training, and more than four years experience. In addition, the course was held in a smart learning environment, which provided not only all necessary equipment, but also a modern infrastructure that allow students to be creative and comfortable during the learning process.

TABLE I. CLASSES ACTIVITIES

Class	Activities	Deliverable
1	Equipment assignment to students. Student's and Instructors presentations. Quick introduction to important tools and shortcuts.	Reflection
2	Introduction to Coding, Introduction to Storyboards, UILabel and UIButton.	Exercise
3	UIView, UIViewController and the Model-View-Controller (MVC) paradigm.	Exercise
4	Introduction to UX (User-Experience) in iOS, Personas and Paper Prototyping.	Exercise
5	UIImageView, UIPickerView, UIDatePicker.	Exercise
6	Navigation using multiple ViewControllers, UINavigationController and UITabBarController.	Exercise
7	UISlider and UIScrollView.	Exercise
8	Challenge Based Learning.	CBL Process Documentation
9	AutoLayout.	Exercise
10	UITableView and Nano-Challenge.	None
11	Nano-Challenge Development and Deliver.	Nano-Challenge Solution
12	UserDefaults and CoreData.	Exercise
13	Design Guidelines.	Exercise
14	MapKit.	Exercise
15	Mini-Challenge - Engage.	None
16-17	Mini-Challenge - Investigate.	None
18-29	Mini-Challenge - Act.	None
30	Mini-Challenge Presentations.	Mini-Challenge Solution Keynote, Reflection

The deliverables described in Table I are related to items used for students development and assessment as described by Nichols *et al.* [5]:

- **Reflection:** It is a video, audio or text file where students reflect on the content and on the process. As described in [5]: "*Much of the deepest learning takes place by considering the process, thinking about one's*

*learning, and analyzing ongoing relationships between the content and concepts.*".

- **Exercise:** Students are encourage to follow a development rule (such as using a specific framework) but are free to increment the solution and to develop it.
- **CBL Documentation:** During the course, learners produce contents using text, video, audio and pictures. These artifacts are helpful, as they expose information of the learning process. These can serve many uses, such as reflections, assessments, evidence of learning, portfolios and for telling the story of their challenges.
- **Nano-Challenge:** One type of CBL activity. These are shorter in length, focused on a particular content area or skill, have tight boundaries and are guided by the instructor. Both *Big Idea* and *Essential Question* are provided to the students. The process includes some level of investigation, but at a lower level of intensity and often stop short of implementation with an external audience.
- **Mini-Challenge:** Another type of CBL activity. It has a longer duration (2-4 weeks) and allows learners to start with a *Big Idea* and work using the entire framework. The research depth and the reach of their solutions increases and the focus can be content specific or multidisciplinary. Mini-Challenges are good for intense learning experiences that stretch learners and prepare them for longer challenges.

#### V. METHODOLOGY

The research methodology for this study was based on a process proposed by Eisenhardt [1]. The proposed research question was: "*Can mobile application development be more effectively learn if taught in a smart learning environment using an active methodology, such as Challenge Based Learning?*". The rationale behind this question was to find out whether students can take advantage of the environment as well as of the methodology in order to better learn the content.

##### A. Data Collection & Analysis

Throughout the course, we collected several data regarding students deliverables. In addition, we conducted a survey with all students that have completed the course described in Section IV. In total, 25 students were interviewed. All questions were open-ended and focused on the following areas: (i) teaching environment; (ii) teaching methodology; (iii) content; and (iv) instructors.

Once all data was collected, we grouped and categorized the information. It is important to point out that part of our evaluation was based on data collected from students. In order to mitigate eventual flaws during data collection, the interviews were conducted by two people that did not directly participate in the course.

#### VI. RESULTS

We found indicatives that the combination of an innovative environment with an active learning methodology was key to the success of the course. Students felt engaged and motivated to learn and even to go beyond the course content.



Regarding the environment, 90% of the students emphasized how impressed they were when they first enter the classroom. The infrastructure, a coworking space equipped with comfortable and adjustable chairs and tables and several spaces to brainstorm and to sketch ideas, not only provided everything students needed, but it was also inspiring.

One key point the final survey allowed us to perceive was regarding the teaching methodology. Some of the students reported feeling uncomfortable with the dynamics used in class at first, as the lecturing (where the instructor would present the content) was not long and the students were quickly given a challenge. However, as the course progressed, they understood that this approach was beneficial, as they reported being more engaged with the content.

None of the students knew Challenge Based Learning prior to the course. Nonetheless, all of them pointed out that the methodology was key to keep them focused and engaged throughout the course. Several students, however, mentioned that the methodology could have been introduced in the first week of the course. As can be noticed in Table I, CBL was only presented to students on the 8th class.

The Mini-Challenge delivered and presented at the end of the course proved (according to the evaluation of the instructors) that students learned the content. Even though some students mentioned that having one or two more weeks would be beneficial to the learning process, they were all able to create a complete mobile application, with some having developed features that were not even covered during classes, such as Speech Recognition.

Regarding the instructors, 50% of the students pointed out that the different teaching styles was a problem. For example, while one instructor explained a specific content in detail, the other chose to give just a quick overview and then let students search for other information. In total, four instructors participated in the course. It became clear that it was not necessary to have that many instructors in this course.

Another set of data collected was students' deliverables. A *Delivered* task (85%) is one that is completed in terms of scope, quality and deadline. A *Partially Delivered* task (2%) is one that fail in scope, quality or deadline. Finally, a *Not Delivered* task (13%) is one that was not delivered. The evaluation for each task was made by at least two instructors.

This information was somehow intriguing to the authors, since they understood that having 13% of the tasks not delivered was too much. By analysing task by task (see Fig. 1) it can be seen that tasks 10 and 11 are the outliers. By talking to the instructors, we found out that since students were performing really well, they decided to raise the bar after task 9. It turned out that almost half of the students were not able even to partially deliver these challenges.

A lesson learned about this approach is that to push students in order to see where they can get can be good. However, this needs to be strategically done in order not to disengage students. For instance, a given challenge can have three different achievement levels (for example, bronze, silver and gold). By doing so, not only instructors can measure how far students can go, but it also keeps students motivated into achieving the highest level.

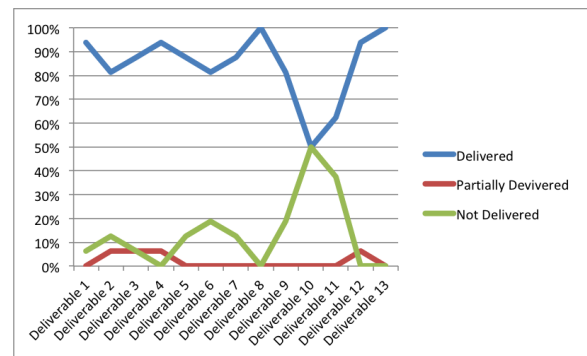


Fig. 1. Deliverables During the Course

## VII. CONCLUSION AND FUTURE WORK

The traditional educational landscape is changing to a more pro-active and collaborative one. In this scenario, instructors also need to adapt themselves in order to better help students throughout the learning process.

This work presented a case study that represents this new trend. We provided findings from a mobile application development course that took advantage of a modern environment, new technologies, and a pro-active teaching methodology - Challenge Based Learning. Our results, although preliminary, reveal that students in fact learn and engage more when they are put in the center of the learning process. Moreover, the use of challenges kept students motivated to find solutions, which makes the learning process more fun and less tedious.

As future work we intend to monitor and evaluate more courses in similar contexts that the one presented in this paper. In addition, we are planning to extend the duration of the course from six to eight weeks in order to give more time for the students to practice the concepts learned as well as more time in the Mini-Challenge activity to develop a more robust mobile application.

## ACKNOWLEDGMENTS

This project is partially funded by FAPERGS, project 17/2551-0001/205-4.

## REFERENCES

- [1] K. Eisenhardt. Building theories from case study research. *Academy of management review*, 14(4):532–550, 1989.
- [2] B. Gros. The design of smart educational environments. *Smart Learning Environments*, 3(1):1–11, 2016.
- [3] R. Koper. Conditions for effective smart learning environments. *Smart Learning Environments*, 1(1):1–17, 2014.
- [4] J. Ng, D. Ruta, A. Al Rubaie, D. Wang, L. Powell, B. Hirsch, L. Ming, C. Ling, and A. Al Dhanhani. Smart learning for the next generation education environment. In *2014 International Conference on Intelligent Environments*, pages 333–340. IEEE, 2014.
- [5] M. Nichols, K. Cator, and M. Torres. *Challenge Based Learning Guide*. Digital Promise, Redwood City, CA, USA, 2016.
- [6] A.R. Santos, A. Sales, P. Fernandes, and M. Nichols. Combining Challenge-Based Learning and Scrum Framework for Mobile Application Development. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE'15)*, pages 189–194, Vilnius, Lithuania, July 2015.

# IM Search: An Agent-based Personalized Metasearch Engine

Meijia Wang, Qingshan Li, Yishuai Lin

Software Engineering Institute, Xidian University, Xi'an, China  
shuilingyi@126.com; qshli@mail.xidian.edu.cn; yslin@mail.xidian.edu.cn

*Abstract—Metasearch engine integrates search results from multiple underlying search engines, improving recall ratio in the big data environment. Multi-agent system is an important way to implement metasearch engine. Great progress has been made in this area, however the previous studies are still short of personalization level. To improve the precision ratio, this paper proposes a personalized metasearch engine which of Agent-based architecture. According to click-through data, the metasearch engine has the ability to schedule the appropriate search engines based on the expertness model, merge all of results into a single list by taking user interest into account, and provide personalized recommendation. Experimental results show that the proposed personalized metasearch engine performs better on precision. It is feasible to provide the required search results more effectively.*

*Keywords: metasearch engine; multi-agent system; personalized search*

## I. INTRODUCTION

With the widespread use of Internet, information manifests an explosive growth. A search engine is a tool that helps users to find useful information on the Internet. However, the number of web documents is daily increasing, consisting more than 10 billion web documents distributed on millions of servers [1]. An individual search engine only indexes a small coverage of web pages., Furthermore, the overlapping documents among different individual search engines is very low. Therefore, metasearch engine has been proposed to solve this problem. By combining multiple search engines, metasearch engine has the potential to extend the information retrieval coverage, providing convenience to users [2]. Although metasearch engine improves recall ratio in the big data environment, it is still limited by the precision ratio. The reason is that most of the existing metasearch engines are content-oriented, they use the similarity between the query and returned documents to search, ignoring the perspectives of users. To reach high precision, a personalized metasearch engine which takes user interest into consideration is needed. Generating schedule strategy and providing returned results according to user interest is very helpful to meet user's requirements.

An intelligent agent is an autonomous entity which observes and acts upon an environment, it is proactive and perceptible. A multi-agent system is a computerized system composed of multiple interacting intelligent agents within an environment [3]. Modeling metasearch engine based on multi-

agent system has notable advantages. User interest changes overtime, metasearch engine which based on multi-agent system has the ability to perceive the change of user interest actively because of the characteristics of agent. It is helpful to analyze user intent, schedule underlying search engines and merge results more flexibly according to search context. Multi-agent system improves the adaptability of metasearch engine.

In this paper, we implement a metasearch engine based on multi-agent architecture to improve the personalization level, named "IM search". Agent is utilized to mine user interest from the click-through data, generate schedule strategy, merges results returned by underlying search engines and recommends results which user might be interested in. DCG@N [4] is used to evaluate the proposed metasearch engine.

## II. PROPOSED ARCHITECTURE

The proposed multi-agent architecture is composed of seven kinds of agents. **Interface Agent** is responsible for interacting with users, including receiving queries from users and displaying the returned results to users. **Search Agent** is utilized to evaluate the expertness of underlying search engines and generate the schedule strategy, the offline search engines will be never invoked. It is also in charge of managing the SE Agent. **SE Agent** is responsible for communicating with underlying search engines. Particularly, each SE Agent corresponds to a search engine. The SE Agents detect the states of underlying search engines. If some search engines are offline, SE Agents perceive the offline state and send the signal to Search Agent. **UserInterest Agent** analyzes the click-through data and obtains user interest. **User Group Agent** clusters users into different groups according to the query data. **ResultRecommended Agent** provides recommended results for a specific query that user might be interested in. **ResultMerge Agent** merges all of returned results into a single list.

The multi-agent architecture works as shown in Figure1: a user submits a query to the InterfaceAgent. The Interface Agent sends the query to the Search Agent and ResultRecommended Agent. The Search Agent receives the query, generates the schedule strategy and sends the query to the SE Agents which will be scheduled. The SE Agents communicate with underlying search engines to complete the search task, and then pass on the returned results to the ResultMerge Agent. Meanwhile, the Interface Agent also passes on user information to the UserGroup Agent and UserInterest Agent. The

UserGroup Agent gets the information of group to which the current user belongs based on the Query data, and sends it to the ResultRecommended Agent. After getting the query and user group information, the ResultRecommended Agent generates the recommended results and sends them to the InterfaceAgent. According to user information, the UserInterest Agent obtains user interest factor and passes it on to the ResultMerge Agent. Then the ResultMerge Agent merges all of results returned by SE Agents into a single list, and returns the list to the InterfaceAgent for displaying. The UserInterest Agent analyzes the query data to obtain user interest. After obtaining all of results, the Interface Agent displays results to users.

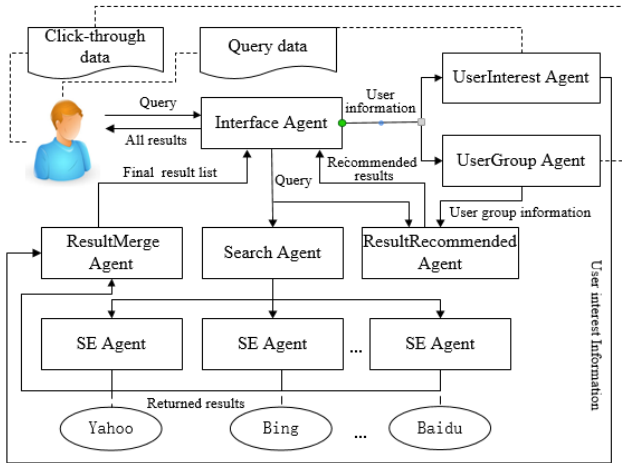


Figure 1. Working diagram of the proposed multi-agent architecture

### III. USER INTEREST MODEL

UserInterest Agent is designed to obtain user interest based on query data. Our previous work [5] describes how to obtain user interest model .

### IV. SCHEDULE STRATEGY

Search Agent is utilized to generate the schedule strategy. In order to evaluate the ability of underlying search engines, According to [6], the expertness model is constructed. When user submits a query, metasearch engine obtains the topic to which the query belongs, then based on the expertness model, the appropriate underlying search engines are selected to complete the search task.

### V. RESULT MERGING METHOD

ResultMerge Agent is utilized to merging the results returned by underlying search engines. When the underlying search engines returns results, the metasearch engine conducts word segmentation for each document and identifies the topic to which document belongs. The topic of user interest is matched with the topic of each returned document so that the user interest factor about the document can be obtained. The final score assigned to a document with user interest can be found in our previous work [5].

## VI. RESULT RECOMMENDATION

Providing result recommendation will help users to find useful information with less effort. ResultRecommended Agent is responsible for recommending user wanted results. Both explicit information and implicit information are used to generate recommendation. This paper obtains explicit information based on the registration of a user, which is helpful to find the similar users in the system. The click-through data is the implicit information. From the click-through data, the relevant documents for a query will be found. The recommendation is generated among similar users. The details can be found in our previous work [7].

## VII. EXPERIMENTAL RESULTS

In this section, the performance of the proposed methods is discussed. “IM search” is a WWW metasearch engine which combines the search engines “Youdao”, “Baidu”, “Bing”, “Yahoo”, and “Sogou”. Two users with different interests are designed to log in IM Search, User1 and User2. For the same query “lincoln”, the results pages for User1 and User2 are shown in Figure 3 and Figure 4 respectively. It is obvious that these two users have different retrieval results. What’s more, IM Search recommends some results for User1. The recommended results are at the top of the result list.

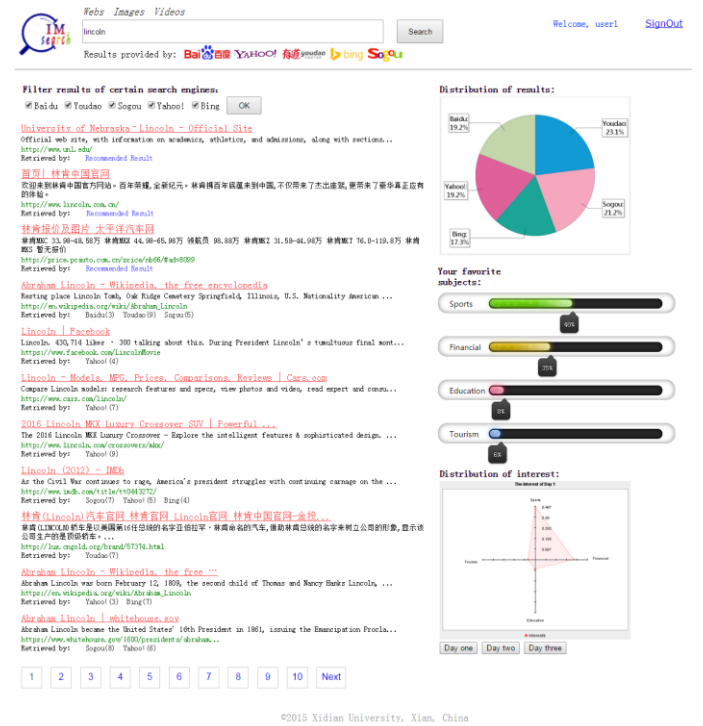


Figure 3. The returned pages for User1

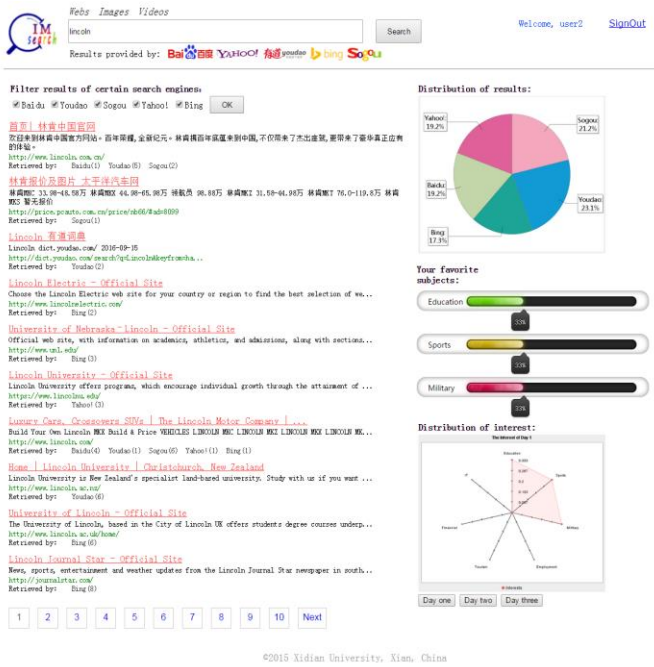


Figure 4. The returned pages for User2

DCG@N (Here, N=10) is utilized to evaluate the precision of the personalization mechanism. Five users are invited to use IM Search for a period of time. Then each user is asked for request 10 queries with login and logout status respectively, and the mean value of DCG@N for the 10 queries is calculated. The results are shown in TABLE. 1. We can see, Users get better results after logging in the system. Because the personalized mechanism only available with login status.

TABLE I. THE MEAN VALUES OF DCG@N

	user1	user2	user3	user4	user5
<b>Login</b>	18.12	17.25	17.93	18.57	16.35
<b>Logout</b>	17.72	16.75	16.31	16.52	16.10

### VIII. CONCLUSIONS

This paper presents a personalized metasearch engine based on multi-agent architecture. By collecting user's click-through data, the metasearch engine has the ability to mining user

interest, schedule the appropriate underlying search engines and obtain the personalized results. According to the group members' behaviors, it can also generate recommended results for users. Experimental Results show that the proposed metasearch engine performs better on precision. But there are still open issues ahead needed to address: 1).User interest is obtained based on the click-through data. But others user behaviors, such as the browsing time, download history are also significant for analyzing user interest. 2).Recommending personalized query words for different users is also necessary.

### ACKNOWLEDGMENT

This work is supported by the Projects (JBZ171001) supported by the Fundamental Research Funds for the Central Universities of China, Projects (2017073CG/RC036 (XDKD004)) supported by the Technology Program of Xi'an, Projects (JB171004, BDY221411, K5051223008) supported by the Fundamental Research Funds for the Central Universities of China, Projects (61373045, 61672401) supported by the National Natural Science Foundation of China; Project (315\*\*\*10101) supported by the Pre-Research Project of the "Thirteenth Five-Year-Plan" of China.

### REFERENCES

- [1] A. H. Keyhanipour , et al. "Aggregation of web search engines based on users' preferences in WebFusion." Knowledge-Based Systems. vol.20, no.4, pp.321-328, 2007
- [2] W. Meng, C. Yu, and K. L. Liu. "Building efficient and effective metasearch engines." AcM Computing Surveys. vol.34, no.1, pp.48-89, 2001
- [3] Y. Lin, P. Descamps, N. Gaud, et al. "Multi-Agent System for intelligent Scrum project management." Integrated Computer Aided Engineering, vol. 22, no. 3, pp.281-296, 2015.
- [4] K. Zhou, et al. "Learning the Gain Values and Discount Factors of DCG." IEEE Transactions on Knowledge & Data Engineering ,2012.
- [5] M. Wang, Q. Li, Y. Lin, et al. "A personalized result merging method for metasearch engine." The International Conference on Software and Computer Applications. pp.203-207, 2017.
- [6] Li Q, Wang J, Chu H, et al. "Personalization mechanism in an intelligent agent-based meta-search engine." Scientia Sinica, vol.45, no.5, pp.605-622, 2015
- [7] Y. Li , Q. Li, Y. Lin. "A Personalized Result Recommendation Method based on Communities." International Conference on Data Mining, Communications and Information Technology. pp. 6-11, 2017.

# Interval-valued Data Clustering Based on Range Metrics

Sérgio Galdino  
Polytechnic School  
UPE  
Recife, PE, Brazil  
sergio.galdino@ieee.org

Wellington Pinheiro dos Santos  
Department of Biomedical Engineering  
UFPE  
Recife, PE, Brazil  
wellington.santos@ufpe.br

Ricardo Paranhos Pinheiro  
Polytechnic School  
UPE  
Recife, PE, Brazil  
paranhos@gmail.com

**Abstract**—This paper introduces new approach to Data Clustering on interval-valued data. We introduce the Width of Range Distance (City Block and Euclidean) matrix dissimilarity to process hierarchical clustering using Ward method. Both width of Range Distances has good clustering results while preserving the topology of the data.

**Index Terms**—Clustering, Interval-valued Data, Interval Arithmetic

## I. INTRODUCTION

Typically, data we analyse are classical data, which means each observation is a single point in a n-dimensional space. However, sometimes the data are represented intervals [1]. Traditional clustering techniques can be easily applied to interval data types by replacing each interval with a representative (e.g, the median of the points in the interval). However, we have limitations of using representative centroids to replace intervals [2]. We introduce the Width of Range distance (City Block and Euclidean) matrix dissimilarity to process hierarchical clustering using Ward method.

## II. RANGE DISTANCES

The set of real numbers  $x$  satisfying  $\underline{x} \leq x \leq \bar{x}$  is the closed interval  $[x] = [\underline{x}, \bar{x}]$ . The width of  $[x]$  are defined as,

$$w([x]) = (\bar{x} - \underline{x}) \quad (1)$$

The magnitude and the mignitude can both be calculated using the end points of  $[x]$ ,

$$mag([x]) = \max\{|\underline{x}|, |\bar{x}|\}, \quad (2)$$

$$mig([x]) = \begin{cases} \min(|\underline{x}|, |\bar{x}|) & \text{if } 0 \notin [x] \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

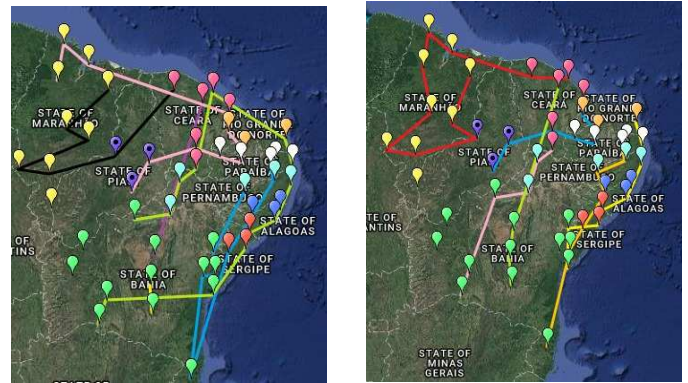
1) Range City Block Distance:

$$\begin{aligned} range[d_1([\mathbf{x}], [\mathbf{y}])] &= \sum_{i=1}^n |[x_i] - [y_i]| \\ &= \left[ \sum_{i=1}^n mig([x_i] - [y_i]), \quad \sum_{i=1}^n mag([x_i] - [y_i]) \right] \end{aligned} \quad (4)$$

2) Range Euclidean Distance:

$$d_2([\mathbf{p}], [\mathbf{q}]) = d_2([\mathbf{q}], [\mathbf{p}]) = \sqrt{\sum_{i=1}^n ([q_i] - [p_i])^2} \quad (5)$$

DOI reference number: 10.18293/SEKE2018-113



(a) Width of Range City Block Dissimilarity.

(b) Width of Range Euclidean Dissimilarity.

Fig. 1: Interval-valued Data Hierarchical Clustering -  $R$  hclust function (method = "Ward.D").

## III. INTERVAL-VALUED CLUSTERING

The data set concerns minimal and maximal of monthly temperatures observed in 50 meteorological stations mounted all over Brazil Northeast from 2017 year (January to November). <http://www.inmet.gov.br/portal/index.php?r=bdmep/bdmep> (2018/12/02). Figure 1 represents the Brazil Northeast map containing the 50 stations over 9 clusters. All stations of the same cluster are drawn with the same line colour.

## IV. CONCLUSIONS

Wards method was used with the dissimilarities matrix using the width from Range City Block Distance Matrix and from Range Euclidean Distance Matrix. We can conclude that the stations located near each other geographically tend to be assigned to the same cluster or to a neighbour cluster.

## REFERENCES

- [1] Billard, L. and Diday, E.: Symbolic Data Analysis: Conceptual Statistics and Data Mining. Wiley, Chichester, (2007)
- [2] S. M. L. GALDINO: Interval-valued Data Clustering Based on the Range City Block Metric. In: SMC 2016, 2016, Budapest. The 2016 IEEE International Conference on Systems, Man, and Cybernetics, 2016.

# A Revisit of Fault-Detecting Probability of Combinatorial Testing

Min Yu Feiyan She Yuanchao Qi Ziyuan Wang\* Weifeng Zhang

School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing, China

\*Corresponding: wangziyuan@njupt.edu.cn

**Abstract**—The lower bound of fault-detecting probability of  $\tau$ -way combinatorial test suite for Boolean-specification testing have been proposed [1]. However, the formula neglected the situation that for different minimal failure-causing schemas, the coverage of test suites may be non-independent events. Hence, multiplying directly the probabilities of non-independent events is incorrect, which causes that the result calculated by previous formula may be greater. In this paper, we give counterexamples to demonstrate the mistakes in the formula derivation. Furthermore, two experiments are designed to illustrate that the actual fault-detecting probabilities and ratios are usually less than the theoretical fault-detecting probabilities calculated from the previous formula.

**Index Terms**—Software testing, combinatorial testing, fault-detecting probability, minimal failure-causing schema.

## I. INTRODUCTION

Combinatorial testing is a well-accepted testing method which has been widely studied and applied [2]. There are growing concerns about the fault-detecting ability of combinatorial testing, due to a controversy about combinatorial test suites effectiveness of detecting faults.

The fault-detecting probability was proposed to evaluate the ease of a combinatorial test suite detecting a fault. Previous works have put forward formulas for fault-detecting probability of combinatorial testing [1][3]. The lower bound of the fault-detecting probability of fixed-strength combinatorial test suite for 2-level system has been raised in [1]. On this basis, the article [3] further proposed the lower bound of the probability that fixed-strength combinatorial test suite detects faults for mixed-level system. However, we find that the events that combinatorial test suites detect faults under different minimal failure-causing schemas may be non-independent sometimes. So it is incorrect to multiply directly the probabilities of non-independent events in the formula. In terms of this mistake, we carry out some relevant researches and experiments.

In this paper, we take Boolean expressions from TCAS system as study object to find counterexamples of the formula and generate combinatorial test suites with different strengths by some classic combinatorial test generation algorithms. By analyzing the test suites of boolean expressions, the fault-detecting probabilities and ratios of the combinatorial test suites are calculated. A contrast of experimental results and theoretical values calculated by the formula is made by us.

The rest of this paper is organized as follows. The 2nd section proposes the formula of fault-detecting probability and puts forward the counterexample of the formula. The 3rd

section is the description of experiments and gives experimental results to illustrate the differences with theoretical values. There is a conclusion finally.

## II. FAULT-DETECTING PROBABILITY

### A. Existing formula of fault-detecting probability

Suppose there are  $m$  minimal failure-causing schemas, with strength  $k_1, k_2, \dots, k_m$  respectively, for a fault. For an arbitrary 2-level  $\tau$ -way combinatorial test suite  $T$ , the probability of event that  $T$  detects this fault should be [1]:

$$p(\tau) = \begin{cases} 1 & : \tau \geq \min_{i=1}^m \{k_i\} \\ 1 - \prod_{i=1}^m \left(1 - \frac{1}{2^{k_i - \tau}}\right)^{C_{k_i}^\tau} & : \tau < \min_{i=1}^m \{k_i\} \end{cases}$$

### B. Counter-example

There is Lemma 6 in [1]: “Suppose there are totally  $m$  schemas, and their strengths are  $k_1, k_2, \dots, k_m$  respectively. Let  $T$  be a 2-level suite, in which all possible  $\tau$ -value subschemas of all  $m$  schemas are covered by at least test case ( $\tau < \min_{i=1}^m \{k_i\}$ ). The probability of event that none of these  $m$  schemas are covered by  $T$  is equal to or less than  $\prod_{i=1}^m \left(1 - \frac{1}{2^{k_i - \tau}}\right)^{C_{k_i}^\tau}$ .”

If there are multiple minimal failure-causing schemas, the lemma 6 calculates probability for each and multiply them. However, only when events are mutually independent, they can be multiplied in probability theory. If a test case covers current minimal failure-causing schema, it may cover others. That is, many overlapping test cases are repeatedly calculated when taking each minimal failure-causing schema as a independent event. The actual probability should discard overlaps.

For instance, there are 2 minimal failure-causing schemas  $S_1=(1\ 1\ 1\ -)$ ,  $S_2=(-\ 1\ 1\ 1\ -)$ ,  $k = 3$ . Their  $\tau = 2$ -value subschemas are  $(1\ 1\ -\ -)$ ,  $(-\ 1\ 1\ -)$ ,  $(1\ -\ 1\ -)$ ,  $(-\ 1\ -\ 1)$  and  $(-\ -\ 1\ 1)$ . Let  $A$  be the event that  $S_1$  is not covered by a 2-way combinatorial test suite, and  $B$  the same event for  $S_2$ . According to Lemma 5 in [1],  $P(A) = P(B) \leq (1 - \frac{1}{2})^3 = \frac{1}{8}$ . According to Lemma 6 in [1],  $P(AB) = P(A) \times P(B) \leq \frac{1}{64}$  and  $P_{\tau=2} = 1 - P(AB) \geq \frac{63}{64}$ .

However, since  $A$  and  $B$  are not independent,  $P(AB) = P(A) \times P(B)$  can't be used to calculate the probability that  $A$  and  $B$  occur at the same time. In fact, if  $A$  occurs, there should be 3 test cases  $(1\ 1\ 0\ x\ x)$ ,  $(1\ 0\ 1\ x\ x)$ ,  $(0\ 1\ 1\ x\ x)$  in combinatorial test suite, in which only  $(0\ 1\ 1\ x\ x)$  has chance to cover schemas  $(-\ 1\ 1\ 1\ -)$ .  $P(AB) = P(A) \times P(B|A) \leq$

$\frac{1}{8} \times \frac{1}{2} = \frac{1}{16}$  since  $P(B|A) \leq \frac{1}{2}$ . There should be  $P_{\tau=2} = 1 - P(AB) \geq \frac{15}{16}$ . It means that, the formula in [1] gives an overvalued low bound of fault-detecting probability.

### III. EXPERIMENTS

#### A. Research questions

*RQ1 (fault-detecting frequency):* In a large number of runnings of combinatorial testing, for each mutant, is the fault-detecting frequency approximate to theoretical fault-detecting probability?

*RQ2 (fault-detecting ratio):* In a large number of runnings of combinatorial testing, for some grouped mutants (from the same original version), is the mean value of the ratios of detected faults approximate to the mean value of theoretical fault-detecting probabilities?

#### B. Experiment setup

We take 20 Boolean expressions that extracted from the TCAS system as the experimental subjects. For each expression, we create mutants by 10 fault types and get 19131 non-equivalent mutants.

Combinatorial test suites with different strengths are generated by some classic combinatorial test generation algorithms including Greedy algorithm [4], DDA algorithm [5], and IPO algorithm [6]. And besides, two algorithms named ReqMerge [7] and DensityRO [8], which were mainly designed for variable strength combinatorial testing, are also utilized in our experiment. When running an algorithm, a random generated seeding test case will be assigned in order to output the rich diversity combinatorial test suites in the large number of runnings of combinatorial testing.

In the first step, for each mutant, the theoretical fault-detecting probability of combinatorial test suite will be calculated according to the formula in [1]. In the second step, for each original Boolean expression from TCAS system, each combinatorial test generation algorithm will generate 100 different  $\tau$ -way combinatorial test suites for  $\tau=2, 3, 4$ . By the running of large number of combinatorial test suites, experimental results could be obtained:

- To answer the 1st research question, for each mutant, the fault-detecting frequency in the large number of the running of combinatorial testing should be collected.
- To answer the 2nd research question, for each combinatorial test suite, the ratio of the number of killed mutants to the total number of mutants should be collected.

#### C. Experimental results

1) *Results for RQ1:* In Fig.1, there are 20 groups of box-graphs in each figure, where each group stands for an original Boolean expression from TCAS. Besides the theoretical faulting-detecting probabilities in the first box-graph of each group, the last 5 box-graphs in each group illustrate fault-detecting frequencies of mutants, which are mutated from current original Boolean expression, in the running of  $\tau$ -way

combinatorial test suites that generated by Greedy algorithm, DDA algorithm, IPO algorithm, DensityRO algorithm, and ReqMerge algorithm respectively, where  $\tau=2, 3, 4$ .

2) *Results for RQ2:* In Fig. 2, there are 20 groups of box-graphs in each figure, where each group stands for an original Boolean expression from TCAS. Besides the theoretical faulting-detecting probabilities in the first box-graph of each group, the last 5 box-graphs in each group illustrate the ratios of killed mutants, which are mutated from current original Boolean expression, by each  $\tau$ -way combinatorial test suite that generated by Greedy algorithm, DDA algorithm, IPO algorithm, DensityRO algorithm, and ReqMerge algorithm respectively, where  $\tau=2, 3, 4$ .

These results indicate that, the mutants' actual fault-detecting frequencies and the mean values of the ratios of detected faults are usually less than the mean values of mutants' theoretical fault-detecting probabilities calculated by formula in [1].

### IV. CONCLUSION

There have been formula about the fault-detecting probability of combinatorial in previous researches. Due to overlooking the independence of probability event, the formula is inaccurate. In this paper, we take 20 boolean expressions as experimental subject to research fault-detecting frequencies and ratios of combinatorial test suites generated by five classic combinatorial test generation algorithms. From the results we come to a conclusion there are obvious deviations between theoretical and real value of fault detection. Because of higher theoretical values than experimental results, the formula in [1] is mistaken.

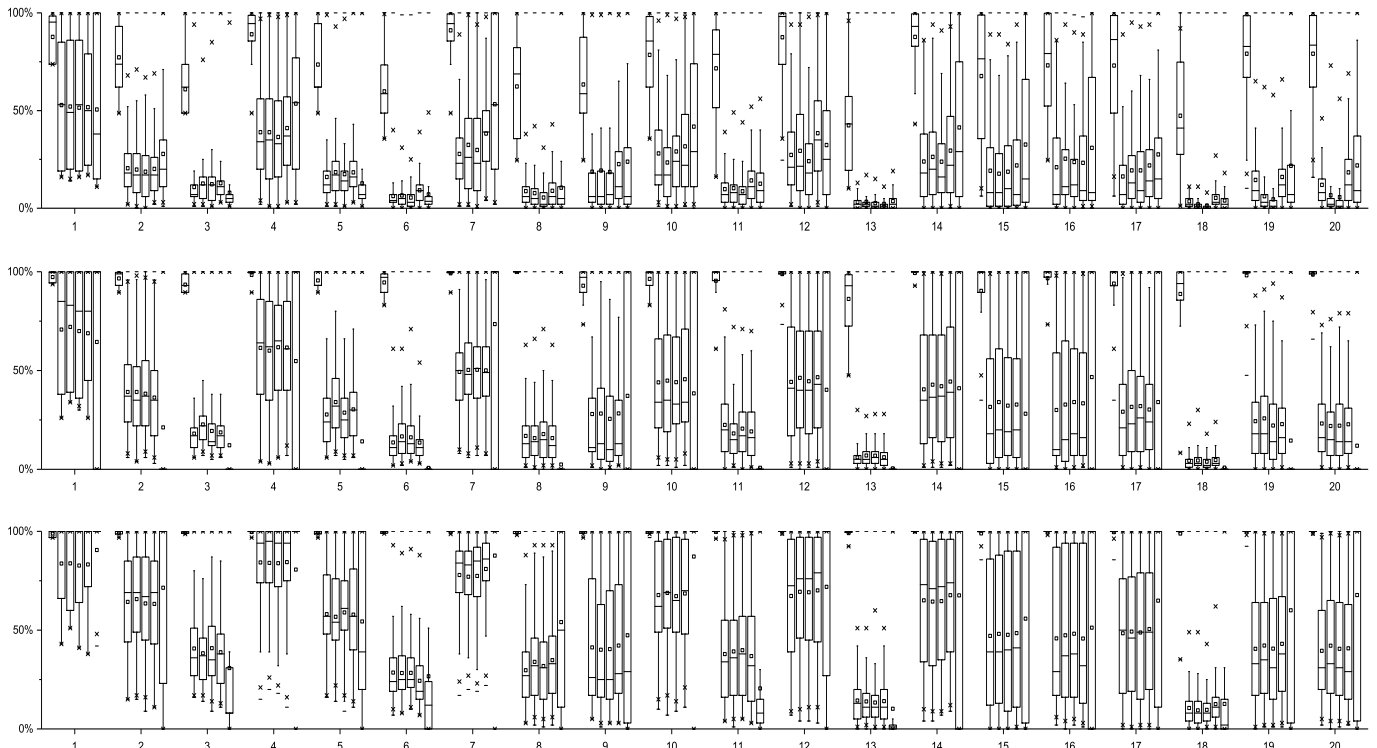
In future, we will correct the mistakes in the formula and figure out the accurate theoretical fault-detecting probability.

### ACKNOWLEDGMENT

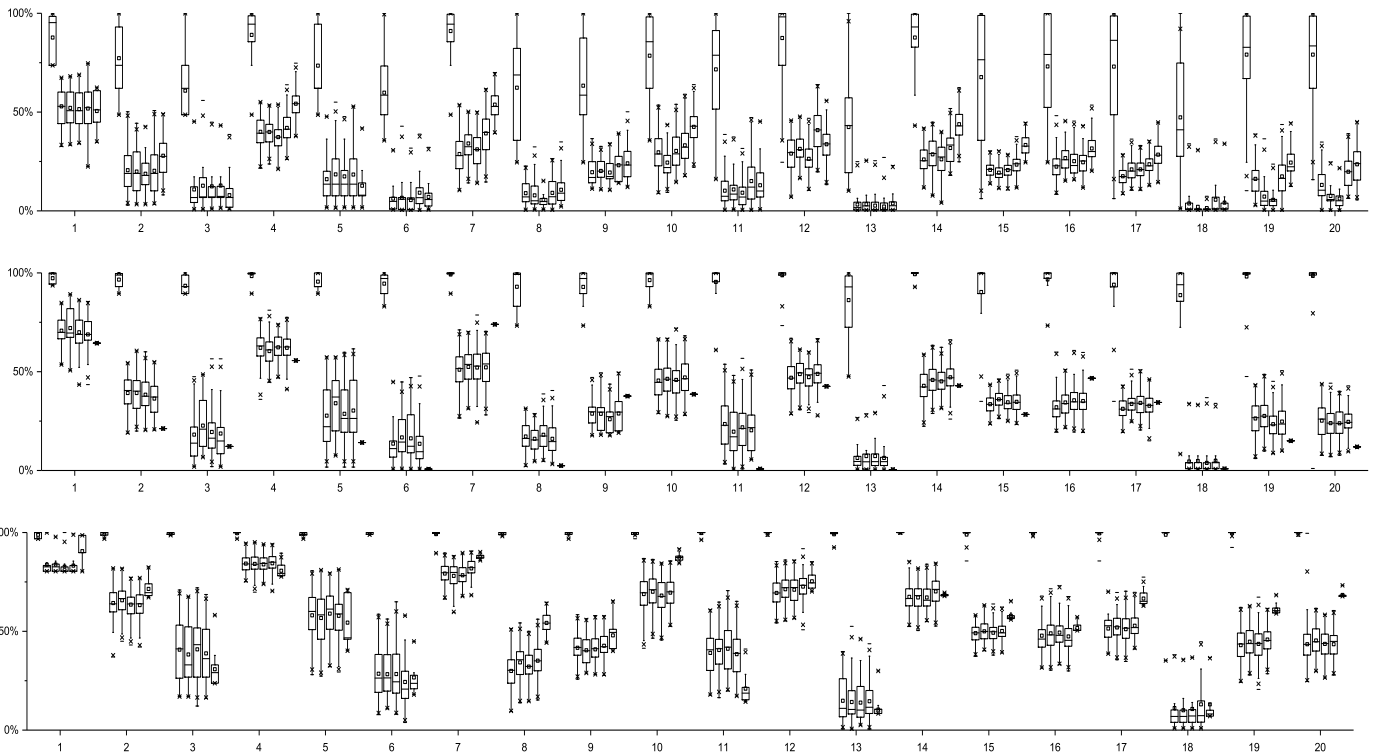
This work is supported by the National Nature Science Foundation of China (61772259).

### REFERENCES

- [1] Ziyuan Wang, Yuanchao Qi. Why Combinatorial Testing Works: Analyzing Minimal Failure-Causing Schemas in Logic Expressions. 2015 IEEE 8th International Conference on Software Testing, Verification and Validation Workshops (ICSTW2015): 4th International Workshop on Combinatorial Testing (IWCT2015).
- [2] C. Nie, H. Leung. A survey of combinatorial testing. ACM Computing Surveys (CSUR), 2011, 43(2): 11.
- [3] Chiya Xu, Yuanchao Qi, Ziyuan Wang, Weifeng Zhang. Analyzing Minimal Failure-Causing Schemas in Siemens Suite. 2016 IEEE 9th International Conference on Software Testing, Verification and Validation Workshops (ICSTW2016): 5th International Workshop on Combinatorial Testing (IWCT2016).
- [4] P. J. Schroeder. Black-box Test Reduction Using Input-Output Analysis: [Dissertation for PhD]. Department of Computer Science, Illinois Institute of Technology, Chicago, IL, USA, 2001.
- [5] Renee C Bryce, Charles J Colbourn. A density-based greedy algorithm for higher strength covering arrays. Software Testing, Verification and Reliability, 2009, 19(1): 37-53.
- [6] Yu Lei, Raghu N Kacker, D Richard Kuhn, Vadim Okun, Jim Lawrence. IPOG-IPOG-D: efficient test generation for multi-way combinatorial testing. Software Testing, Verification and Reliability, 2008, 18(3): 125-148.



**Fig.1** Compare Fault-Detecting Probabilities and Frequencies of  $\tau$ -way Combinatorial Test Suite for 20 Boolean Expressions( $\tau = 2, 3, 4$ )



**Fig.2** Compare Fault-Detecting Probabilities and Ratios of Detected Faults of  $\tau$ -way Combinatorial Test Suite for 20 Boolean Expressions( $\tau = 2, 3, 4$ )

[7] Wang Ziyuan, Nie Changhai, Xu Baowen. Generating Combinatorial Test Suite for Interaction Relationship. In Proceedings of the 4th International Workshop on Software Quality Assurance (SOQUA2007), Croatia, September 3-4, 2007: 55-61

of the 8th International Conference on Quality Software (QSIC2008), Oxford, UK, August 12-13, 2008: 155-160.

[8] Wang Ziyuan, Xu Baowen, Nie Changhai. Greedy Heuristic Algorithms to Generate Variable Strength Combinatorial Test Suite. In Proceedings



# Big Data ETL Implementation Approaches: A Systematic Literature Review

Joshua C. Nwokeji\*, Faisal Aqlan<sup>†</sup>, Anugu Apoorva\*, and Ayodele Olagunju<sup>‡</sup>

\*Comp. & Info. Sys. Dept. Gannon Uni. <sup>†</sup> Indus.Engr., Dept., Penn. State Uni. <sup>‡</sup> Uni., of Saskatchewan;

Email: Nwokeji001@gannon.edu

**Abstract**—Extract, transform, load (ETL) is an essential technique for integrating data from multiple sources into a data warehouse. ETL is applicable to data warehousing, big data, and business intelligence. Through a systematic literature review of 97 papers, this research identifies and evaluates the current approaches used to implement existing ETL solutions. We found that conceptual modeling such as UML, BPMN, and MDA is the most popular approach used to implement ETL solutions. However, innovative approaches such as machine learning, artificial intelligence, and robotics are either under-utilized or not used at all to develop ETL solutions. Additionally, we discuss the implications of these to ETL research and practice.

## I. INTRODUCTION

Organizations are rapidly generating 'big data' from various sources, e.g., social media and e-commerce systems. The term big data is used to define data that are too voluminous and complex to be processed by traditional data processing systems [1]. Big data becomes more meaningful to organizations when analyzed to derive business intelligence that support decision making. Normally, big data analytics starts with integrating the generated data into a data warehouse using various techniques; ETL (extract, transform, load) is a popular technique used for this purpose [1].

A typical ETL process is carried out in three steps. In the first step, data in various formats (e.g., txt, csv, xls) are extracted from different data sources. The second step involves applying transformation techniques such as normalization, filtering, and sorting to clean these data. Finally, the cleaned data are migrated (loaded) into a data warehouse to be processed and analyzed to derive intelligence, knowledge, and wisdom [1]. In this era of big data, ETL research is becoming increasingly important and a means to developing new approaches to solve the growing challenges of data integration in organizations.

As ETL evolve, researchers and practitioners should be aware of the current techniques used to implement ETL solutions, and the implications of these to research and practice. Hence, this paper identifies and evaluates existing ETL implementation techniques from 97 papers selected using systematic literature review (SLR) method. In sections II, III, and IV we respectively discuss SLR method, present and discuss result, and conclude our research.

DOI reference number: 10.18293/SEKE2018-152

## II. METHOD

To meet the objectives of this research, we adopt the method for conducting SLR in software engineering proposed by Kitchenham and Charters [2]. In the paragraphs that follow, we discuss how this research conforms to this method.

*a) Define Review Objectives and Question:* The main objective of this review is to identify and evaluate approaches used in implementing existing ETL solutions. We identified a review question that closely align with this objective:

*RQ:* What approaches are currently used to implement ETL solutions?

*b) Develop Search Strategy:* This include selecting data sources, identifying search keywords, and conducting the search [2]. We selected the following data sources based on their relevance to software engineering and computer science: IEEE Xplore, ACM Digital Library, Science Direct, ProQuest, and Google Scholar. After a series of pilot searches, we identified and selected the keywords shown in Table I. To conduct the search, we combined each keywords in concept [A] with the keywords in concepts [B], and [C] using boolean operators (and/or).

TABLE I: SLR Keywords

Main Concept	Corresponding Keywords
[A] Extract Transform Load	[A1] Extract*, Transform*, Load*; [A2] ETL
[B] Approach	[B1] Framework; [B2] Models; [B3] Tools; [4] Technology; [B5] Software
[C] Quality	[C1] Quality Attributes; [C2] Quality Measures; [C3] Quality Features; [C4] Quality Characteristics; [C5] QoX

*c) Specify Selection Criteria:* After developing the search strategy, the authors discussed and agreed on a set of criteria for including and excluding papers from review. These criteria are described in Table II.

TABLE II: Selection Criteria

Criteria	Include	Exclude
Year	Papers must be published between 2006 and 2016	Papers published before 2006
Relevance	Papers whose title and abstracts are relevant to ETL and its related concepts, as well as the review objectives and questions.	Papers that do not relate to the ETL and other key concepts of this research
Quality	Peer reviewed papers in journals, conferences, and workshops.	Non-peer reviewed papers. Also keynotes and presentations.
Rigor	Papers that demonstrate rigor through the use of appropriate scientific research and validation methods.	Papers that do not use appropriate scientific research and validation methods

d) *Extract Data*:: We searched the data sources with the keywords shown in Table I, this returns a total of 865 papers. Then, we removed duplicate papers and we excluded papers that do not meet the year criterion described in Table II. Afterwards, we applied the relevance criteria to exclude papers whose titles and abstracts are not related to the main concepts of the review. Finally, we selected a total of 97 papers as primary studies, after we applied the quality and rigor criteria. Please note that we did not show these primary studies due to space restrictions.

### III. RESULT AND DISCUSSION

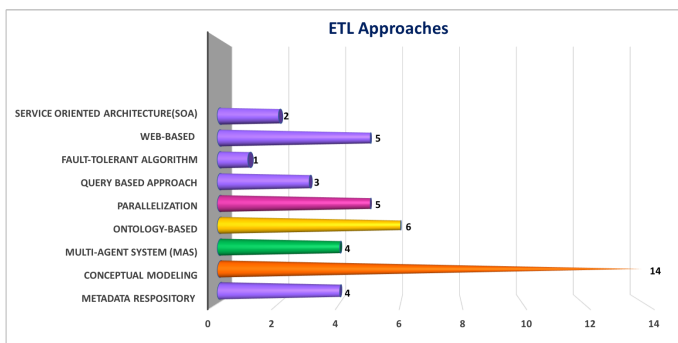


Fig. 1: Approaches used to Implement ETL Solutions

a) **RQ: What approaches are currently used to implement ETL solutions?:** We found nine (9) popular implementation approaches from selected papers (primary studies). As shown in Figure 1, these approaches are plotted in the vertical axis, while the horizontal axis shows the number of papers that reported them. For clarity, we also listed them here as follows : (i) Service oriented architecture (SOA) (ii) Web-based technologies, e.g., semantic web (iii) Fault-tolerant algorithm. (iv) Structured query languages (SQL). (v) Parallelization (parallel computing paradigm), e.g., Map Reduce. (vi) Domain ontology (vii) Multi-agent system (MAS).(viii) Conceptual modeling e.g., unified modeling language (UML) and business process modeling (BPMN) (ix) Meta-data repository.

b) **Variety of Approaches:** The results from this systematic literature review reveal that a variety of approaches are used to develop ETL solutions, see Figure 1. The leading among these is conceptual modeling approaches, such as the unified modeling language (UML), business process model and notation (BPMN), and model driven architecture (MDA) or model driven development (MDD). Conceptual models tend to represent real world concepts at level of abstraction that facilitates easy comprehension and analysis [3].

Although the application of conceptual modeling to ETL offers some advantages such as automatic code generation [4]; the overly emphasis on this, at the expense of other approaches, calls for concerns. For instance, while conceptual modeling approaches appear useful in present times; there is no clear research direction and indication of how conceptual modeling approaches can be applied to develop effective solutions to address future exponential increase in data complexity, volume, and heterogeneity.

More so, the focus on conceptual modeling appears to overshadow the application of new and innovative approaches such as artificial intelligence, machine learning, and robotics to developing ETL solutions. It is clear if or how these innovative can be applied to develop ETL solutions. This calls for research questions that should be focus of current studies in ETL. For instance, can artificial intelligence be used to automate data extraction from heterogeneous sources, or can machine learning and robotics be applied to transform and load data? Hence, we expect the future directions of ETL research to be in the area of developing new approaches based on current innovative technologies such as machine learning and artificial intelligence. These new approaches will be driven by the new data requirements, and the technology advancements.

### IV. CONCLUSION AND FUTURE WORK

This paper presents a systematic literature review, conducted to identify and discuss current approaches used to implement ETL solution; quality attributes to be considered when selecting ETL approach; and prevailing challenges in ETL research. In addition, we identify and discuss current trends in ETL research focusing on application domain, frequency or articles published per year and number of articles published per region. Based on the results identified from 96 papers, published between 2006 and 2016, we found that approaches for implementing ETL solutions overly focus on conceptual modeling, neglecting emerging and innovative approaches such as machine learning.

These challenges should call for concerns and questions among big data researchers and practitioners. The concern, for instance, would be if we (humans and machines) will ever catch up with the demands, and solve the problems, of big data integration and management. The questions are: Will big data eventually over grow human intelligence and machine capacity? Can we invent newer approaches (beyond what we currently have) to dealing with big data management? certainly these questions can be answered through intensified concerted effort by ETL researchers and practitioners. These should be the focus of future work.

### REFERENCES

- [1] V. N. Gudivada, R. A. Baeza-Yates, and V. V. Raghavan, "Big data: Promises and problems." *IEEE Computer*, vol. 48, no. 3, pp. 20–23, 2015.
- [2] B. Kitchenham and S. Charters, "Guidelines for performing Systematic Literature reviews in Software Engineering Version 2.3," *Engineering*, vol. 45, no. 4ve, p. 1051, 2007.
- [3] J. C. Nwokeji, F. Aqlan, B. Barn, T. Clark, and V. Kulkarni, "A modelling technique for enterprise agility," *Proceedings of the 51st Hawaii International Conference on System Sciences*, 2018.
- [4] Z. El Akkaoui, E. Zimanyi, J. N. Mazon Lopez, J. C. Trujillo Mondejar *et al.*, "A bpmn-based design and maintenance framework for etl processes," *International Journal of Data Warehousing and Mining*, 2013.

# Research on Crowd-based mobile application testing platforms

Wenguang Xie

School of Airworthiness  
Civil Aviation University of China  
Tianjin, China  
caucxwg@foxmail.com

Kenian Wang

School of Airworthiness  
Civil Aviation University of China  
Tianjin, China  
wangkenian@126.com

**Abstract**—In recent years, mobile applications market develop rapidly. In this paper, a research is provided for platforms of crowd-based mobile app testing. The infrastructure, framework, work flow of crowdsourcing for mobile applications are explained in this paper.

**Keywords**—Crowdsourcing; testing platforms; work flow

## I. INTRODUCTION

Because of the wide application of crowdsourcing, it has received extensive attention in the academic circle in recent years. Many experts and scholars have made a summary of the relevant research work on crowdsourcing in top conferences and periodicals. Ke Mao studied the applications of the crowdsourcing in the life cycle of software engineering[2]. Doan categorized the crowdsourced systems according to the type of problem and the way of cooperation[3]. Zhao Y summarized the research status of crowdsourced technology and predict the future research area[4]. Additionally, there is a small amount of content related to mobile application testing. Jerry Gao[5], Kirubakaran[6] gave the basic concepts of mobile application testing, and analyzed the needs, challenges and main problems of mobile application testing. However, they didn't combine mobile application with crowdsourced testing. In this paper, we focus on the applications of Crowdsourced testing technology in mobile applications. We then introduce the concept and advantage of crowd-based mobile application testing. Next, we study three key issues, and conclude open issues and opportunities for future research on crowd-based mobile application testing.

## II. NORMAL CROWDSOURCED TEST INFRASTRUCTURE

Crowdsourced testing needs intermediary to connect mobile app developers with thousands of crowdsourced testers. The normal infrastructure for most of crowdsourced mobile test platforms is shown in Figure 1.

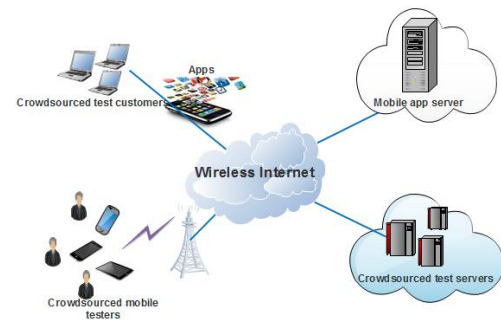


Figure 1. Crowdsourced Test infrastructure

Crowdsourced test customers upload under-test mobile apps, assign test requirements & criterion, and submit test tasks to crowdsourced test servers.

Crowdsourced mobile testers apply and execute mobile test tasks, and report test results to crowdsourced test servers. Crowdsourced mobile testers may be professional test engineers, or normal mobile users.

Crowdsourced test servers manage under-test apps, test tasks, and crowdsourced testers, assign test tasks to crowdsourced testers, collect test results, and generate test bills.

Mobile app servers communicate with mobile apps on mobile devices through wireless networks, and provide information and services for mobile apps.

## III. CROWDSOURCED TEST FRAMEWORK

Crowdsourced test server provides test services using crowdsourced test approaches. As shown in Figure 2 below, we present a recommended framework for Crowdsourced test server, which includes a set of management services, a set of test services, and mobile test device cloud.

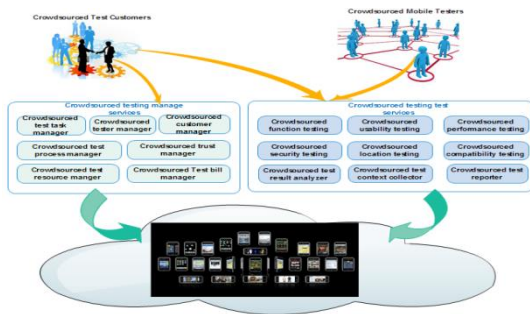


Figure 2. Recommend architecture for crowdsourced test framework

There are seven management services: crowdsourced test task manager, crowdsourced tester manager, crowdsourced customer manager, crowdsourced test process manager, crowdsourced trust manager, crowdsourced test resource manager and crowdsourced test bill manager.

Crowdsourced test server provides various types of crowdsourced test services, including function testing, usability testing, performance testing, security testing, location testing, and compatibility testing. There are also three supporting services, including test results analyzer, test context collector, and test reporter.

Mobile test device cloud provides test resources for crowdsourced test services. Crowdsourced mobile testers can use test devices in cloud to perform test tasks.

#### IV. WORK FLOW

The main steps of crowd-based mobile application testing are: crowdsourced tester management, crowdsourced customer management.

##### A. Crowdsourced Tester Management Process

As shown in Figure 3, there are five core steps for crowdsourced mobile tester management.

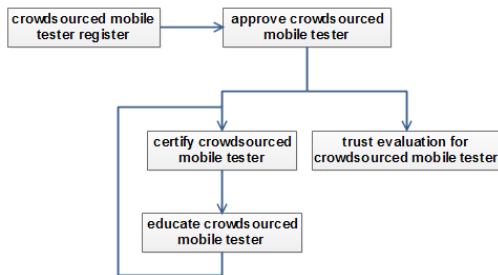


Figure 3. Core steps for crowdsourced mobile tester management

Step 1, crowdsourced mobile tester register. Testers submit basic personal information and test experiences, and apply to be crowdsourced mobile testers.

Step 2, approve crowdsourced mobile tester. Crowdsourced test server investigates information about crowdsourced mobile testers, and approves or denies the applications.

Step 3, certify crowdsourced mobile tester. Crowdsourced test server evaluates the abilities of

crowdsourced mobile testers, and then certifies the levels for testers. The levels of testers will affect their payments.

Step 4, educate crowdsourced mobile tester. Crowdsourced test server provides training courses for crowdsourced mobile testers to improve their test skills. After they have finished the training courses, crowdsourced mobile testers can apply for certifying higher levels.

Step 5, trust evaluation for crowdsourced mobile tester. Trustworthiness of crowdsourced mobile testers will be evaluated by checking dishonesty in test results.

##### B. Crowdsourced Test customer management process

Crowdsourced test customer management include three core activities:

Step 1, crowdsourced test customer application. Customers submit basic personal or corporate information, and apply to be crowdsourced test customers.

Step 2, crowdsourced test customer approving. Crowdsourced test server investigates information about crowdsourced test customers, and approves or denies the applications.

Step 3, trust evaluation for crowdsourced test customer. Trustworthiness of crowdsourced test customer will be evaluated by checking if he is paying for crowdsourced mobile testers honestly.

#### V. CONCLUSION

The infrastructure, framework, work flow of crowdsourcing for mobile applications are explained in this paper. As there are more constructions and deployments of mobile apps on devices, more research on key technology of crowdsourced testing should be given.

#### ACKNOWLEDGMENT

#### REFERENCES

- [1] Ke Mao, Licia Capra, Mark Harman, Yue Jia, "A survey of the use of crowdsourcing in software engineering." The journal of systems and software, vol.126, pp.57-84. May 2016. K. Elissa, "Title of paper if known," unpublished.
- [2] Doan A, Ramakrishnan R, Halevy A Y, "Crowdsourcing systems on the word-wide web." Communications of the ACM, vol.54, pp.86-96, February 2011.
- [3] Zhao Y, Zhu Q, "Evaluation on crowdsourcing research: Current status and future direction." Information Systems Frontiers, vol.16, pp.417-434, October 2014.
- [4] Jerry Gao, X. Bai, W. T. Tsai, and T. Uehara, "Mobile application testing: a tutorial", IEEE Computer, vol.47(2), pp.26-35, April 2014.
- [5] .Kirubakaran, B., and Karthikeyani, V., "Mobile application testing – Challenges and solution approach through automation," In: Proceedings of the 2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering (PRIME). 2013, pp.79-84.

# Mobile App Development Using Software Design Patterns

Nicole Barakat, Doan Nguyen  
California State University, Sacramento 95819

## I. Introduction

Computer Science (CSC) 133 course is an Object-Oriented Computer Graphics Programming class offered at California State University, Sacramento. This poster paper describes a class project utilized in Spring 2018. The course materials where this assignment was based from the pedagogical study [2] that redesigned CSC 133 by utilizing CodenameOne (CN1) [1], a cross-platform mobile application development environment, to teach topics of this course that originally utilized Java Standard Edition and desktop application development. With the purpose of incorporating mobile application development experience to the existing course, this pedagogical study updated the course content and modified the original course materials [3] (e.g., slides, projects, and code samples) developed mainly by colleagues who previously taught CSC 133. The specification and design of this class project in Spring 2018 (i.e., Asteroid game) are also originated from the original CSC 133 course materials. The work here converted the Asteroid game from Java into CN1 by using the converted project (Race Car game) from [2] as its model. The most challenging aspect was to find the proper components in CN1 which can perform the comparable functions and at the same time preserving the core design of the system [3].

As inferred by the class title, this class focuses on the fundamental concepts of the object-oriented (OO) paradigm, introduction to Computer Graphics, as well as Mobile App Development. The object-oriented paradigm encompasses many pertinent topics such as: polymorphism, inheritance, encapsulation, and abstraction. These concepts are supported through formalisms such as UML diagrams and software design patterns. Emphasis is put on implementing event-driven systems through the study of computer graphics.

These concepts are all tied together through the construction of an enticing cross-platform mobile application. CN1 is the mobile app development environment tool chosen due to its Java focus and cross-platform capabilities. The Java programming language works exceptionally well for the purpose of this course, being that it is an object-oriented, ubiquitous, and versatile language. Many applications are limited to Android/iOS, which makes this deployment even more salable and overall appealing. Throughout the semester, the students are taught how to develop a mobile application given the knowledge of efficient structuring principles and tools. Incentive for hard work is encouraged by the semester end

goal, the creation of an “Asteroid Game.” This paper is organized into three parts. The first part describes the design aspects of the application. The second part highlights the testing areas. The paper ends with remarked conclusion and future work.

## II. Design

A specification is given for the assignment [2,3], requirements are analyzed, then a UML class diagram is created. This diagram gives an excellent visualization of the organization and basic structure needed. It shows many of the important relationships and concepts needed to structure and execute an effective system. The underlying structure of this application is largely inspired by, and incorporates, many software design patterns including an emphasis on the Model View Controller (MVC) architecture. The MVC architecture allows for the organization and compartmentalization of code into three main modules, which in turn largely contributes to scalability and maintainability. In the context of this course, an interactive game application is created with the following modules: the Game class is the user’s “controller”, the GameWorld class is the “model” that holds the majority of the complex data which is accessed and manipulated by the controller, and the PointsView and MapView are the views through which the game state may be conveyed to the user. Figure 1 shows a condensed UML class diagram of the overall system architecture.

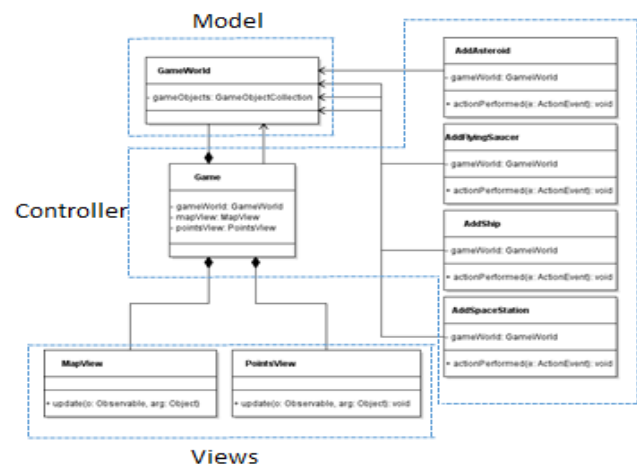


Figure 1. Condensed UML of the game’s overall architecture

This application is further coordinated using various creational, structural, and behavioral design patterns such as the Proxy, Composite, Command, and Observer/Observable patterns.

**Proxy Design Pattern:** The Proxy structural pattern is used as a protection proxy to protect the game's state by prohibiting the views from modifying the game world. This structure defines an interface which strictly specifies what methods and actions a client program may perform. For example, IGameWorld contains the restricted methods which will be implemented by both the real GameWorld class, as well as the GameWorldProxy class.

**Composite Design Pattern:** The objects of the "Asteroid Game" are organized in a hierarchical manner: a GameObject can be a FixedObject or a MoveableObject, a MoveableObject can then be further divided into: Ship, Asteroid, FlyingSaucer, or Missile. As can be seen, some objects, such as MoveableObject are groups of other objects. Additionally, these moveable objects, which are all moveable, are treated uniformly through inheritance and implementation of an IMoveable interface.

**Command Design Pattern:** A user has a multitude of commands available upon beginning the game. Maintenance of state information of each command is done by encapsulating the various commands that the player may invoke. The constructor of the main Game class creates strictly one instance of each command object, and reuses that object as needed. Despite the origin of invocation of the command, the execution of a code block is the same. This strategy, therefore, avoids the need for multiple copies of code that perform the same task.

**Observer/Observable Design Pattern:** The MapView and PointsView classes are registered as observers of the observable, GameWorld. Whenever any changes are made that would affect the state of the views, GameWorld notifies its observers of updates. This relationship allows for proper coordination and communication between participating classes when changes occur.

#### Animation, Objects Interaction, Sound

A system timer's tick function is generated. For each tick, an object movement is computed using its current positions, headings, and speeds. Collisions are detected and handled polymorphically through the use of an ICollider interface containing collidesWith and handleCollision methods which are implemented by all objects that have the capability of colliding. The collidesWith method checks whether or not two given objects are colliding or not by using either a 2D bounding circle. If a collision is confirmed, it must be handled appropriately in the handleCollision method; for an example, if a missile hits an asteroid, both objects must be deleted from the world and the user score increases.

### III. Testing:

The verification and validation of this program is done using the test cases derived from the system specification [2,3]. We are focusing on black box testing. Additionally, a few model based models were also using to verify different stages of the game. For an example a sequence of the following actions: pressing pause, turning off sound, and pressing play, should yield a result of the sound not coming back on. A sample of Asteroid Game GUI is shown in Figure 2.

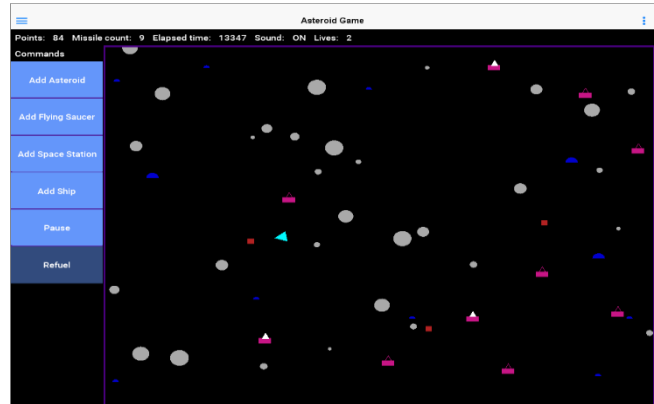


Figure 2. A sample of the Asteroid Game running under a CN1 simulator.

### IV. Conclusion:

A project such as this, provides invaluable knowledge and skills to participating students, consequently building them up for future success. The production of a game showcases a rich understanding of vital concepts needed for being prosperous in the Computer Science field. No matter what area of expertise one pursues, the lessons learned in a class like this aids in a stronger resume, larger repertoire, and well-rounded abilities.

### References

- [1] CodenameOne Guide: Build Cross Platform Apps using Java, <http://codenameone.com/manual/index.html>, Accessed May 2018.
- [2] Pinar Muyan-Ozcelik, A Hands-on Cross-Platform Mobile Programming Approach to Teaching OOP Concepts and Design Pattern, In Proceedings of Software Engineering Curricula for Millennials (SECM) Workshop at ACM/IEEE 39th International Conference on Software Engineering (ICSE), May 20-28, 2017, Buenos Aires, Argentina, doi: 10.1109/SECM.2017.12.
- [3] J. Clevenger and S. Gordon, "CSC 133 - Object-oriented Computer Graphics Programming Lecture Notes and Assignments," Spring, 2014, California State University, Sacramento.

# BoolMuTest: A Prototype Tool for Fault-Based Boolean-Specification Testing

Ziyuan Wang\* Min Yu

School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing, China

\*Corresponding: wangziyuan@njupt.edu.cn

**Abstract**—In order to perform mutation testing for general-form Boolean specifications, a prototype tool called *BoolMuTest* is designed for fault-based Boolean-specification testing. There are several function modules including generating mutants for general-form Boolean expressions, finding all possible test cases to kill a mutant, analyzing minimal failure-causing schemas for a mutant, calculating mutation score for a give set of mutants, and etc. All these functions are provided via command-line programs.

**Index Terms**—Software testing, Boolean-specification testing, mutation testing, fault-based testing, prototype tool.

## I. INTRODUCTION

Fault-based Boolean-specification testing is one of important weak mutation testing techniques, since the running paths of program are usually dependent on Boolean-specifications in predicates. People usually pay their attention on 10 mutation types including ASF, CCF, CDF, ENF, LNF, LRF, MLF, ORF, SA0, and SA1 in the field of Boolean-specification testing [1]. In order to perform mutation testing for general-form Boolean specifications, a prototype tool called *BoolMuTest* is designed. It can be utilized to generate mutants for general-form Boolean expressions, find all possible test cases to kill a mutant, analyze minimal failure-causing schemas for a mutant, calculate mutation score for a give set of mutants, and etc.

## II. FUNCTION MODULES

### A. CreateBoolMutant

*CreateBoolMutant* can be utilized to generate mutants for given general-form Boolean expressions with given mutation types. The command format is:

```
CreateBoolMutant origin_expr_file mutant_type_file  
mutant_expr_file [-disp]
```

Where the *origin\_expr\_file* is an input file including original Boolean expressions. The *mutant\_type\_file* is an input file including mutation types. The *mutant\_expr\_file* is output file including mutant expressions with given mutation types. And the [-disp] is an option parameter to print all the expressions (include original and mutant expressions) on the screen.

An example input file including original Boolean expressions is shown as follow, where there are two original Boolean expressions. To represent Boolean expressions by plain text, operators  $\wedge$ ,  $\vee$ , and  $\neg$  are replaced by \*, +, and ! separately.

```
a*(!b+!c)*d+e  
a1*!a5+(!a2+!a3+!a1)*a4+a5
```

An example input file including mutation types is shown as follow, where there are two types ASF and ENF.

```
ASF  
ENF
```

The output file generated by *CreateBoolMutant* for above two input files should is shown as follow.

```
#Original Expression File: input.txt  
#Original Expression 1: a*(!b+!c)*d+e  
#Mutation Type: ASF  
a*!b+!c*d+e  
#End Mutation Type  
#Mutation Type: ENF  
a*!(!b+!c)*d+e  
#End Mutation Type  
#End Original Expression  
#Original Expression 2: a1*!a5+(!a2+!a3+!a1)*a4+a5  
...
```

### B. BoolCodeTransform

*BoolCodeTransform* can be utilized to translate original Boolean expressions or mutant Boolean expressions in a given file to C language codes. The command format And is:

```
BoolCodeTransform expr_file code_file [-mu]
```

If [-mu] is used, mutant expressions will be translated; otherwise, original expressions will be translated. The *expr\_file* is an input file that includes Boolean expressions, and the *code\_file* is a C language header file.

By assuming the name of file that contain original expressions is “input.txt”, the C code file for original Boolean expressions in section III.A should be:

```
bool input1(bool a, bool b, bool c, bool d, bool e)  
{  
    return a&&(!b||!c)&&d||e;  
}  
bool input2(bool a1,bool a2,bool a3,bool a4,bool a5)  
{  
    return a1&&!a5||(!a2||!a3||!a1)&&a4||a5;  
}
```

And the code file for first two mutant Boolean expressions in section III.A should be:

```
bool input1ASF1(bool a, bool b, bool c, bool d, bool e)  
{  
    return a&&!b||!c&&d||e;  
}  
bool input1ENF1(bool a, bool b, bool c, bool d, bool e)  
{  
    return a&&!(!b||!c)&&d||e;  
}
```

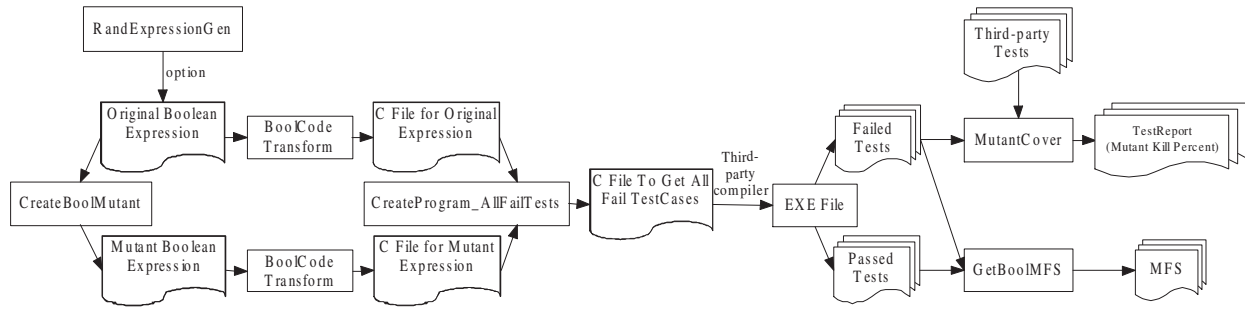


Figure.1 Work process of BoolMuTest

### C. CreateProgram\_AllFailTests

*CreateProgram\_AllFailTests* can be utilized to create a C++ program that finds all possible test cases to kill given mutants. The command format is:

```
CreateProgram origin_expr_code_file
mutant_expr_code_file cpp_program_file
[-W]|[-WO]|[-L]|[-LO]
```

Where both *origin\_expr\_code\_file* and *mutant\_expr\_code\_file* are input files generated by *BoolCodeTransform*. If [-W] or [-WO] is used, the created C++ program could be compiled and run in the Windows environment. If [-L] or [-LO] is used, it could be compiled and run in the Linux environment.

The C++ program, we named as *GetAllFailTests*, could be run as the following format on Windows/Linux console:

```
GetAllFailTests [-i]|[-b] target_directory
```

Where the [-b] means that test cases are stored as binary string, and [-i] means that test cases are stored as an integer. For each mutant, all test cases that kill such a mutant are stored in an independent file in the folder *arget\_directory*.

### D. GetBoolMFS

The minimal failure-causing schemas reflect characteristic of failed test cases [2]. *GetBoolMFS* can be utilized to find minimal failure-causing schemas by comparing passed test cases and failed test cases. The command format is:

```
GetBoolMFS expr_common_name min_expr_index -
max_expr_index test_directory target_directory
[ASF | CCF | CDF | ENF | LNF | LRF | MLF |
ORF | SA0 | SA1 | VNF | VRF] [min_mutant_index
- max_mutant_index] | [min_mutant_index -] |
[mutant_index]*
```

Where *expr\_common\_name* is used to identify input files. Parameters *min\_expr\_index* and *max\_expr\_index* indicate the range of index of original expressions. Parameters *test\_directory* and *target\_directory* indicate folders that store failed test cases and final results. And parameters *min\_mutant\_index* and *max\_mutant\_index* indicate range of index of mutant expressions.

### E. MutantCover

In mutation testing, the mutation score is used to evaluate quality of given test cases. *MutantCover* can be utilized to

evaluate percent of killed mutants in Boolean-specification testing. The command format is:

```
MutantCover origin_expr_name mutant_type
all_fail_tests_dir input_tests_file
```

Where parameter *origin\_expr\_name* and *mutant\_type* are used to indicate mutations. The folder *all\_fail\_tests\_dir* store files that contain all failed test cases for each mutant. Test cases under evaluation are stored in *input\_tests\_file*.

### F. RandExpressionGen

There is a program called *RandExpressionGen* to generate Boolean expressions randomly according to given configurations. The usage of *RandExpressionGen* is omitted here since the limitation of the length of pages.

## III. CONCLUSION

The work process of *BoolMuTest*, which is a prototype tool for fault-based Boolean-specification testing, could be found in Figure 1. The *BoolMuTest*, which provides six command-line programs, supported many previous experimental studies in the field of fault-based Boolean-specification testing [3][4][5]. There should be some future works of making the tool user friendly by design graphic user interface. The current version could be found in <https://github.com/princeyuan/BoolTest/>.

## ACKNOWLEDGMENT

This work is supported by the National Nature Science Foundation of China (61772259).

## REFERENCES

- [1] Z. Chen, T. Y. Chen, B. Xu. A Revisit of Fault Class Hierarchies in General Boolean Specifications. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 2011, 20(3): 13.
- [2] C. Nie, H. Leung. The Minimal Failure-causing Schema of Combinatorial Testing. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 2011, 20(4): 15.
- [3] Ziyuan Wang, Yuanchao Qi. Why Combinatorial Testing Works: Analyzing Minimal Failure-Causing Schemas in Logic Expressions. 2015 IEEE 8th International Conference on Software Testing, Verification and Validation Workshops (ICSTW2015): 4th International Workshop on Combinatorial Testing (IWCT2015).
- [4] Chunrong Fang, Zhenyu Chen, Baowen Xu. Comparing Logic Coverage Criteria on Test Case Prioritization. *Science China Information Science*, 2012, 55(12): 2826-2840.
- [5] Min Yu, Ziyuan Wang, Yuanchao Qi, Feiyan She, Weifeng Zhang. A Revisit of Fault-Detecting Probability of Combinatorial Testing for Boolean-Specifications. 30th International Conference on Software Engineering and Knowledge Engineering (SEKE2018).





## Author Index

Abuaiadah, Diab	368
Acco Tives Leão, Heloise	502
Acuña, Silvia T.	182
Affonso, Frank	286
Aguiar, Rui	276
Akram, Junaid	354
Alam, Dewan Mohammad Moksedul	397
Albuquerque, Regina	560
Almeida, Hyggo	40, 149, 474, 496
Angeles, Maria Del Pilar	225
Antonelli, Leandro	524
Anugu, Apoorva	714
Aqlan, Faisal	714
Arshad, Rehman	572
Assunção, Joaquim	364
Badri, Mourad	653
Bakhti, Khadidja	131
Barakat, Nicole	718
Baset, Selena Sohaila	29
Benferhat, Salem	64
Bennin, Kwabena	368
Bhushan, Bharat	256
Bi, Zhongqin	11
Bischoff, Vincius	304
Bischoff, Vinicius	326, 372
Blanc, Xavier	518
Bosu, Michael	368
Bouraoui, Zied	64
Busch, Kiana	104
Caetano, Josemar	445
Caires, Vivyane	506
Calegari, Daniel	330
Campos, Ursula	298
Cao, Gaofeng	50
Cao, Min	7
Carreiro Da Silva, Bruno	304, 372
Castro, John W.	182
Chang, Shikuo	622
Chanin, Rafael	704
Chen, Hui	56
Chen, Rong	703
Chen, Xiao Hong	292

Chen, Yuxin	1
Cheng, Can	193
Chondamrongkul, Nacha	23
Chu, William	397
Chunhua, Li	427
Clancy, Kadie	622
Conte, Tayana	68, 298, 376, 451
Costa, Alexandre	474
Cristo, Marco	68
Cunha, Francisco	221
D'Ornelas Filipakis Souza, Cristina	502
Da Costa, Arthur Fortes	697
Da Silva, Raissa	40
de Lima Silva, Luís Alvaro	74
de Oliveira Cardoso, Maxwell	34
de Souza, Clarisse	298
Delahaye, David	199
Delgado, Andrea	330
Deng, Lin	542
Di, Zhang	165
Dias Canedo, Edna	34
Dilorenzo, Ednaldo	496
Do, Thanh-Nghi	64
Dong, Wei	604
Dong, Zhijiang	456
Dorneles Soares, Heder	215
Du, De Hui	292
Du, Dongdong	263
Du, Qingfeng	175
Duan, Yucong	46
Durao, Frederico	697
Escalante Ramírez, Boris	225
Exman, Iaakov	110, 122
Fakhfakh, Faten	348
Fan, Yuyou	50
Fang, Yucheng	342
Far, Behrouz	360
Farias, Kleinner	304, 326, 372
Farias, Mário	681
Fei, Yuan	169, 659
Fernandes, Paulo	364
Figueiredo, Eduardo	314
Filho, Emanuel Dantas	496
Fontoura, Lisandra	161
Freire, Arthur	40, 474

Fu, Yujian	456
Fuchs, Andreas	647
Galdino, Sérgio	710
Gao, Honghao	7
Gao, Jerry	256
Gao, Qing	263
García, Félix	330
Ge, Ning	512, 518
Giachetti, Giovanni	116
Gonçales, Lucian	304, 326, 372
Gorgônio, Kyller	40
Guimarães, Everton	304, 372
Guimarães, Gleyser	40
Guo, Junxia	548
Guo, Shikai	703
Guo, Yao	665
Guo, Yi	80
Hackney, William	409
Hadj Kacem, Ahmed	348
Hanakawa, Noriko	687
Harush, Avihu	122
He, Hui	137
He, Xudong	397, 456
He, Yu	175
Heinrich, Robert	104, 336
Heitor Bordini, Rafael	74
Hendijani Fard, Fatemeh	360
Higashi, Kazuyuki	125
Holvitie, Johannes	506
Hong, Zhong	381
Hou, Fu	236, 282
Hu, Vincent	586
Huai, Beibei	320
Huang, Congyu	1
Huchard, Marianne	199
Ibarguengoitia González, Guadalupe	225
Ieti, Michael	246
Ishii, Nobukazu	641
Jang, Dongsoo	155
Jia, Weixi	604
Jiang, Hao	17
Jiang, Jianmin	381
Jiang, Jing	512
Jin, Hai	616

Jin, Peiquan	230
Kacker, Raghu	586
Kagdi, Huzefa	598
Kais, Ben Salah	530
Kassarnig, Valentin	629
Khaled, Ghedira	530
Khan, Mohammed Salman	486
Kishi, Tomoji	675
Kolp, Manuel	468
Kuang, Li	50
Kuhn, Rick	586
Kumar, Lov	421
Larenas, Felipe	116
Lau, Kung-Kiu	572
Le Borgne, Alexandre	199
Lehmbecker, Turner	188
Lei, Jeff	586
Leppänen, Ville	506
Li, Bing	165, 193
Li, Chunlei	1
Li, Gaofeng	616
Li, Haoming	480
Li, Jiechu	175
Li, Qingshan	707
Li, Shenzhi	604
Li, Tao	610
Li, Wenbo	320
Li, Wenrui	256
Li, Xiao	566
Li, Yanhui	415
Li, Ying	56
Li, Youhuizi	7
Li, Zengyang	165, 193
Li, Zheng	548
Li, Zhixing	143
Liang, Bin	433
Liang, Peng	165, 193
Lima, Crescencio	681
Lin, Lan	635
Lin, Li	671
Lin, Yishuai	707
Lin, Yiwei	578
Litvak, Claudia	524
Liu, Chiyu	610
Liu, Jinze	143
Liu, Kaixin	86

Liu, Lifei	598
Liu, Peini	236, 282
Liu, Yan	578
Liu, Zhe	209
Liu, Zheng	610
Llerena, Lucrecia	182
Longo, Douglas Hiura	592
Lopes Leite, Leticia	34
Lopes, Adriana	298
Lu, Yuteng	270
Lucena, Carlos	203, 221
Luo, Ping	354
Lyu, Mengyuan	230
M Sunil, Jinu	421
Ma, Truong-Thanh	64
Malucelli, Andreia	560
Man, Wu	427
Manica, Mateus	304
Manoel, Fabian	242
Manzato, Marcelo	697
Manzoni Fontoura, Lisandra	74
Mao, Xinjun	209, 236, 282
Mao, Xinya	671
Marijan, Dusica	536
Marques, Anna Beatriz	376
Marques, Leonardo	451
Marques-Neto, Humberto	314
Martinuzzi De Lima, Renata	161
Martoglia, Riccardo	98
Marín, Beatriz	116
Massitela, Ildo	364
Meireles, André	40
Mendonça, Manoel	462, 506, 681
Mensah, Solomon	368
Morales Trujillo, Miguel Ehécatl	225
Mortágua Pereira, Óscar	276
Mosbah, Mohamed	348
Moura Costa, Antonio Alexandre	149
Mumtaz, Majid	354
Máximo, Eduardo	492
Nakagawa, Hiroyuki	125, 641
Nakamura, Walter	451
Nascimento, Nicolas	704
Nawaz, M. Saqib	391
Neti, Lalita Bhanu Murthy	421
Neumann, Milena	104

Nguyen Huynh Anh, Vu	468
Nguyen, Doan	718
Nguyen, Huu-Hoa	64
Ni, Jian	671
Niu, Zhendong	131
Nogle, Jacob	372
Nwokeji, Joshua	714
Nyamawe, Ally	131
Obana, Masaki	687
Offutt, Jeff	542
Oktaba, Hanna	225
Olagunju, Ayodele	714
Oliveira, Edson	68
Oliveira, Johnatan	314, 445
Oshima, Naito	675
Pantoja, Carlos	215, 242
Paranhos, Ricardo	710
Parente Da Costa, Ruyther	34
Passini, William	286
Passos, Amanda	681
Pathirage, Don	155
Peng, Shuai	188
Pereira Aranha, Dandara	34
Perkusich, Angelo	40, 149, 474, 496
Perkusich, Mirko	40, 149, 474, 496
Pinheiro, Vladia	492
Pompermaier, Leandro	704
Prikladnicki, Rafael	704
Qi, Tianmei	80
Qi, Yong	137
Qi, Yuanchao	711
Qian, Ju	582
Queiroz, Randerson	376
Raghunathan, Janani	598
Ramos, Felipe	149, 474
Rao, Qifan	137
Rechia Machado, Nielsen Luiz	74
Reddivari, Sandeep	409, 486
Regateiro, Diogo	276
Reinehr, Sheila	560
Rios, Nicolli	462, 506
Rivero, Luis	451
Rodríguez, Nancy	182
Rohella, Anshuman	308

Rosemberg, Marcio	221
Rossi, Gustavo	524
Rouahi, Aouatef	530
Sales, Afonso	704
Santos, Alan	364, 704
Santos, Danilo	496
Santos, José Amâncio	462
Santos, Mateus	314, 445
Santos, Welligton	710
Sastre, Jefry	203
Seghrouchni, Amal El Fallah	215
Sen, Sagar	536
Shanly, Catherine	246
She, Feiyan	711
Shi, Kun	175
Shi, Zhendong	354
Shin, Michael	155
Shrestha, Roshan	188
Sirqueira, Tassio	221
Siyao, Wang	92
Sizhe, Ye	427
Slama, Anja	360
Song, Fengguang	635
Song, Zhengyang	46
Souza Cabral, Bruno	697
Souza de Jesus, Vinicius	242
Spinola, Rodrigo	462, 506, 681
Stoffel, Kilian	29
Sun, Caihong	439
Sun, Jing	23, 60, 246
Sun, Linjie	548
Sun, Meng	270, 385, 391
Sun, Xiaobing	46
Sun, Xin	635
T. Marques-Neto, Humberto	445
Tabia, Karim	64
Takada, Shingo	308
Tian, Bing	86
Tounsi, Mohamed	348
Toure, Fadel	653
Tsuchiya, Tatsuhiro	125, 641
Urtado, Christelle	199
Valentim, Natasha	68, 451
Vauttier, Sylvain	199



Viana, Marx	203, 221
Vigliato, Markos	314
Vilain, Patrícia	592
Virgulino Ribeiro, Tayse	502
Viterbo, José	215, 242
von Hof, Vincent	647
Wan, Shouhong	230
Wang, Chuanqi	415
Wang, Hao	433
Wang, Haoyu	665
Wang, Huaimin	566
Wang, Huiwen	169, 342, 659
Wang, Kenian	716
Wang, Long	582
Wang, Meijia	707
Wang, Meiling	320
Wang, Ran	665
Wang, Tao	143, 566
Wang, Weiwei	548
Wang, Xin	175
Wang, Yan	292
Wang, Yi	60
Wang, Zhihong	80
Wang, Zhuangzhuang	691
Wang, Ziyuan	480, 711, 720
Warren, Ian	23, 246
Wautelet, Yves	468
Wei, Bingyang	60
Wei, Miaomiao	703
Wen, Junye	554
Winograd, Yakir	122
Wotawa, Franz	629
Wu, Qiansheng	320
Xi, Meng	17, 56
Xia, Bin	610
Xiangyu, Xi	263
Xiao, Lili	169
Xie, Wenguang	716
Xing, Chunxiao	86, 250
Xu, Baowen	415, 433
Xu, Dianxiang	188
Xu, Guoai	665
Xu, Jie	250
Xu, Jincheng	175
Xu, Luhang	604
Xu, Rongfei	512, 518

Xue, Yufeng	635
Yaga, Dylan	586
Yan, Jinpei	137
Yang, Chen	578
Yang, Guowei	554
Yang, Qing	11
Yang, Shuo	209
Yang, Xiaoxian	46
Yao, Xianhe	11
Yendluri, Akhil	622
Yin, Gang	143
Yin, Jianwei	56
Yin, Liangze	604
Yin, Yuyu	17
Yu, Jia	578
Yu, Jie	566
Yu, Li	691
Yu, Min	711
Yu, Xiaofei	381
Yu, Yue	143
Yuan, Pingpeng	616
Yue, Lihua	230
Zeng, Lingbin	566
Zhang, Chengpeng	665
Zhang, Chuan	1
Zhang, Gefei	403
Zhang, Guigang	250
Zhang, Huan	50
Zhang, Jianbiao	671
Zhang, Jingwei	11
Zhang, Li	512, 518
Zhang, Meina	439
Zhang, Rong	17
Zhang, Shikun	263
Zhang, Shuai	236, 282
Zhang, Tao	480
Zhang, Tong	263
Zhang, Xiaofang	433
Zhang, Xinyue	46
Zhang, Xiyue	385
Zhang, Xunhui	566
Zhang, Yong	86, 250
Zhang, Zhou	230
Zhang, Zijian	1
Zhao, Guoliang	263
Zhao, Jianjun	403

Zhao, Ruilian	548
Zhao, Wen	263
Zhong, Wen	292
Zhongwang, Fu	427
Zhou, JinPeng	622
Zhou, Qian	433
Zhou, Sijing	7
Zhou, Xiaoyu	582
Zhu, Huibiao	169, 342, 659
Zhu, Liehuang	1
Ziyi, Ma	427
Zou, Quan	46
Zuo, Meiyun	439

## Program Committee

Silvia T. Acuña	Universidad Autonoma de Madrid
Shadi Alawneh	Oakland University
Taisira Albalushi	Sultan Qaboos University
Mark Allison	University of Michigan-Flint
Vaibhav Anu	North Dakota State University
John Anvik	Department of Mathematics and Computer Science, University of Lethbridge
Doo-Hwan Bae	Korea Advanced Institute of Science and Technology
Hamid Bagheri	University of California, Irvine
Xiaoying Bai	Tsinghua University
Ellen Barbosa	ICMC/USP
Fevzi Belli	Univ. Paderborn
Ateet Bhalla	Independent Consultant, India
Alessandro Bianchi	Department of Informatics - University of Bari
Guoray Cai	The Pennsylvania State University
Keith C.C. Chan	The Hong Kong Polytechnic University
Chih-Hung Chang	College of Computing and Informatics, Providence University
Lily Chang	University of Wisconsin, Platteville
Rong Chang	IBM
Shikuo Chang	University of Pittsburgh
Wen-Hui Chen	National Taipei University of Technology
Kim-Kwang Raymond Choo	The University of Texas at San Antonio, USA
William Chu	Department of Computer Science and Information Engineering, TungHai University
Stelvio Cimato	University of Milan
Fabio Costa	Federal University of Goias
Jose Luis de La Vara	Carlos III University of Madrid
Peng Di	The University of New South Wales
Zhijiang Dong	Middle Tennessee State University
Weichang Du	University of New Brunswick
Christof Ebert	Vector
Ali Ebneenasir	Michigan Technological University
Magdalini Eirinaki	Computer Engineering Dept, San Jose State University
Omar El Ariss	Texas A&M University-Commerce
Abdelrahman Elfaki	University of Tabuk
Ruby Elkharboutly	Quinnipiac University
Iaakov Exman	JCE - The Jerusalem College of Engineering - Azrieli
Behrouz Far	University of Calgary
Liana Fong	IBM T. J. Watson Research
Honghao Gao	Shanghai University
Kehan Gao	Eastern Connecticut State University
Kun Gao	zhejiang business technology institute
Ignacio García	University of Castilla-La Mancha
Raúl García-Castro	Universidad Politécnica de Madrid
Swapna Gokhale	Univ. of Connecticut
Wolfgang Golubski	Westfälische Hochschule Zwickau

Anurag Goswami	Bennett University
Des Greer	Queen's University Belfast
Christiane Gresse von Wangenheim	Federal University of Santa Catarina - UFSC
Katarina Grolinger	University of Western Ontario
Hao Han	The University of Tokyo
Xudong He	Florida International University
Rubing Huang	Jiangsu University
Shihong Huang	Florida Atlantic University
Basseyy Isong	University of Venda
Clinton Jeffery	University of Idaho
Jason Jung	Chung-Ang University
Pankaj Kamthan	Concordia University
Ananya Kanjilal	Department of CSE, B.P.Poddar Institute of Management & Technology, Kolkata-52
Ajay Kattapur	TCS
Taghi Khoshgoftaar	Florida Atlantic University
Jun Kong	North Dakota State University
Aneesh Krishna	Curtin University, Australia
Li Kuang	Central South University
Vinay Kulkarni	Tata Consultancy Services Research
Olivier Le Goear	LIUPPA, Université de Pau et des Pays de l'Adour
Meira Levy	Shenkar Engineering, Design, Art
Bixin Li	SOUTHEAST UNIVERSITY
Yuan-Fang Li	Monash University
Zhi Li	College of Computer Science and Information Technology, Guangxi Normal University
Jianhua Lin	Eastern Connecticut State University
Lan Lin	Department of Computer Science, Ball State University
Shih-Hsi Liu	California State University, Fresno
Ting Liu	Xi'an Jiaotong University
Xiaodong Liu	Edinburgh Napier University
Luanna Lopes Lobato	Universidade Federal de Pernambuco - UFPE
Baojun Ma	School of Economics and Management, Beijing University of Posts and Telecommunications
Ivan Machado	UFBA - Federal University of Bahia
Marcelo Maia	UFU
Riccardo Martoglia	FIM - University of Modena
Beatriz Marín	Universidad Diego Portales
Santiago Matalonga	University of the West of Scotland
Andre Menolli	Universidade Estadual do Norte do Paraná - UENP
Ali Mili	NJIT
Alok Mishra	Atilim University, Incek 06836, Ankara - Turkey
Óscar Mortágua Pereira	University of Aveiro
Hiroyuki Nakagawa	Osaka University
Nanjangud Narendra	Ericsson Research
Alex Norta	Department of Informatics, Tallinn University of Technology
Amjad Nusayr	UHV

Edson Oliveirajr Angelo Perkusich	State University of Maringá Electrical Engineering Department, Federal University of Campina Grande
Antonio Piccinno Alfonso Pierantonio Rick Rabiser	University of Bari University of L'Aquila Christian Doppler Lab. MEVSS, Johannes Kepler University Linz
Claudia Raibulet Damith Rajapakse Rajeev Raje Henrique Rebêlo Marek Reformat Stephan Reiff-Marganiec Hassan Reza Ivan Rodero Daniel Rodriguez Samira Sadaoui Seyed Masoud Sadjadi Claudio Sant'Anna Klaus-Dieter Schewe Abdelhak Seriai Michael Shin Martin Solari George Spanoudakis Vijayan Sugumaran	University of Milano-Bicocca National University of Singapore IUPUI Federal University of Pernambuco - UFPE University of Alberta Department of Computer Science, University of Leicester University of North Dakota Rutgers University The University of Alcalá University of Regina Florida International University Federal University of Bahia Software Competence Center Hagenberg Lirmm/université Montpellier 2 Texas Tech University Universidad ORT Uruguay Department of Computer Science, City University School of Business Administration, Oakland University, Rochester, MI 48309, USA
Jing Sun Meng Sun Yanchun Sun Gerson Sunyé Kumiko Tadano Chuanqi Tao Joe Tekli Mark Trakhtenbrot Burak Turhan Christelle Urtado Sylvain Vauttier Gennaro Vessio Sergiy Vilkomir Arndt Vonstaa Huanjing Wang Xiaoyin Wang Ye Wang Yong Wang Zhongjie Wang Ziyuan Wang Hironori Washizaki Bingyang Wei	The University of Auckland Peking University Peking University Université de Nantes NEC Corporation Nanjing University of Aeronautics and Astronautics Lebanese American University Holon Institute of Technology Brunel University LGI2P - Ecole des Mines d'Alès LG2IP / Ecole des Mines d'Alès University of Bari East Carolina University PUC-Rio Western Kentucky University University of Texas at San Antonio Zhejiang Gongshang University Texas A&M University Harbin Institute of Technology Nanjing University of Posts and Telecommunications Waseda University Midwestern State University

Xiao Wei	Shanghai University
Guido Wirtz	University of Bamberg
Franz Wotawa	Technische Universitaet Graz
Peng Wu	State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences
Qing Wu	College of Computer Science and Technology, Hangzhou Dianzi University
Dianxiang Xu	Boise State University
Haiping Xu	University of Massachusetts Dartmouth
Lai Xu	Bournemouth University
Weifeng Xu	Bowie State University
Guowei Yang	Texas State University
Hongji Yang	Bath Spa University
Yuyu Yin	HDU
Huiqun Yu	East China University of Science and Technology
Du Zhang	California State University
Pengcheng Zhang	College of Computer and Information, Hohai University
Shunxiang Zhang	Anhui University of Science & Technology
Yong Zhang	Tsinghua University
Zhenyu Zhang	State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences
Xu Zheng	Shanghai University
Zhigao Zheng	HUST
Nianjun Zhou	IBM
Hong Zhu	Oxford Brookes University
Huibiao Zhu	East China Normal University
Xingquan Zhu	Florida Atlantic University
Eugenio Zimeo	University of Sannio

## Additional Reviewers

Alharbi, Mohammed  
Allian, Ana Paula

Baek, Youngmin  
Barat, Souvik  
Basciani, Francesco  
Bauder, Richard  
Bublitz, Frederico

Carbonnel, Jessie  
Cassano, Fabio  
Castro, John W.  
Cheng, Yuxia

de Andrade, Paulo Roberto Martins  
de Groof, Richard  
Deval, Vipin  
Di Rocco, Juri  
Díaz-Vico, David

Elkharboutly, Ruby

Fioravanti, Maria Lydia

Gallege, Lahiru  
Ge, Ruiquan  
Ghosh, Aritra

Kane, Shridhar  
Kang, Taeghyun

Le Borgne, Alexandre  
Li, Yi  
Li, Zijie  
Liu, Yuzhen

Magües, Daniel A.  
Malhotra, Ruchika  
Marcolino, Anderson  
Mujumdar, Anusha

Pandey, Saurabh  
Perkusich, Mirko



Qayumi, Karima

Reza, Hassan  
Richter, Aaron  
Rocca, Ignacio  
Russo, Juan Pablo

Saay, Salim  
Sunkle, Sagar

Tibermacine, Chouki

Umuhoza, Eric

Villalobos, J. J.

Wen, Junye  
Wu, Xundong

Yan, Jun  
Yan, Rongjie

Zhang, Donghong  
Zhang, Long  
Zhang, Xiyue